

DISSERTATION

for the degree of

Doctor of Natural Sciences (Dr. rer. nat.)

Frameworks in Medical Image Analysis with Deep Neural Networks

Dominik Müller



University of Augsburg
Department of Computer Science
IT-Infrastructure for Translational Medical Research
December 2022

Examiner:

Supervisor: Prof. Dr. Frank Kramer
Department of Computer Science,
University of Augsburg, Germany

Secondary Advisor: Prof. Dr. Bernhard Bauer
Department of Computer Science,
University of Augsburg, Germany

Tertiary Advisor: Prof. Dr. Tim Beißbarth
Department of Medical Bioinformatics,
University Medical Center Göttingen, Germany

Examination Board:

Prof. Dr. Frank Kramer Prof. Dr. Bernhard Bauer

Prof. Dr. Elisabeth André Prof. Dr. Tim Beißbarth

Defense: 17. March 2023

Abstract

In recent years, deep neural network based medical image analysis has become quite powerful and achieved similar results performance-wise as experts. Consequently, the integration of these tools into the clinical routine as clinical decision support systems is highly desired. The benefits of automatic image analysis for clinicians are massive, ranging from improved diagnostic as well as treatment quality to increased time-efficiency through automated structured reporting. However, implementations in the literature revealed a significant lack of standardization in pipeline building resulting in low reproducibility, high complexity through extensive knowledge requirements for building state-of-the-art pipelines, and difficulties for application in clinical research.

The main objective of this work is the standardization of pipeline building in deep neural network based medical image segmentation and classification. This is why the Python frameworks MIScnn for medical image segmentation and AUCMEDI for medical image classification are proposed which simplify the implementation process through intuitive building blocks eliminating the need for time-consuming and error-prone implementation of common components from scratch. The proposed frameworks include state-of-the-art methodology, follow outstanding open-source principles like extensive documentation as well as stability, offer rapid as well as simple application capabilities for deep learning experts as well as clinical researchers, and provide cutting-edge high-performance competitive with the strongest implementations in the literature.

As secondary objectives, this work presents more than a dozen in-house studies as well as discusses various external studies utilizing the proposed frameworks in order to prove the capabilities of standardized medical image analysis. The presented studies demonstrate excellent predictive capabilities in applications ranging from COVID-19 detection in computed tomography scans to the integration into a clinical study workflow for Gleason grading of prostate cancer microscopy sections and advance the state-of-the-art in medical image analysis by simplifying experimentation setups for research. Furthermore, studies for increasing reproducibility in performance assessment of medical image segmentation are presented including an open-source metric library for standardized evaluation and a community guideline on proper metric usage.

The proposed contributions in this work improve the knowledge representation of the field, enable rapid as well as high-performing applications, facilitate further research, and strengthen the reproducibility of future studies.

Zusammenfassung

In den letzten Jahren ist die auf tiefen neuronalen Netzwerken basierende medizinische Bildanalyse besonders leistungsfähig geworden und konnte ähnliche Vorhersagegenauigkeiten wie Experten erzielen. Daher ist die Integration dieser Werkzeuge in die klinische Routine als Entscheidungsunterstützungssysteme besonders erhofft. Die Vorteile der automatischen Bildanalyse für Kliniker sind enorm und reichen von verbesserter Diagnose- und Behandlungsqualität bis hin zu erhöhter Zeiteffizienz durch automatisierte strukturierte Befundung. Die Implementierungen in der Literatur zeigten jedoch einen erheblichen Mangel an Standardisierung bei der Erstellung von Pipelines, was zu geringer Reproduzierbarkeit, hoher Komplexität durch umfangreiche Wissensanforderungen für den Bau von modernen Pipelines und Schwierigkeiten bei der Anwendung in der klinischen Forschung führte.

Das Hauptziel dieser Arbeit ist die Standardisierung des Pipeline-Baus für die Segmentierung und Klassifizierung medizinischer Bilder mittels tiefer neuronaler Netze. Aus diesem Grund werden die Python-Frameworks MIScnn für die Segmentierung medizinischer Bilder und AUCMEDI für die Klassifizierung medizinischer Bilder vorgestellt, die den Implementierungsprozess durch intuitive Bausteine vereinfachen und die zeitaufwändige als auch fehleranfällige Implementierung gängiger Komponenten überflüssig machen. Die vorgeschlagenen Frameworks beinhalten modernste Methodik, folgen herausragenden Open-Source Prinzipien wie ausführliche Dokumentation sowie Stabilität, bieten schnelle und einfache Anwendungsmöglichkeiten für Deep-Learning Experten sowie klinische Forscher und bieten wegberaubende Hochleistung in Sachen Genauigkeit, welche sich mit den stärksten Implementierungen in der Literatur messen kann.

Als sekundäre Ziele werden in dieser Arbeit mehr als ein Dutzend interner Studien vorgestellt sowie verschiedene externe Studien diskutiert, die die vorgeschlagenen Frameworks nutzen, um das Potenzial der standardisierten medizinischen Bildanalyse zu beweisen. Die vorgestellten Studien demonstrieren hervorragende Vorhersagefähigkeiten in Anwendungen, die von der COVID-19-Erkennung in Computertomographie-Scans bis zur Integration in einen klinischen Studien-Workflow zur Gleason-Einstufung von Prostatakrebs-Mikroskopie-Schnitten reichen, und bringen den Stand der Technik in der medizinischen Bildanalyse mittels Vereinfachung der Durchführung von Experimenten für die Forschung voran. Darüber hinaus werden Studien zur Verbesserung der Reproduzierbarkeit bei der Leistungsbewertung der medizinischen Bildsegmentierung vorgestellt, einschließlich einer Open-Source Metrik-Bibliothek zur standardisierten Auswertung und Community-Richtlinien zur korrekten Verwendung von Metriken.

Die in dieser Arbeit erbrachten Beiträge verbessern die Wissensrepräsentation des Fachgebiets, ermöglichen schnelle und leistungsstarke Anwendungen, erleichtern die weitere Forschung und stärken die Reproduzierbarkeit zukünftiger Studien.

Acknowledgments

Many people have contributed to making it possible for me to be where I am today and being able to do something that I not only believe in but also have a great interest in.

First of all, I would like to thank my supervisor Prof. Dr. Frank Kramer for supporting me as well as my work, for giving me freedom in research as well as education, for pushing me forward to pursue my ideas, for the ambitions to acquire the broad resources needed for my research, and for giving me the opportunity to work at the MISIT lab. Thank you for providing the best and most open working environment that one could wish for a dissertation. I would also like to thank Prof. Dr. Bernhard Bauer for the general support, for providing us shelter at the start before we moved to our own lab, offering me hardware resources to start my journey in deep neural network based medical image analysis, for introducing me to Frank which made all this even possible, and for being my advisor for this dissertation. Furthermore, I would like to thank Prof. Dr. Tim Beißbarth and Prof. Dr. Elisabeth André for the review of this dissertation. Consequently, I want to thank the many people who helped me in this dissertation through proof-reading: Dennis Hartmann, Dennis Klonek, Nadja Bramkamp, and Remo Griesbaum.

I would also like to thank my colleagues at the MISIT lab, other labs at the University of Augsburg, and at the University Hospital Augsburg. The wonderful, hilarious, and relaxed atmosphere has made my workplace a place that I am happy to go to every day. In particular, I would like to thank Dennis H., Dr. Florian Auer, Dr. Iñaki Soto Rey, Johann Frei, and Peter Parys for supporting me through giving excellent advice on problems, enduring fundamental discussions, proof-reading publications, and the many joyful moments during my Ph.D. journey. In particular, thank you Peter and Johann for keeping all the big machines running. I want to thank the students at the MISIT lab like Adrian Pfleiderer, Philip Meyer, Simone Mayer, and formerly Dennis H. for their outstanding contributions to the projects presented in this dissertation. I am also thankful for the great collaboration with Silvan Mertes, Niklas Schröter, Fabio Hellmann, and Miriam Elia. Furthermore, I want to thank Iñaki, Dr. Johannes Raffler, as well as Prof. Dr. Ralf Huss for giving me a second foothold at the MeDIZ at the University Hospital Augsburg and offering me insights into conducting clinical application studies. Moreover, I want to thank my former professors Prof. Dr. Dmitrij Frishman and Prof. Dr. Hans-Werner Mewes at the Technical University of Munich for starting my interest in research.

I would also like to thank my friends and close ones: Florian, Johann, Remo, Simone, Angela Beer, Stephan Brandl, Christoph Wolf, Felix Weiland, and Katharina Grunau. Thank you for your support by listening to my worries, motivating me, cheering me up in tough times, and having so many great activities as well as moments together. Finally, I want to thank my family: My brother André Rogozik for always being there for me and my parents Edmund Müller as well as Birgit Müller-Rogozik for backing me up and constantly supporting me in anything I can imagine. Thank you for everything!

Table of Contents

Abstract	I
Zusammenfassung	II
Acknowledgments	III
Table of Contents	IV
List of Figures	VIII
List of Tables	XI
List of Code Snippets	XII
Nomenclature	XIII
1 Introduction	1
1.1 Motivation	2
1.2 Challenges	5
1.3 Research Objectives and Contributions	7
1.3.1 Objective I: Development of Frameworks for standardized Pipeline Building	7
1.3.2 Objective II: Application Studies demonstrating the Capabilities of the proposed Frameworks	7
1.3.3 Objective III: Research Studies utilizing the proposed Frameworks to advance the state-of-the-art of Medical Image Analysis	8
1.3.4 Objective IV: Improvement of Reproducibility in Medical Image Segmentation	8
1.3.5 Grouping of Contributions into Research Fields	8
1.4 Thesis Outline	10
1.5 Publications	12
2 Background	15
2.1 Medical Imaging	16
2.1.1 Imaging Modalities	16
2.1.2 Digital Imaging Data	20
2.2 Artificial Neural Networks	22
2.2.1 Theory	22
2.2.2 Supervised Learning	23
2.2.3 Deep Learning	24
2.2.4 Convolutional Neural Networks	25
2.3 Computer Vision	28
2.3.1 Image Classification	28
2.3.2 Image Segmentation	31
2.3.3 Hardware Requirements	34
2.4 Medical Image Analysis	36
2.4.1 Research Fields	37
2.4.2 Clinical Decision Support	39
3 Workflow in Medical Image Analysis	41
3.1 Data Management	43
3.1.1 Imaging Formats in Medicine	43
3.1.2 Image Dimensions	44
3.1.3 Image Information: Representation, Annotation, and Metadata	45
3.2 Image Augmentation	49
3.2.1 Types of Image Augmentation	50
3.2.2 Application Techniques	51

3.3 Image Preprocessing.....	52
3.3.1 One Hot Encoding	52
3.3.2 Normalization	53
3.3.3 Image Filtering.....	54
3.3.4 Bias Correction in Medical Imaging	54
3.3.5 Clipping	55
3.3.6 Padding	55
3.3.7 Cropping	56
3.3.8 Resizing	56
3.3.9 Resampling	57
3.3.10 Patching	57
3.4 Neural Network Model.....	59
3.4.1 Hyperparameters.....	59
3.4.2 Training and Prediction Process.....	61
3.4.3 Frameworks	64
3.5 Performance Assessment.....	67
3.5.1 Evaluation Metrics	67
3.5.2 Loss Functions	70
3.6 Sampling and Validation Strategies	73
3.6.1 Sampling Strategies	73
3.6.2 Validation Monitoring	75
4 Medical Image Segmentation	77
4.1 History and Current State	78
4.1.1 Methods	78
4.1.2 Challenges.....	81
4.2 MIScnn: a Framework for Medical Image Segmentation with Convolutional Neural Networks.....	82
4.2.1 Implementation	83
4.2.2 Open-Source Development and Deployment.....	88
4.2.3 API Usage.....	96
4.2.4 Community Contributions, Support, and Popularity	101
4.3 Study: Kidney Tumor Segmentation	103
4.3.1 Dataset	103
4.3.2 Application	104
4.3.3 Results and Discussion	107
4.3.4 Conclusions.....	108
4.4 Study: Nucleus Segmentation and Analysis in Breast Cancer.....	110
4.4.1 Dataset	110
4.4.2 Application	111
4.4.3 Results and Discussion	113
4.4.4 Conclusions.....	116
4.5 Study: Segmentation of COVID-19 Lung Infection based on limited Data	117
4.5.1 Dataset	117
4.5.2 Related Work	119
4.5.3 Application	120
4.5.4 Results and Discussion	124
4.5.5 Conclusions.....	130
5 Medical Image Classification	131
5.1 History and Current State	132
5.1.1 Methods	132
5.1.2 Challenges.....	135
5.2 AUCMEDI: a Framework for Automated Classification of Medical Images	136
5.2.1 Implementation	137
5.2.2 Open-Source Development and Deployment.....	143
5.2.3 API Usage.....	151

5.2.4 Community Contributions, Support, and Popularity	157
5.3 Study: Standardized Image Classification across Medical Disciplines	159
5.3.1 Datasets.....	159
5.3.2 Application	161
5.3.3 Results and Discussion	161
5.3.4 Conclusions.....	165
5.4 Study: Multi-Disease Detection in Retinal Imaging	166
5.4.1 Dataset	166
5.4.2 Application	167
5.4.3 Results and Discussion	170
5.4.4 Conclusions.....	173
5.5 Study: COVID-19 Infection and Severity Prediction.....	174
5.5.1 Dataset	174
5.5.2 Application	176
5.5.3 Results and Discussion	179
5.5.4 Conclusions.....	182
5.6 Study: Informative Value of Explainable AI.....	183
5.6.1 Clinical Study Design	184
5.6.2 Dataset	185
5.6.3 Application	187
5.6.4 Current Outcomes and Insights.....	188
5.6.5 Future Outlook.....	192
6 Ensemble Learning	193
6.1 Idea.....	194
6.2 Methods.....	195
6.2.1 Ensemble Learning Strategies.....	195
6.2.2 Pooling Functions	197
6.3 Challenges	199
6.4 Study: Performance Impact in Image Segmentation	200
6.4.1 Dataset	200
6.4.2 Application	201
6.4.3 Results and Discussion	201
6.4.4 Conclusions.....	203
6.5 Study: Performance Impact in Image Classification	204
6.5.1 Datasets.....	204
6.5.2 Application	206
6.5.3 Results	209
6.5.4 Discussion.....	214
6.5.5 Conclusions.....	217
7 Reproducibility of Performance Assessment	218
7.1 MISEval: a Metric Library for Medical Image Segmentation Evaluation	219
7.1.1 Implementation	219
7.1.2 Quantitative Evaluation	222
7.1.3 Qualitative Evaluation	222
7.1.4 Discussion.....	223
7.2 Towards a Guideline for Evaluation Metrics	224
7.2.1 Bias: Class Imbalance	224
7.2.2 Bias: Influence of the Region-of-Interest Size	225
7.2.3 Bias: Influence of the Segmentation Task.....	226
7.2.4 Bias: Multi-class Evaluation	227
7.2.5 Proposed Guideline.....	227
7.2.6 Experiments on Metric Behavior	228
7.3 Performance Assessment in Presence of Control Samples.....	230
7.3.1 Proposed Metric: MISm.....	230

7.3.2 Theoretical Analysis	232
7.3.3 Weighting Coefficient Analysis.....	232
7.3.4 Experimental Application	233
7.3.5 Discussion.....	234
8 Discussion	235
8.1 Advancements in Medical Image Segmentation	236
8.1.1 The proposed Framework: MIScnn	237
8.1.2 Scientific Impact and Contributions.....	241
8.1.3 Limitations and further Challenges.....	247
8.2 Advancements in Medical Image Classification	249
8.2.1 The proposed Framework: AUCMEDI.....	250
8.2.2 Scientific Impact and Contributions.....	256
8.2.3 Limitations and further Challenges.....	258
8.3 Advancements in further Research Fields	260
8.3.1 Ensemble Learning	260
8.3.2 Reproducibility of Performance Assessment	261
8.4 Software Integration Process: From University Research to Clinical Application.....	264
8.4.1 Challenges of Integration.....	264
8.4.2 Chances by Interoperability in Medical Imaging.....	265
8.4.3 Design of a Clinical Application Study.....	266
8.5 Future Work and Outlook.....	269
9 Conclusions	271
References	274

List of Figures

Figure 1.1: Literature search for publications in relevant fields for medical image analysis via PubMed.....	3
Figure 1.2: Overview of the objectives and contributions presented in this dissertation.	9
Figure 2.1: Example visualizations of radiography based imaging modalities and corresponding devices.....	17
Figure 2.2: Example visualization of magnetic resonance imaging and an MRI scanner.....	18
Figure 2.3: Example visualization of ultrasound imaging.....	19
Figure 2.4: Example visualization of four modalities based on visible light imaging.	19
Figure 2.5: Representation of a digital image by an image and volume matrix.	21
Figure 2.6: Structure of an artificial neuron.	22
Figure 2.7: Overview of different neural network topologies.	24
Figure 2.8: Illustration of the convolutional transfer function.	25
Figure 2.9: Illustration of a maximum pooling operation on a two-dimensional input matrix.	26
Figure 2.10: Architecture of a convolutional neural network.	27
Figure 2.11: Illustration of image classification and segmentation in computer vision.	28
Figure 2.12: Image classification architecture overview (selection) and comparison by number of parameters.	30
Figure 2.13: Setup of an image classification model including backbone and classification head structure.	31
Figure 2.14: Image segmentation architecture overview (selection) and comparison by number of parameters.	32
Figure 2.15: Illustration of the U-Net architecture.....	33
Figure 3.1: Illustration of a typical workflow in deep learning based medical image analysis.	41
Figure 3.2: Illustration of quality flaws in annotations for medical image analysis.....	47
Figure 3.3: Overview on common image augmentation techniques.	50
Figure 3.4: Visualization of the one hot encoding procedure.	53
Figure 3.5: Application overview of multiple image preprocessing techniques.	56
Figure 3.6: Illustration of a simple patching application on a CT scan.....	58
Figure 3.7: Overview of important hyperparameters in a MIA pipeline.....	59
Figure 3.8: Fitting curve visualizing of a typical loss function trend during a training process.	64
Figure 3.9: Illustration of sampling strategies.....	74
Figure 4.1: Thresholding approach for vessel segmentation in retinal imaging.....	79
Figure 4.2: Logo of the proposed framework MIScnn.....	82
Figure 4.3: Flowchart diagram of the MIScnn pipeline.	83
Figure 4.4: Flowchart visualization of the deep learning model selection process.	86
Figure 4.5: Starting page of the MIScnn repository on GitHub.	90
Figure 4.6: Extract of the task management system on GitHub for MIScnn.	91
Figure 4.7: Extract of the MIScnn wiki starting page on GitHub.	92
Figure 4.8: Extract of the CI pipeline logs for MIScnn on GitHub.....	94
Figure 4.9: Workflow of the four core classes in the MIScnn API.....	97
Figure 4.10: Extract of the documentation entry for the Data I/O class from the MIScnn wiki.	97
Figure 4.11: Docstring of the Data Augmentation class from the MIScnn wiki.	98

Figure 4.12: Documented argument descriptions of the Data Augmentation class from the MIScnn wiki.	98
Figure 4.13: Extract of the documentation entry for the Preprocessor class from the MIScnn wiki.....	99
Figure 4.14: Example initialization of the Preprocessor class from the MIScnn wiki.	99
Figure 4.15: Extract of the documentation entry for the Neural Network class from the MIScnn wiki.	100
Figure 4.16: Community contribution estimation for the MIScnn framework.	101
Figure 4.17: Popularity estimation for the MIScnn framework.	102
Figure 4.18: Computed tomography image from the KiTS19 dataset.	103
Figure 4.19: Fitting curves and performance assessment for the kidney tumor segmentation pipeline.	107
Figure 4.20: Visualization of annotated and predicted segmentation masks for kidney tumor segmentation. ..	108
Figure 4.21: Annotation of the same sample represented through different masks.	110
Figure 4.22: Overview of the class categories in the NuCLS dataset.	111
Figure 4.23: Class distribution of the super class annotations in the NuCLS dataset.	111
Figure 4.24: Box plots showing the result distribution for the single-rater dataset.....	113
Figure 4.25: Box plots showing the result distribution for the multi-rater dataset.....	114
Figure 4.26: Visualization of COVID-19 infected regions in a thorax CT.	118
Figure 4.27: Flowchart diagram of the COVID-19 lung infection segmentation pipeline.	120
Figure 4.28: Structure diagram of the standard 3D U-Net architecture.	122
Figure 4.29: Boxplots showing validation and testing performance for COVID-19 segmentation.	125
Figure 4.30: Fitting curves of the COVID-19 segmentation pipeline.	126
Figure 4.31: Visual comparison between radiologist annotations and the proposed segmentation pipeline.....	127
Figure 5.1: Workflow overview in comparison of traditional and modern MIC.	132
Figure 5.2: Logo of the proposed framework AUCMEDI.	136
Figure 5.3: Flowchart diagram of the AUCMEDI pipeline.	137
Figure 5.4: Overview of application environments in AUCMEDI.	142
Figure 5.5: Starting page of the AUCMEDI repository on GitHub.	145
Figure 5.6: Website of the AUCMEDI framework hosted on GitHub Pages.	146
Figure 5.7: Extract of the AUCMEDI wiki on the project website.	147
Figure 5.8: Extract of the CI pipeline logs for AUCMEDI on GitHub.	148
Figure 5.9: Extract of a release changelog of AUCMEDI on GitHub.	150
Figure 5.10: Workflow of the three pillars in AUCMEDI.	152
Figure 5.11: Extract of the documentation entry for the data interface pillar from the AUCMEDI wiki.	152
Figure 5.12: Extract of the documentation entry for the neural network pillar from the AUCMEDI wiki.....	153
Figure 5.13: Extract of the documentation entry for the data generator pillar from the AUCMEDI wiki.	154
Figure 5.14: Workflow of the AutoML module in AUCMEDI.	156
Figure 5.15: Community contribution estimation for the AUCMEDI framework.....	157
Figure 5.16: Popularity estimation for the AUCMEDI framework.....	158
Figure 5.17: Exemplary samples of the analyzed datasets in this study.....	160
Figure 5.18: Summary of the achieved performance in the conducted experiments by mean and standard deviation.	162
Figure 5.19: XAI visualization of samples from the radiology (X-ray) and dermatology dataset.	164

Figure 5.20: Flowchart diagram of the implemented MIC pipeline for multi-disease detection in retinal imaging. 168

Figure 5.21: Fitting curves showing the training and validation loss during the training process. 171

Figure 5.22: Receiver operating characteristic curves for each model type applied in the proposed pipeline... 172

Figure 5.23: Visualization of CT scans categorized by COVID-19 classification from the STOIC dataset. 175

Figure 5.24: Workflow diagram of the implemented pipeline for COVID-19 presence and severity prediction. 176

Figure 5.25: Results of the performance evaluation for COVID-19 infection and severe outcome prediction. 180

Figure 5.26: Flowchart diagram of the EKIPRO project. 184

Figure 5.27: Visualization of tiles from histological sections generated in the EKIPRO project. 186

Figure 5.28: Receiver operating characteristic curves for each architecture validated in the EKIPRO study. .. 188

Figure 5.29: Fitting curve and confusion matrix of the EKIPRO ResNeXt model. 190

Figure 5.30: Generated XAI heatmaps based on the ResNeXt model for prostate carcinoma classification. 191

Figure 5.31: Prototype of the EKIPRO browser application for visualizing microscopy and XAI imaging. 192

Figure 6.1: Overview of strategies in the field of ensemble learning. 194

Figure 6.2: Workflow illustration of deep ensemble learning strategies. 195

Figure 6.3: Computed tomography scan of the thorax from a Sars-CoV-2 positive patient. 200

Figure 6.4: Box plot showing the performance distribution of all experiments. 202

Figure 6.5: Average performance increase by ensemble learning compared to the baseline performance. 203

Figure 6.6: Exemplary samples of the four used datasets used in the analysis. 205

Figure 6.7: Visualization of performance results for the Baseline pipeline. 210

Figure 6.8: Visualization of performance results for the Augmenting pipeline. 211

Figure 6.9: Visualization of performance results for the Stacking pipeline. 212

Figure 6.10: Visualization of performance results for the Bagging pipeline. 214

Figure 6.11: Summary of all experiments on performance impact of ensemble learning techniques in MIC. .. 215

Figure 7.1: Illustration of various selected metrics from the library of MISeval to evaluate model performance on the use case COVID-19 infected region segmentation. 222

Figure 7.2: Metric behavior demonstration in the context of different-sized ROIs compared to the total image. 225

Figure 7.3: Demonstration of metric behavior for a trained segmentation model in the context of different medical imaging modalities. 226

Figure 7.4: Comparison of the performance metrics considered with the presented MISm in terms of the ratio of false positives to actual negatives if the class observed is not present in the image. 232

Figure 7.5: Scoring comparison between MISm and multiple common MIS metrics by application on normal as well as edge cases. 233

Figure 8.1: Popularity estimation for the MISeval framework. 262

Figure 8.2: Chances (green) and challenges (red) of the software integration process for AI in medical imaging. 265

Figure 8.3: Major steps in a clinical application study illustrated with a schematic timeline. 266

List of Tables

Table 3.1: Overview of Hounsfield unit ranges for selected tissue types.....	46
Table 3.2: Overview of preprocessing techniques including their types, uses, and application references.	52
Table 3.3: Overview of available medical image analysis frameworks for research.	65
Table 4.1: Overview of available examples for MIScnn.....	93
Table 4.2: Distribution of unit tests for the MIScnn framework.	95
Table 4.3: Performance results of the 3-fold cross-validation for tumor and kidney segmentation.	107
Table 4.4: Achieved macro-averaged results on the single-rater dataset.....	114
Table 4.5: Achieved macro-averaged results of the multi-rater dataset.	114
Table 4.6: Comparison of nuclei segmentation performance between NuCLS, Mask R-CNN, and MIScnn.	115
Table 4.7: Achieved results of the sensitivity analysis for COVID-19 segmentation.	124
Table 4.8: Performance impact of data augmentation and preprocessing on COVID-19 segmentation.	125
Table 4.9: Related work overview and comparison for COVID-19 segmentation.....	129
Table 5.1: Overview of implemented deep learning architectures in AUCMEDI	140
Table 5.2: Distribution of unit tests for the AUCMEDI framework.	149
Table 5.3: Arguments of the data interface pillar in AUCMEDI.	152
Table 5.4: Arguments of the neural network pillar in AUCMEDI.....	153
Table 5.5: Arguments of the data generator pillar in AUCMEDI.asd.....	154
Table 5.6: Overview of the datasets utilized in the broad application experiments.	159
Table 5.7: Annotation frequency for each class in the RFMiD dataset.	166
Table 5.8: Achieved results of the internal performance evaluation.	171
Table 5.9: Ranking of the top-7 participants for the final evaluation phase of the RIADD challenge.....	173
Table 5.10: Summary of the STOIC dataset characteristics and proposed sampling.	175
Table 5.11: Spearman correlation coefficients between patient features and target variables.	179
Table 5.12: Computed performance results of the COVID-19 Classifier and Severity Detector.....	180
Table 5.13: Top-10 ranking of the STOIC challenge including achieved prediction results (AUC).	181
Table 5.14: Summary of the EKIPRO dataset annotation and proposed sampling.	186
Table 5.15: Achieved results of the proposed image classification for prostate cancer.	189
Table 6.1: Overview of pooling functions for ensemble learning.	197
Table 6.2: Achieved results of all experiments on ensemble learning based MIS.	202
Table 6.3: Overview of utilized datasets with descriptive details and sampling distributions.	204
Table 6.4: Configuration overview of implemented MIC pipelines.....	206
Table 6.5: Achieved results of the Baseline approach grouped by architecture and dataset.	210
Table 6.6: Achieved results of the Augmenting approach grouped by architecture and dataset.	211
Table 6.7: Achieved results of the Stacking approach grouped by architecture and dataset.	212
Table 6.8: Achieved results of the Bagging approach grouped by architecture and dataset.	213
Table 7.1: Overview and comparison of currently implemented metrics in MISeval.	221

List of Code Snippets

Code Snippet 4.1: Installation process of the proposed MIScnn framework.....	96
Code Snippet 4.2: Setup of a medical image segmentation pipeline with MIScnn.	100
Code Snippet 4.3: MIScnn pipeline utilization for training and prediction processes.....	101
Code Snippet 4.4: KiTS19 dataset loading in MIScnn.	104
Code Snippet 4.5: Initialization of the Preprocessor class for kidney tumor segmentation.....	105
Code Snippet 4.6: Initialization of the Neural_Network class for kidney tumor segmentation.....	106
Code Snippet 4.7: Performing a 3-fold cross-validation on the KiTS19 dataset with MIScnn.	106
Code Snippet 5.1: Installation process of the proposed AUCMEDI framework.	151
Code Snippet 5.2: Foundation setup of a medical image classification pipeline with AUCMEDI.	155
Code Snippet 5.3: AUCMEDI pipeline utilization for training and prediction.	155
Code Snippet 5.4: AUCMEDI AutoML utilization through the command line interface.	156
Code Snippet 5.5: AUCMEDI AutoML utilization through the Docker interface.	157
Code Snippet 7.1: Usage example of the MISeval package.	219
Code Snippet 7.2: MISeval core function <i>evaluate()</i> with docstring documentation.	220

Nomenclature

I

1D *One-Dimensional*

2

2D *Two-Dimensional*

3

3D *Three-Dimensional*

A

Acc *Accuracy*

Adam *adaptive moment estimation*

AHD *Average Hausdorff Distance*

AI *Artificial Intelligence*

AK *actinic keratosis*

API *Application Programming Interface*

AUC *Area Under the ROC Curve*

AutoML *Automated Machine Learning*

AVG *Average*

B

Base FM *Base Framework*

BCC *basal cell carcinoma*

BKL *benign keratosis*

C

CD *Continuous Delivery*

CDS *Clinical Decision Support*

CE *cross-entropy*

CNN *Convolutional Neural Network*

CorrFABR *Correlated Feature Aggregation by*

Region

COVID-19 *coronavirus disease 2019*

CPU *Central Processing Unit*

CT *Computed Tomography*

D

DataAug *Data Augmentation*

DF *dermatofibroma*

DICOM *Digital Imaging and Communications in*

Medicine

DKFZ *German Cancer Research Center*

DR *diabetic retinopathy*

DRD *diabetic retinopathy detection*

DSC *Dice Similarity Coefficient*

F

FN *False Negative*

FP *False Positive*

FPR *False Positive Rate*

G

GAN *Generative Adversarial Network*

GGO *ground-glass opacity*

GI *gastrointestinal*

GIF *Graphics Interchange Format*

GPL-3.0 License *GNU General Public License*

Version 3

GPU *Graphics Processing Unit*

H

HD *Hausdorff Distance*

HU *Hounsfield Unit*

I

I/O *Input and Output, Input and Output*

ICU *Intensive Care Unit*

ILR *Infection Lung Ratio*

IoU *Intersection-over-Union*

ISIC. *The International Skin Imaging Collaboration*

J

JPG *Joint Photographic Experts Group*

K

Kap *Cohen's Kappa*

L

LRP *Layer-wise Relevance Propagation*

M

MEL *melanoma*

MHA *MetaImage*

MIA *Medical Image Analysis*

MIC *Medical Image Classification*

MIR *Medical Image Registration*

MIS *Medical Image Segmentation*

MIScnn *Medical Image Segmentation with*

Convolutional Neural Networks

MISeval *metric library for Medical Image*

Segmentation EVALuation

MISm *Medical Image Segmentation metric*
 ML *Machine Learning*
 MONAI..... *Medical Open Network for AI*
 MRI..... *Magnetic Resonance Imaging*

N

NIFTI.....*Neuroimaging Informatics Technology Initiative*
 nMCC *normalized Matthews Correlation Coefficient*
 no-new-U-Net.....*nnU-Net*
 NPY *NumPy file format*
 NV..... *melanocytic nevus*

O

OCT *Optical Coherence Tomography*
 OD..... *Object Detection*

P

PACS*Picture Archiving and Communication System*
 PET *Positron Emission Tomography*
 pixel *Picture Element*
 PNG *Portable Network Graphics*
 POI..... *Percentage of Infection*
 PreProc.....*Preprocessing*
 PyPI.....*Python Package Index*

Q

QIA *Quantitative Image Analysis*

R

RCC *renal cell carcinoma*
 ReLU.....*Rectified Linear Unit*
 RGB *Red-Green-Blue*
 ROC *Receiver Operating Characteristic*

ROI *Region of Interest*
 RT-PCR ..*Reverse Transcription Polymerase Chain Reaction*

S

SARS-CoV-2 *severe respiratory syndrome coronavirus 2*
 SCC.....*squamous cell carcinoma*
 Sens..... *Sensitivity*
 SGD *Stochastic gradient descent*
 SIRM *Italian Society of Medical and Interventional Radiology*
 Spec *Specificity*
 SPECT *Single-Photon Emission Computed Tomography*
 sTILs.....*stromal tumor-infiltrating lymphocytes*
 SV *Splenic Volume*

T

Thorax..... *Chest*
 TIFF *Tagged Image File Format*
 TN..... *True Negative*
 TP *True Positive*
 TPR..... *True Positive Rate*
 TSV..... *Tab Separated Vector*

V

VASC..... *vascular lesion*
 VGG *Visual Geometry Group*
 ViT *Vision Transformer*
 VRAM *Video Random Access Memory*

W

WHO..... *World Health Organization*
 wSpec_α *weighted Specificity*

1

Introduction

“An expert is a man who has made all the mistakes which can be made in a very narrow field.”

Niels Bohr, Danish physicist, 1885-1962, [1].

In this chapter, the Author presents the motivation for this dissertation and points out multiple challenges in the field of deep neural network based medical image analysis. To solve these challenges, four objectives are defined and corresponding contributions by the Author are highlighted. Furthermore, this chapter provides an outline of the content in this thesis and an overview of the achieved publications.

1.1 Motivation

In the last decade, artificial intelligence (AI) has become a powerful tool that is involved in our everyday lives [2–5]. The Oxford Dictionary defines ‘artificial intelligence’ as follows [6]:

“The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.”

The most widely utilized concept behind AI algorithms is machine learning (ML) [3, 7]. Hereby, an AI model is trained on sample data, which is also called training data, in order to generate artificial data or predictions. The difference between ML-based AI to traditional algorithms is that the model itself learns the required knowledge from the training data instead of being explicitly implemented as program code in order to conduct the task [8]. A core feature of ML is the generalization that allows the definition of an entity by its characteristics resulting in the capability of handling variance in samples. By analyzing large quantities of complex data, AI demonstrated to be capable of solving tasks that were thought to could be done only by humans [8–10]. To achieve such capabilities, ML algorithms are often based on statistical models enabling the representation of complex settings [8]. Today, the most popular ML algorithm for AIs is the deep neural network [8, 9, 11, 12]. A deep neural network is inspired by the human brain in which linked neurons form a large network capable of solving challenging tasks [8, 10]. Due to deep neural networks being one of the major methods used in this dissertation, the theory of this ML algorithm is introduced further in detail in Chapter 2.2.

“Following the trend towards a human-level general AI, researchers predict that AI will automate many tasks, including translating languages, writing best-selling books and performing surgery — all within the coming decades” (Hosny et al. [8]).

In the last years, research on AI has seen rapid growth with deep neural network models [8, 13, 14]. The progress of the digital era with a large quantity of devices in any household and industry enabled the collection of massive amounts of data [8]. In combination with the advancements in computational hardware, AI models can utilize the increasingly available data for training. This allowed researchers to develop AI models which demonstrate powerful prediction capabilities and achieve similar performance as or even surpass humans [8, 11, 13, 15]. Consequently, AI methods have been used in a wide variety of applications ranging from autonomous driving to natural language processing [8, 16]. One of the most popular fields of AI application is computer vision in which the objective is to teach a computer the ability of visual understanding [8, 13, 17]. The breakthrough which makes this objective possible was enabled by the convolutional architecture for deep neural network models [8, 9, 11, 13, 17]. Due to computer vision in medicine being one of the central topics in this dissertation, the research field is introduced further in detail in Chapter 2.3.

“With the emergence of big data, advanced deep learning algorithms and powerful hardware accelerators, modern computer vision systems have dramatically evolved” (Feng et al. [14]).

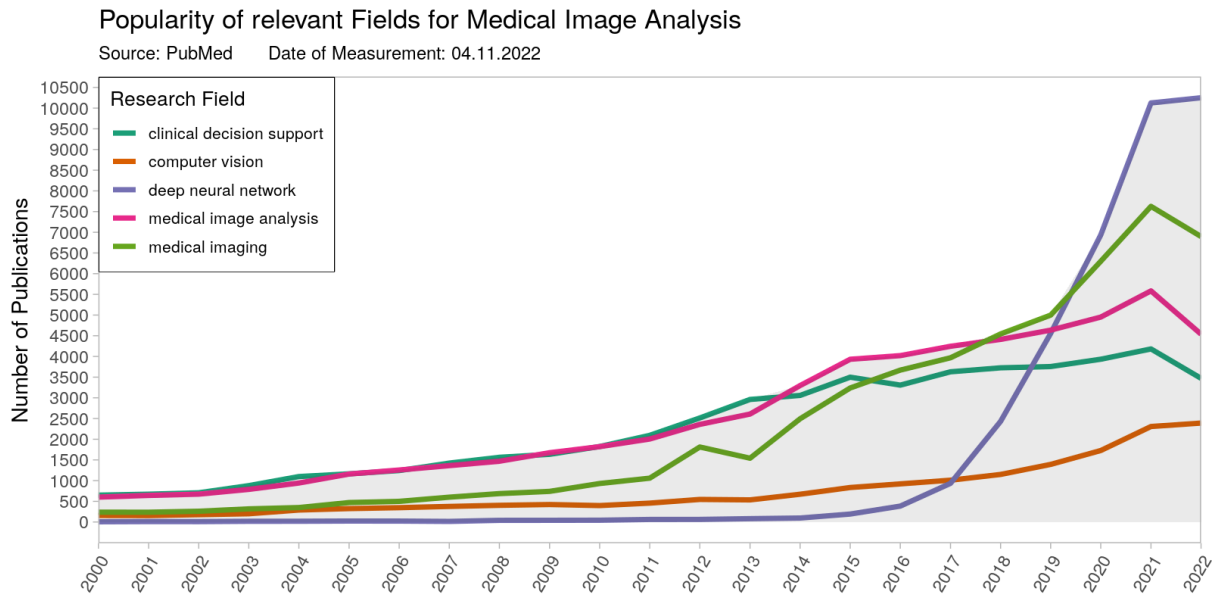


Figure 1.1: Literature search for publications in relevant fields for medical image analysis via PubMed.

In the year 1895, Roentgen discovered the characteristics of X-rays to generate non-invasive visualizations of the insides of a human body and created thereby the first medical imaging modality [9, 18]. Since then, medical imaging became a standard in diagnosis and medical intervention for the visual representation of the functionality of organs and tissues [10, 19–21]. Even though the increasing availability of high-resolution imaging and advanced 3D imaging techniques like magnetic resonance imaging resulted in higher sensitivity and accuracy, these improvements also increased the workload. Additionally, annotations of medical images are highly influenced by clinical experience [22, 23]. Currently, medical image assessment is a manual and tedious process. To shorten the time-consuming inspection and evaluation process, computer-assisted analysis methods for medical images are strived by medical experts [24, 25].

“In recent years the number of publications applying computer vision techniques to static medical imagery has grown from hundreds to thousands” (Esteva et al. [9]).

The field of automated medical image analysis has shown immense growth in recent years [13, 26, 27]. Medical image analysis describes the computer-assisted processing and analysis of medical images in order to generate further insights or support for medical experts [10, 19, 28]. The field can be categorized according to the task into medical image classification, which describes the recognition as well as interpretation of an image, and medical image segmentation, which describes the detection as well as localization of structures within an image. Whereas the subfield of medical image classification aims to automatically label a complete image to a predefined class, the subfield of medical image segmentation aims to automatically label one or multiple regions of interest within an image [10]. In recent studies, medical image analysis models based on deep neural networks proved powerful prediction capabilities and achieved similar results as radiologists regarding performance [8, 9, 13, 15]. Due to medical image analysis being the core topic of this dissertation, the research field is

introduced further in detail in Chapter 2.4. Esteva et al. [9] described the potential of AI in medical imaging as follows:

“As medical AI advances into the clinic, it will simultaneously have the power to do great good for society, and to potentially exacerbate long-standing inequalities and perpetuate errors in medicine.”

The integration of deep neural network based medical image analysis into the clinical routine is currently a highly popular research topic [8, 11, 13, 25, 29]. The idea is to use these powerful models as clinical decision support to improve diagnosis reliability or automate time-consuming processes [13, 30]. Whereas the majority of methods for medical image analysis are currently only applied in research labs, the concept behind clinical decision support is the real-world application in clinical workflows. While clinicians like radiologists visually assess medical images to characterize the condition of a patient based on education and experience resulting in a qualitative assessment, AI models are able to provide a quantitative assessment by automated detection and analysis of medically relevant conditions [8]. The benefits of deep neural network based medical image analysis as a tool for physicians are massive through potential improved diagnostic as well as treatment quality, higher decision reliability by additional information, increased reproducibility, transparency, and time-efficiency by automated structured reporting [8, 9, 11, 13, 25, 29]. The progress of utilizing the capabilities of AI methods has just begun in the field of medical imaging and signals a new era of digital medicine.

The ultimate goal of a medical image analysis pipeline is the integration as clinical decision support in a clinical routine [10]. This is why the central topic of this dissertation is to contribute to deep neural network based medical image analysis in order to advance the progress of utilizing AI models for medical imaging in clinical workflows.

1.2 Challenges

In this chapter, four major challenges are presented that impact the implementation and integration of medical image analysis pipelines based on deep neural networks in clinical workflows. In the following, these challenges are briefly introduced to provide an overview of the objectives of this dissertation. The challenges are further discussed in the corresponding chapters.

➤ Knowledge of state-of-the-art Medical Image Analysis

The setup of a modern medical image analysis pipeline utilizing a state-of-the-art deep neural network to be capable of accurate as well as reliable prediction capabilities is a challenging task [8]. The design and implementation of such pipelines require extensive knowledge in the fields of medical imaging and deep learning [31]. Whereas deep learning experts lack the knowledge in adequate preprocessing of medical images, clinicians lack the computer science background in implementing deep neural networks [31–33]. Therefore, the interdisciplinary field of modern medical image analysis demands highly skilled experts experienced in multiple research fields. To solve this issue and enable clinical applications, enhanced and comprehensive knowledge representations for the implementation of medical image analysis pipelines are needed [31, 33].

➤ Lack of Reproducibility

Applications for medical image analysis are widely prevalent in the literature and demonstrated excellent performance [8, 13, 34–37]. However, clinicians reported immense issues with the reusability of these published models making it impossible for practical usage in clinical research [32, 35, 38, 39]. The causes for this lack of reproducibility could be identified in inferior generalizability through overfitting and intentional optimization on a single dataset [40, 41]. In particular, custom implementations without any standardization are a huge contributor to the deficiency of reproducibility [42, 43]. Another critical cause that was identified in multiple studies is the statistical bias in performance assessment including cherry-picking, incorrect metric implementation, and flawed validation procedures [32, 44–48]. These serious issues in reproducibility present a major threat to the value of medical image analysis pipelines for clinicians and have to be solved to be integrable in health-sensitive workflows.

➤ Lack of Standardization in Implementation

Implementations in the literature commonly are ‘island solutions’ which were developed and optimized for a single dataset [32, 35, 39, 43, 45, 49]. Even though these pipelines are capable of achieving strong performances, the implementations were not designed for reusability in other environments or on other datasets [32, 43, 45]. Such a pipeline often consists of custom-implemented data loaders and optimized preprocessing as well as architectures for a specific dataset. This issue revealed a significant lack of standardization for pipeline implementations in the field which drastically hampers reproducibility, generalization, and reusability [32, 35, 39]. Furthermore, custom implementations also unnecessarily increase the required knowledge

for application due to mandatory modifications for utilizing the pipeline on new data. The current lack of standardization has a critical impact on the research progress in the field through the constant need of reimplementing the ‘wheel’ in any pipeline-building process [43]. As the previously introduced challenges are closely linked to or even a direct consequence of the lacking standardization in the field, this major challenge is the main focus of this dissertation.

➤ **Clinical Application**

Regardless of the achieved strong prediction capabilities of modern medical image analysis models, recent articles like “*Hundreds of AI tools have been built to catch covid. None of them helped*” [50] and studies revealed that clinicians are not able to integrate such pipelines into clinical workflows [32, 38, 39] representing a major challenge in the field. Through the direct impact on diagnosis and treatment decisions, the correctness and reliability of clinical applications are crucial. In addition, the IT infrastructure in hospitals or other medical institutions is critical for modern healthcare [24, 25, 51, 52] which is why novel tools have to be capable of secure integration into such sensitive infrastructures. As a consequence of the required interdisciplinary knowledge resulting in complexity for setup, application as well as maintenance, the lack of reproducibility as well as reusability, and the missing standardization in pipeline implementations, clinicians currently face an insuperable obstacle for utilizing deep neural network based medical image analysis methods as clinical decision support. In order to make reliable clinical application possible, it is essential to solve the presented challenges.

1.3 Research Objectives and Contributions

In the previous chapter, four major challenges in the field of deep neural network based medical image analysis were identified. The Author addressed these challenges by presenting four objectives in this dissertation for which multiple contributions were developed.

1.3.1 Objective I: Development of Frameworks for standardized Pipeline Building

The main objective of this dissertation is the development of frameworks for the standardization of deep neural network based medical image analysis pipelines focusing on medical image segmentation and classification as the most widely applied tasks. Such frameworks would allow rapid and state-of-the-art pipeline building without the need for continuous reimplementations of common components. As all presented challenges are linked to the lacking standardization in the field, solutions for this objective would contribute to an improved knowledge representation of the field, simplify application as well as experimentation by standardization, and strengthen the reproducibility of studies. Consequently, achieving this objective would solve the presented challenges and pave the way for reliable clinical applications.

To enable standardized pipeline building, this dissertation presents the following contributions:

- A comprehensive meta-analysis for defining the state-of-the-art in deep neural network based medical image analysis pipelines
- The development of a framework for state-of-the-art medical image segmentation utilizing deep neural networks
- The development of a framework for state-of-the-art medical image classification utilizing deep neural networks

1.3.2 Objective II: Application Studies demonstrating the Capabilities of the proposed Frameworks

In order to standardize pipeline building and reduce custom implementations, created pipeline applications have to demonstrate excellent capabilities to be competitive in the field. For the reliable application of pipelines created by the proposed frameworks in clinical workflows, it is essential to prove adaptability, state-of-the-art methods, robustness, and high performance.

To demonstrate the capabilities of the proposed frameworks, this dissertation presents the following contributions:

- Kidney tumor segmentation utilizing the proposed segmentation framework
- Standardized medical image classification utilizing the proposed classification framework
- Multi-disease detection in retinal imaging utilizing the proposed classification framework
- COVID-19 infection and severity prediction utilizing the proposed classification framework
- Analyzing the informative value of explainable AI utilizing the proposed classification framework

1.3.3 Objective III: Research Studies utilizing the proposed Frameworks to advance the state-of-the-art of Medical Image Analysis

Besides the advantages of application via frameworks, standardized pipeline building has the potential to substantially support research. Frameworks are capable of rapidly integrating novel methods for experimentation or comparison with existing methods without the efforts of reimplementing a complete state-of-the-art pipeline. Thus, standardization of medical image analysis directly contributes to progressing the field by facilitating further research.

To further advance the state-of-the-art in medical image analysis utilizing the proposed frameworks, this dissertation presents the following contributions:

- Nucleus segmentation based on noise-introduced annotations utilizing the proposed segmentation framework
- COVID-19 lung infection segmentation based on limited training data utilizing the proposed segmentation framework
- Analyzing the performance impact of ensemble learning utilizing the proposed segmentation and classification framework

1.3.4 Objective IV: Improvement of Reproducibility in Medical Image Segmentation

Next to the substantial impact of standardization by frameworks for pipeline building to reproducibility in the field, medical image segmentation revealed significant statistical biases in the performance assessment within studies. For increasing the reproducibility in medical image segmentation as well as the reusability of models in clinical applications, it is essential to strengthen performance assessment for robust evaluation.

To improve the reproducibility of medical image segmentation, this dissertation presents the following contributions:

- The development of a metric framework for medical image segmentation evaluation
- A guideline for evaluation metrics discussing pitfalls as well as metric behavior
- A novel metric for performance assessment in the presence of control samples

1.3.5 Grouping of Contributions into Research Fields

Due to the large number of conducted studies and to provide a better overview, the contributions were grouped according to the utilized framework and into the corresponding research fields. Studies in the context of ensemble learning research and reproducibility improvement were summarized in the ‘further research fields’ group.

- Medical image segmentation
- Medical image classification
- Further research fields: Ensemble learning and reproducibility of performance assessment

The respective categorization of contributions into research fields is outlined in the next chapter. In order to provide an overview of the objectives and contributions that are presented in this dissertation, Figure 1.2 illustrates a summary of the conducted studies categorized to the corresponding objectives.

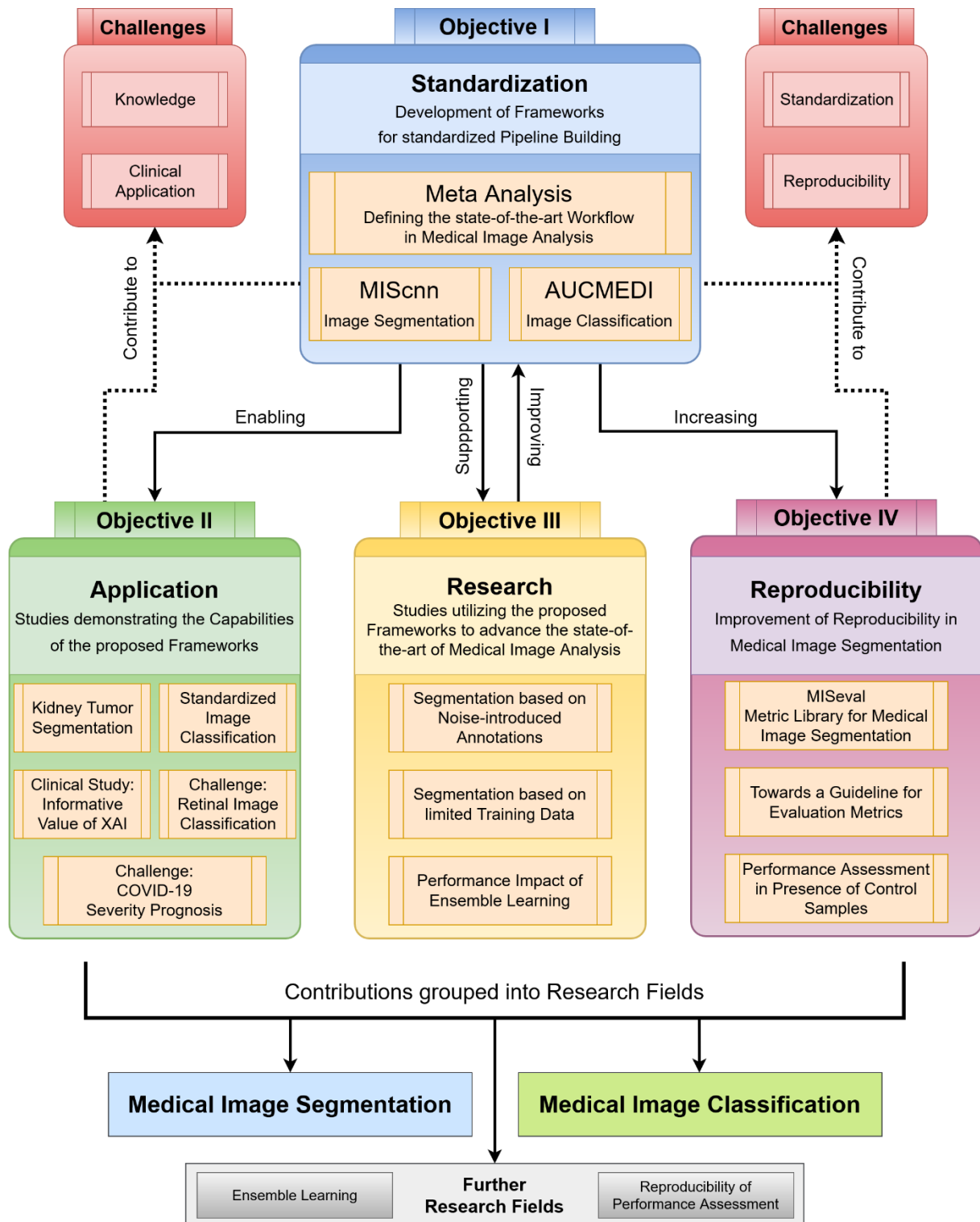


Figure 1.2: Overview of the objectives and contributions presented in this dissertation.

1.4 Thesis Outline

This chapter outlines the structure of the dissertation. The overall structure consists of an introduction, a background chapter as well as a literature review, two core chapters presenting the Author's contributions in medical image segmentation as well as classification, two chapters presenting the contributions in further research fields, the discussion, and the conclusions.

The individual chapters can be summarized as follows:

- **Chapter 1:** The Introduction chapter describes the motivation, challenges, and objectives of the dissertation. Furthermore, the thesis outline and achieved publications are provided.
- **Chapter 2:** The Background chapter summarizes the foundation of this work in a brief literature review.
- **Chapter 3:** The Workflow in Medical Image Analysis chapter provides a meta-analysis to define the state-of-the-art of deep neural network based medical image analysis pipelines.
- **Chapter 4:** The Medical Image Segmentation chapter is one of the core parts of this dissertation presenting a short introduction to the current state of the research field (Chapter 4.1), the proposed framework MIScnn for standardized medical image segmentation (Chapter 4.2), and three studies utilizing the framework: One application study about kidney tumor segmentation (Chapter 4.3) and two research studies about noise-introduced annotations (Chapter 4.4) as well as limited training data (Chapter 4.5).
- **Chapter 5:** The Medical Image Classification chapter is one of the core parts of this dissertation presenting a short introduction to the current state of the research field (Chapter 5.1), the proposed framework AUCMEDI for standardized medical image classification (Chapter 5.2), and four application studies utilizing the framework: Standardized image classification across medical disciplines (Chapter 5.3), the first challenge participation study about multi-disease detection in retinal images (Chapter 5.4), the second challenge participation study about COVID-19 infection and severity prediction (Chapter 5.5), and the clinical study about the informative value analysis of explainable AI (Chapter 5.6).
- **Chapter 6:** The Ensemble Learning chapter belongs to the further research field section and presents a short introduction to the field including the idea, methods, as well as challenges (Chapters 6.1-6.3), and two research studies analyzing the performance impact of ensemble learning in medical image segmentation (Chapter 6.4) as well as classification (Chapter 6.5) utilizing the proposed frameworks.
- **Chapter 7:** The Reproducibility of Performance Assessment chapter belongs to the further research field section and presents three studies about increasing reproducibility in medical image segmentation by providing a metric framework (Chapter 7.1), a guideline for evaluation metrics (Chapter 7.2), and a novel metric for performance assessment in presence of control samples (Chapter 7.3).
- **Chapter 8:** The Discussion chapter discusses individually the advancements of medical image segmentation (Chapter 8.1) and classification (Chapter 8.2) with a focus on general changes in the field, the proposed framework, contributions of the Author to the field, and limitations as well as further challenges. Moreover, the advancements in and contributions

to further research fields are discussed (Chapter 8.3). In addition, the software integration process from the university research into clinical application is addressed (Chapter 8.4). Finally, a future outlook on deep neural network based medical image analysis is provided (Chapter 8.5).

- **Chapter 9:** The Conclusions chapter summarizes the achieved contributions in the context of the defined objectives in this dissertation.

1.5 Publications

From this dissertation, multiple parts and results have been already published in peer-reviewed scientific journals and conferences (SJR listed [53]) or summarized in preprints, which are currently in the submission process. Therefore, this dissertation contains selective text passages from these publications. The publications are listed and summarized in this subchapter, in which the Author's contribution is also highlighted. The contribution by the Author to all presented publications was either as first author, representing the majority contribution to the work, or as team leader indicated through the role of corresponding author for the work. Exceptions to this are the two STOIC studies as well as the EKIPRO study, which were created in equal collaborative team efforts, and the RFMiD review, which was drafted by the RIADD challenge organizers. The chapters containing text passages from these publications are also highlighted in this section. The publications of the Author are not cited again in the associated chapters.

Workflow in Medical Image Analysis

- Title: *A comprehensive Review: The Basics of Semantic Medical Image Segmentation using Deep Convolutional Neural Networks*
 Authors: Dominik Müller, Adrian Pfeiderer, Iñaki Soto-Rey, Frank Kramer
 Role: First Author - Project lead, literature review, meta-analysis, and manuscript drafting
 Journal: Unpublished (currently still in drafting process)
 DOI: /
 Chapter: 3

Medical Image Segmentation

- Title: *MIScnn: a framework for medical image segmentation with convolutional neural networks and deep learning*
 Authors: Dominik Müller, Frank Kramer
 Role: First Author - Project lead, study design, implementation, data analysis, interpretation, and manuscript drafting
 Journal: BMC Medical Imaging. Volume 21, Article 12, 2021.
 DOI: <https://doi.org/10.1186/s12880-020-00543-7>
 Note: Awarded as Editor Highlights for the year 2021 in the journal BMC Medical Imaging
 Chapter: 4.2 and 4.3
- Title: *Nucleus Segmentation and Analysis in Breast Cancer with the MIScnn Framework*
 Authors: Adrian Pfeiderer, Dominik Müller, Frank Kramer
 Role: Corresponding Author - Project lead, interpretation, and manuscript review
 Journal: Submitted as full article to the ISBI 2023 conference.
 Available as preprint in arXiv (Cornell University). 2022.
 DOI: <https://doi.org/10.48550/arXiv.2206.08182>
 Chapter: 4.4
- Title: *Robust chest CT image segmentation of COVID-19 lung infection based on limited data*
 Authors: Dominik Müller, Iñaki Soto-Rey, Frank Kramer
 Role: First Author - Project lead, study design, implementation, data analysis, interpretation, and manuscript drafting
 Journal: Elsevier - Informatics in medicine unlocked. Volume 25, 2021.
 DOI: <https://doi.org/10.1016/j.imu.2021.100681>
 Chapter: 4.5
- Title: *Assessing the Role of Random Forests in Medical Image Segmentation*
 Authors: Dennis Hartmann, Dominik Müller, Iñaki Soto-Rey, Frank Kramer
 Role: Corresponding Author - Project lead, study design, interpretation, and manuscript review
 Journal: German Medical Data Sciences 2021: Digital Medicine: Recognize–Understand–Heal
 Poster at the 66th conference of the German Association for Medical Informatics, Biometry and Epidemiology (GMDS). 2021 in Kiel, Germany.
 DOI: <https://doi.org/10.3205/21gmids015>
 Chapter: Not mentioned

Medical Image Classification

- Title: *AUCMEDI: a Framework for Automated Classification of Medical Images*
 Authors: Dominik Müller, Simone Mayer, Dennis Hartmann, Inaki Soto-Rey, Frank Kramer
 Role: First Author - Project lead, implementation, and presenter
 Journal: German Medical Data Sciences 2022: Future Medicine: More Precise, More Integrative, More Sustainable! Presentation at the 67th conference of the German Association for Medical Informatics, Biometry and Epidemiology (GMDS). 2022 in Kiel, Germany.
 DOI: Conference Talk: <https://doi.org/10.3205/22gmds051>
 Repository: <https://doi.org/10.5281/zenodo.6633540>
 Chapter: 5.2

- Title: *Standardized Medical Image Classification across Medical Disciplines*
 Authors: Simone Mayer, Dominik Müller, Frank Kramer
 Role: Corresponding Author - Project lead, study design, data analysis, interpretation, manuscript drafting, and reviewing
 Journal: Available as preprint in arXiv (Cornell University). 2022.
 DOI: <https://doi.org/10.48550/arXiv.2210.11091>
 Chapter: 5.3

- Title: *Multi-Disease Detection in Retinal Imaging Based on Ensembling Heterogeneous Deep Learning Models*
 Authors: Dominik Müller, Inaki Soto-Rey, Frank Kramer
 Role: First Author - Project lead, study design, implementation, data analysis, interpretation, and manuscript drafting
 Journal: IOS Press - German Medical Data Sciences 2021: Digital Medicine: Recognize–Understand–Heal
 Published as full article in the proceedings of the 66th conference of the German Association for Medical Informatics, Biometry and Epidemiology (GMDS). 2021 in Kiel, Germany.
 DOI: <https://doi.org/10.3233/shti210537>
 Chapter: 5.4

- Title: *RFMiD: Retinal Image Analysis for multi-Disease Detection Challenge*
 Authors: Samiksha Pachade, Prasanna Porwal, Manesh Kokare, Girish Deshmukh, Vivek Sahasrabudhe, Zhengbo Luo, Feng Han, Zitang Sun, Li Qihan, Sei-ichiro Kamata, Edward Ho, Edward Wang, Asaanth Sivajohan, Saerom Youn, Kevin Lane, Jin Chun, Xinliang Wang, Yunchao Gu, Sixu Lu, Young-tack Oh, Hyunjin Park, Hung Yeh, Kai-Wen Cheng, Chia-Yen Lee, Haoyu Wang, Jin Ye, Junjun He, Lixu Gu, Dominik Müller, Inaki Soto-Rey, Frank Kramer, Hidehisa Arai, Yuma Ochi, Takami Okada, Luca Giancardo, Gwenolé Quellec, Fabrice Mériaudeau
 Role: Drafting a subchapter and reviewing
 Journal: Submitted to Elsevier - Medical Image Analysis. 2022.
 DOI: /
 Chapter: Mentioned in 8.2.2 Application Research

- Title: *COVID-19 Severity Prediction with Transfer Learning based SOTA Image Classification Networks, Infection-Lung-Ratio, and Meta-Data*
 Authors: Dominik Müller, Silvan Mertes, Niklas Schröter, Fabio Hellmann, Miriam Elia
 Role: All authors contributed equally to the project - Focus on implementation
 Journal: Submitted to the MICCAI challenge STOIC2021. 2022.
 DOI: /
 Chapter: 5.5

- Title: *Towards Automated COVID-19 Presence and Severity Classification with Ensembles, Transfer Learning and Deep Learning*
 Authors: Dominik Müller, Silvan Mertes, Niklas Schröter, Fabio Hellmann, Miriam Elia
 Role: All authors contributed equally to the project - Focus on implementation and manuscript drafting
 Journal: Submitted to Medical Informatics Europe 2023: “Caring is Sharing - Exploiting Value in Data for Health and Innovation” (EFMI MIE 2023). Göteborg, Sweden 22-25 May 2023. 2022.
 DOI: /
 Chapter: 5.5

- Title: *Klinische Entscheidungshilfen dank erklärbarer Künstlicher Intelligenz am Beispiel des Prostata-Karzinoms („EKIPRO“)*
 Authors: Inaki Soto Rey, Johannes Raffler, Ralf Huss, Lukas Rentschler, Dominik Müller, Philip Meyer, Christoph Wengenmayr, Robin Manz, Samantha Craemer, Jonas Bäcker
 Role: All authors contributed equally to the project - Development lead for AI and XAI
 Journal: Unpublished.
 Accepted as proposal by the ethics committee of the Ludwig Maximilian University of Munich and funding program: “Intramurale Forschungs- und Nachwuchsförderung der Medizinischen Fakultät Augsburg”. 2021.
 DOI: /
 Chapter: 5.6

- Title: *Classification of Viral Pneumonia X-ray Images with the Aucmedi Framework*
 Authors: Pia Schneider, Dominik Müller, Frank Kramer
 Role: Corresponding Author - Project lead, interpretation, and manuscript review
 Journal: Available as preprint in arXiv (Cornell University). 2021.
 DOI: <https://doi.org/10.48550/arXiv.2110.01017>
 Chapter: Not mentioned

Ensemble Learning

- Title: *COVID-19 Image Segmentation Based on Deep Learning and Ensemble Learning*
 Authors: Philip Meyer, Dominik Müller, Iñaki Soto-Rey, Frank Kramer
 Role: Corresponding Author - Project lead, study design, data analysis, interpretation, manuscript drafting, and reviewing
 Journal: IOS Press - Studies in Health Technology and Informatics: Public Health and Informatics
 Published as short communication article in the proceedings of the 31st Medical Informatics in Europe Conference (MIE) organized by the European Federation for Medical Informatics (EFMI). 2021 in Athens, Greece, but virtual hosted due to the corona pandemic.
 DOI: <https://doi.org/10.3233/shti210223>
 Chapter: 6.4

- Title: *An Analysis on Ensemble Learning optimized Medical Image Classification with Deep Convolutional Neural Networks*
 Authors: Dominik Müller, Iñaki Soto-Rey, Frank Kramer
 Role: First Author - Project lead, study design, implementation, data analysis, interpretation, and manuscript drafting
 Journal: IEEE Access. Volume 10, 2022.
 DOI: <https://doi.org/10.1109/ACCESS.2022.3182399>
 Chapter: 6.1, 6.2, and 6.5

Reproducibility of Performance Assessment

- Title: *MISeval: A Metric Library for Medical Image Segmentation Evaluation*
 Authors: Dominik Müller, Dennis Hartmann, Philip Meyer, Florian Auer, Iñaki Soto-Rey, Frank Kramer
 Role: First Author - Project lead, study design, implementation, data analysis, interpretation, and manuscript drafting
 Journal: IOS Press - Studies in Health Technology and Informatics: Challenges of Trustable AI and Added-Value on Health
 Published as full article in the proceedings of the 32nd Medical Informatics in Europe Conference (MIE) organized by the European Federation for Medical Informatics (EFMI). 2022 in Nice, France.
 DOI: <https://doi.org/10.3233/shti220391>
 Chapter: 7.1

- Title: *Towards a guideline for evaluation metrics in medical image segmentation*
 Authors: Dominik Müller, Iñaki Soto-Rey, Frank Kramer
 Role: First Author - Project lead, study design, implementation, data analysis, interpretation, and manuscript drafting
 Journal: BMC Research Notes. Volume 15, Article 210, 2022.
 DOI: <https://doi.org/10.1186/s13104-022-06096-y>
 Chapter: 3.5 and 7.2

- Title: *MISm: A medical image segmentation metric for evaluation of weak labeled data*
 Authors: Dennis Hartmann, Verena Schmid, Philip Meyer, Iñaki Soto-Rey, Dominik Müller, Frank Kramer
 Role: Corresponding Author - Project lead, study design, data analysis, interpretation, manuscript drafting, and reviewing
 Journal: German Medical Data Sciences 2022: Future Medicine: More Precise, More Integrative, More Sustainable!
 Poster at the 67th conference of the German Association for Medical Informatics, Biometry and Epidemiology (GMDS). 2022 in Kiel, Germany.
 Submitted as full article to MDPI - Imaging - Special Issue "Current Methods in Medical Image Segmentation".
 Available as preprint in arXiv (Cornell University). 2022.
 DOI: Conference: <https://doi.org/10.3205/22gmds031>
 Preprint: <https://doi.org/10.48550/arXiv.2210.13642>
 Chapter: 7.3

2

Background

In this chapter, the Author provides an overview of the important fundamentals that are relevant for this thesis. The development of frameworks for medical image analysis with deep neural networks is a highly interdisciplinary topic that combines the broad fields of medical imaging and computer vision. This review focuses on methods, concepts, and domains that are essential for understanding and contextualizing the described challenges as well as proposed solutions in this work.

The overview consists of an introduction to medical imaging including imaging modalities as well as digital imaging data, artificial neural networks including the theory, supervised learning, deep learning, as well as convolutional neural networks, computer vision including image classification, segmentation, as well as hardware requirements, and medical image analysis including major research fields as well as clinical decision support.

2.1 Medical Imaging

Medical imaging is defined as procedures and techniques for generating visual representations of the body as well as of biological functions [18, 19, 28]. These visual representations are referred to as medical images. Techniques for medical imaging are utilized by health professionals from a large number of different medical fields for assessing information about the human body [18, 19]. This information provides essential insights for supporting the diagnosis and treatment of diseases [10, 18, 19, 28]. Isaac Bankman from Johns Hopkins University describes medical imaging in his work as follows [28]:

“The discoveries of seminal physical phenomena such as X-rays, ultrasound, radioactivity, and magnetic resonance, and the development of imaging instruments that harness them have provided some of the most effective diagnostic tools in medicine.”

Medical imaging is present in all phases of patient treatment, starting from possible prior documented conditions by the patient to diagnosis as well as treatment by a physician and ending with possible documentation [10, 18, 19]. There are various purposes for medical imaging application: From providing insights into internal structures hidden by skin and bones, visual representation of the functionality of organs or tissues, visualization of disease and progression, and magnifying body features for detailed analysis up to documentation as well as archiving medical conditions [10, 18, 19, 21, 28, 54]. Therefore, medical imaging has four objects for supporting medical decision-making:

- Gain information for the diagnosis, clinical analysis, and medical intervention for disease treatment.
- Enable monitoring and visualization of disease progression.
- Allow collecting samples of abnormal conditions for research and education.
- Enable detailed documentation of visual features as well as prior conditions.

Whereas a large number of medical fields utilize medical imaging procedures, the largest field of processing medical images is radiology [10, 19, 21, 28]. Since the discovery of X-rays in 1895 which marks the start of medical imaging, multiple distinctive techniques for medical image generation have been developed [10, 18]. These techniques can be categorized according to the following features: Representation of internal or external structures, imaging of the full body or a specific organ, invasive or non-invasive procedure, two- or three-dimensional imaging, and the requirement of prior body modification through e.g. contrast or radioactive agents.

2.1.1 Imaging Modalities

The individual types of techniques for medical imaging are called medical imaging modalities. This subchapter provides a brief overview of the major imaging modalities in medicine. Further details on the fundamentals of medical imaging can be found in the work by Paul Suetens [21] or in the work by Roger Bourne [19]. The figures demonstrating the imaging modalities

originate from multiple datasets which were utilized in this thesis and the devices from license-free image databases [55–63].

Radiography

The field of radiography is defined as utilizing X-rays, gamma rays, or similar ionizing radiation for viewing the internal form of an object. The X-ray imaging technique was discovered first by Wilhelm Conrad Röntgen in the year 1895 which also dates the start of medical imaging field [10, 18]. By measuring the absorption rate of sending ionizing radiation through an object, it is possible to visualize as well as compute the density of the object and its structural composition [10, 21, 54]. Whereas the simple X-ray projection of an object returning a two-dimensional representation is a standard procedure in medical diagnostics and commonly referred to as ‘X-ray’, the further advanced technique is computed tomography (CT) in which multiple X-ray projections from different angles are combined into a three-dimensional representation of the object by computer algorithms [10, 21, 54]. However, repeated exposure to ionizing radiation can lead to DNA damage resulting in cancer [19, 21]. Nevertheless, radiography methods like X-ray and CT are widely used for imaging today due to the low costs and high-resolution [10, 21].

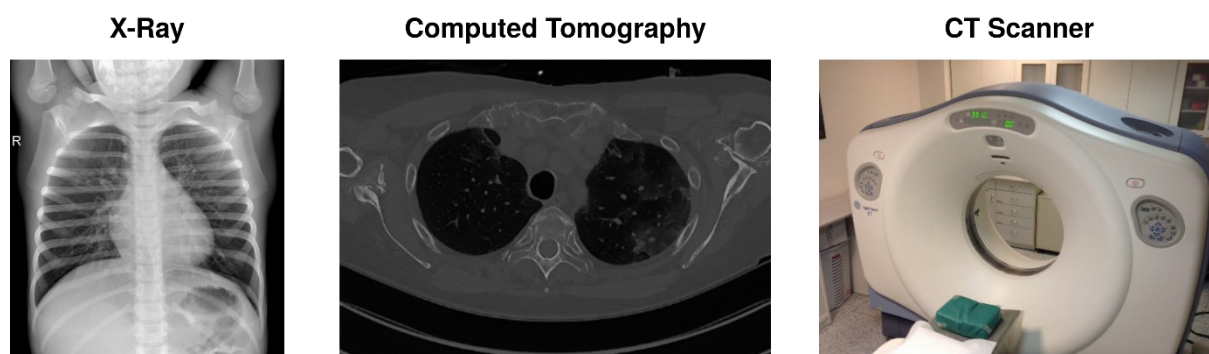


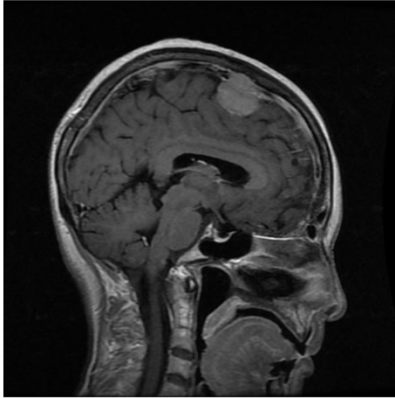
Figure 2.1: Example visualizations of radiography based imaging modalities and corresponding devices.

Magnetic Resonance Imaging

The technique of magnetic resonance imaging (MRI) utilizes powerful magnets to polarize and align hydrogen protons with the magnetic field [10, 21, 54]. Through a radiofrequency current pulse, the protons are stimulated and spin out of their equilibrium [10]. Afterward, the protons realign back to the magnetic field leading to the protons producing detectable signals [10]. These proton signals can be used to distinguish between different chemical environments of their molecules, which can be used to determine the specific human tissue and environment [10, 21, 54]. Additionally, the frequency of the magnetic dipole change, which is also called MRI sequence, allows also highlighting of different chemical environments [10, 21]. The most popular sequences for MRI scans are T1 and T2 [10, 21, 54]. Through computer-assisted processing, the signals are utilized to generate three-dimensional visualizations [10, 21, 54]. Magnetic resonance imaging is a non-invasive and painless procedure that can produce high-

resolution imaging of organs, tissues, and disease states [21]. However, MRI scans require that the patient is capable of laying still for a couple of minutes which is ineffective for quick imaging assessment or for patients suffering from high pain like in an emergency department [21].

Magnetic Resonance Imaging



MRI Scanner

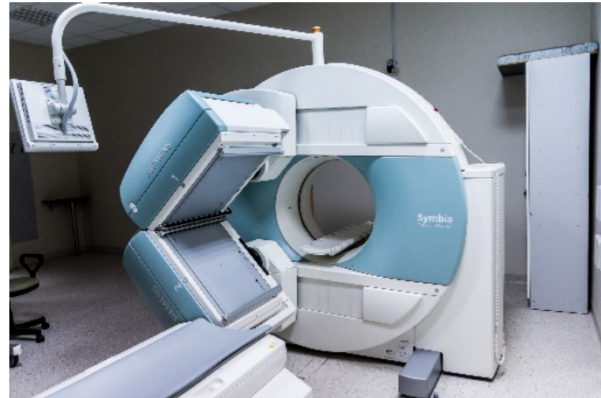


Figure 2.2: Example visualization of magnetic resonance imaging and an MRI scanner.

Nuclear Medicine

The strategy of nuclear medicine is to utilize isotopes and energetic particles emitted from radioactive material to diagnose or obtain insight into pathologies [10, 21, 64]. By injecting radioactive material as a contrast agent in a patient, the detectable isotopes can be used in metabolism and its activity observed [10, 21, 64]. The three major techniques in nuclear medicine are scintigraphy, SPECT, and PET. Scintigraphy describes the general method of generating images by measuring radioactive substances in a body through cameras sensitive to gamma rays [10]. The remaining two methods single-photon emission computed tomography (short SPECT) and positron emission tomography (short PET) are based on scintigraphy [10]. Whereas SPECT measures gamma rays for visualization, the PET technique is based on measuring positrons which are emitted for a short time from an injected radiotracer and allows visualization of functional activity as well as processes [10, 64]. Besides the static imaging approaches to visualize an object in momentum, functional imaging approaches allow object visualization transitioning between different states. Commonly, PET scans are combined with MRI or CT imaging [54].

Ultrasound

The technique in ultrasound imaging is to utilize high-frequency broadband sound waves (in the megahertz range) which are reflected by the tissue [10, 19, 21]. The different tissue reflections can be then used to generate real-time two-dimensional visualization. Because ultrasound imaging is in real-time, without adverse effects, inexpensive, quick as well as simple to perform, it is widely spread and used [21]. Commonly known for pregnancy examinations, ultrasound can also be used for abdominal organs, heart, breast, muscle, tendons, arteries, and vein imaging [21]. However, images based on ultrasound lack noticeably image resolution and penetration strength which is why it is not effective for detailed visualization [10, 19, 21].

Ultrasound Imaging

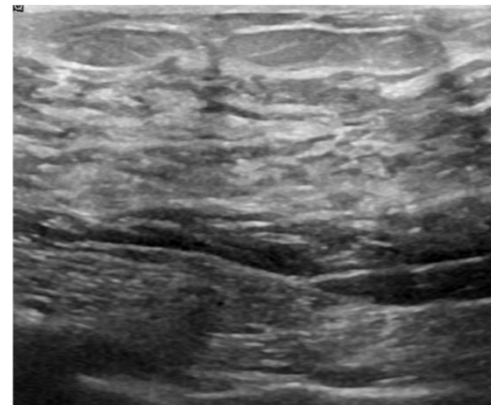


Figure 2.3: Example visualization of ultrasound imaging.

Visible Light Imaging

The utilization of cameras based on the optics of photography is defined as visible light imaging in medicine [10, 19]. The causes of using visible light imaging are diverse ranging from invasive insights into the body and magnification to documentation as well as archiving [10, 19, 65]. Furthermore, the visible light camera can also be represented through a large variety of devices ranging from regular digital cameras to microscopes [10]. Due to visible light imaging having a wide landscape of applications in medical imaging, image quality and resolution also heavily varies depending on the procedure and device age [10, 19]. Common applications are endoscopy for internal as well as invasive visualization of organs, microscopy for magnifying biological samples, optical coherence tomography (OCT) for visualization of internal dermal layers through light waves, ophthalmoscopy for detailed visualization of the retina, and dermatology for documentation as well as detailed visualization through regular photography [10, 19, 65].

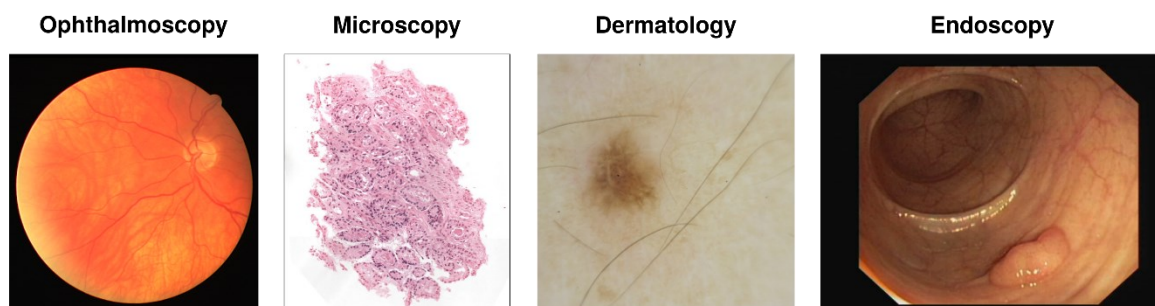


Figure 2.4: Example visualization of four modalities based on visible light imaging.

2.1.2 Digital Imaging Data

In contrast to analogous medical imaging in the last century through visualization on film negatives generated through chemical processes, an image is stored in a digital data array, nowadays [19]. Roger Bourne defines a digital image in his work as follows [19]:

“... a digital image is an encoding of an image amenable to electronic storage, manipulation and transmission. ... No matter how a digital image is stored or handled inside a computer it is displayed as a rectangular array (or matrix) of independent pixels.”

The image is defined through a digital data array which is represented in a two-dimensional (2D) or three-dimensional (3D) matrix [10, 19]. Therefore, a grayscale image matrix can be defined as follows [10]:

$$f : [0, \dots, N_x] \times [0, \dots, N_y] \rightarrow \{0, \dots, 256 - 1\} \quad (2.1)$$

Here, the variables N_x and N_y represent the number of rows and columns, respectively, as well as also determine the image spatial resolution [10, 19]. Each element of the image matrix is assigned a value in the provided value range [10]. The elements and the value range of an image matrix are called pixels (short form of ‘picture element’) and pixel intensity ranges, respectively [10, 19]. The pixels hold and encode the image information through the distributed intensity values. In medical imaging, the pixel intensities are determined by the imaging modality. For example, the pixel intensities of an image from an MRI scan represent the measured MR signal intensities [10]. Whereas the maximum amount of information which is stored in an image is defined by the number of pixels as well as the size of the intensity range, the actual information content is variable [10]. Furthermore, to identify information content it is commonly necessary to analyze not only individual pixels but also complete regions including neighboring pixels [10]. In medical imaging, digital images can also be represented in 3D matrices which are called volumes [10, 19]. A volume matrix is defined equivalently to the image matrix as follows [10]:

$$f : [0, \dots, N_x] \times [0, \dots, N_y] \times [0, \dots, N_z] \rightarrow \mathbb{Z} \quad (2.2)$$

Here, the matrix is extended through the additional N_z axis and the intensity value range is represented through integers [10, 19]. N_z is often referred to as the number of slices in medical imaging [10]. In a volume, an element is called a voxel (short form of ‘volume element’) instead of a pixel [10, 19]. However, the terms image and volume as well as the terms pixel and voxel are used equivalently in this thesis. More details on medical image analysis relevant differences between 2D and 3D matrices can be found in Chapter 3.1.2.

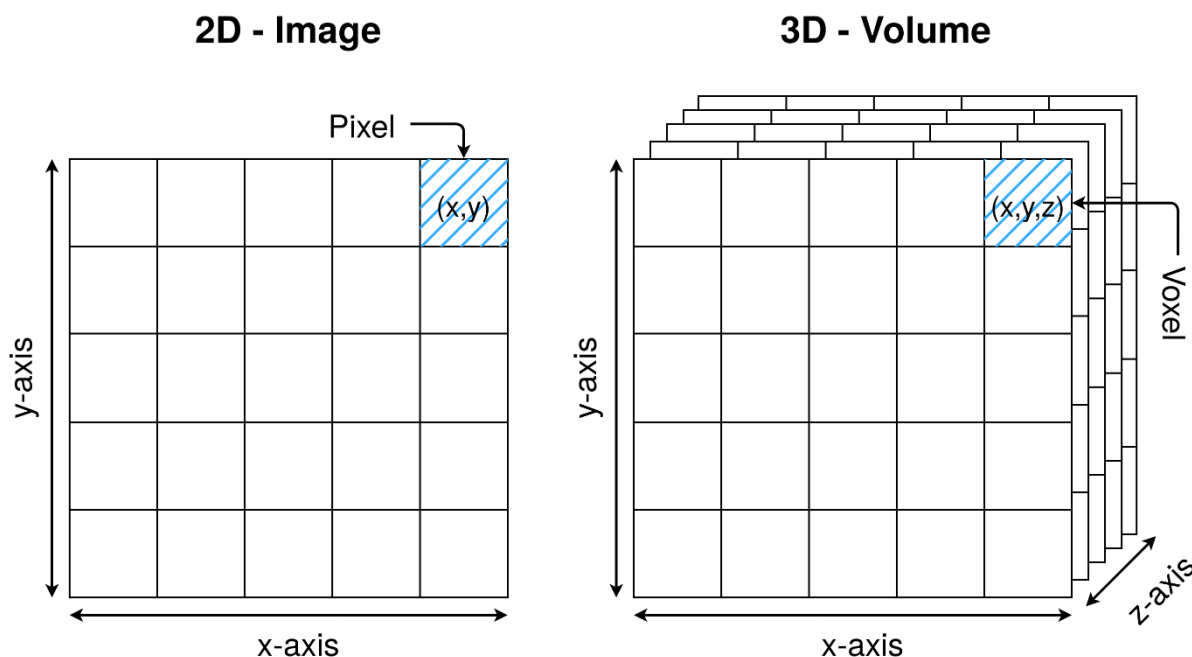


Figure 2.5: Representation of a digital image by an image and volume matrix.

Further important features of digital imaging are the resolution, dots per inch (DPI), and bit depth [10, 19]. The image resolution is defined by the number of pixels and determines the spatial distinctiveness of an object in an image [19]. Whereas a high enough image resolution allows a clear separation between objects as well as a fine visualization of object features, a low image resolution can lead to not distinctively detectable objects with blurry features [19]. The associated metavariable for recommended visualization of the image matrix is defined by the optional DPI which advocates the number of pixels per inch for printing or visualization on a monitor [19]. Next to the image resolution as a measurement for spatial image quality, the bit depth, which defines the size of the intensity value range, is also a measurement for image quality [19]. A high enough bit depth is required to measure the difference between objects with close intensity values which is why a small bit depth can lead to indistinctive and blurred regions in which objects as well as the background have similar intensity values [19].

More about image representation as well as file formats in medical imaging is described in Chapter 3.1.

2.2 Artificial Neural Networks

Neural networks are the most popular artificial intelligence or machine learning algorithms for solving complex problems like natural language processing or computer vision [10, 16, 30]. This chapter describes the basic functionality, structure, and concepts of a neural network model. The aim is to gain a rough understanding of neural networks in order to better assess their importance in medical image analysis.

2.2.1 Theory

Artificial neural networks were originally inspired by biologically self-learning systems such as the strongly networked nerve cells in the human brain [10, 13, 66]. A neuron forms the basic unit of the network, which produces a single output signal, also called activation, from several incoming signals. An artificial neuron is defined by four properties [10, 13, 16, 30, 67, 68]:

- The weighting W of the individual inputs of the neuron describes the extent of the influence of the respective inputs.
- The transfer function $w^L x$ (also expressed as Σ) bundles and weights the input x of the neuron in layer L and transfers it to the activation function as a network input.
- The activation function σ calculates the activation using the network input and a threshold value.
- The threshold value b (also called bias value) controls the sensitivity of the neuron and thus its action potential.

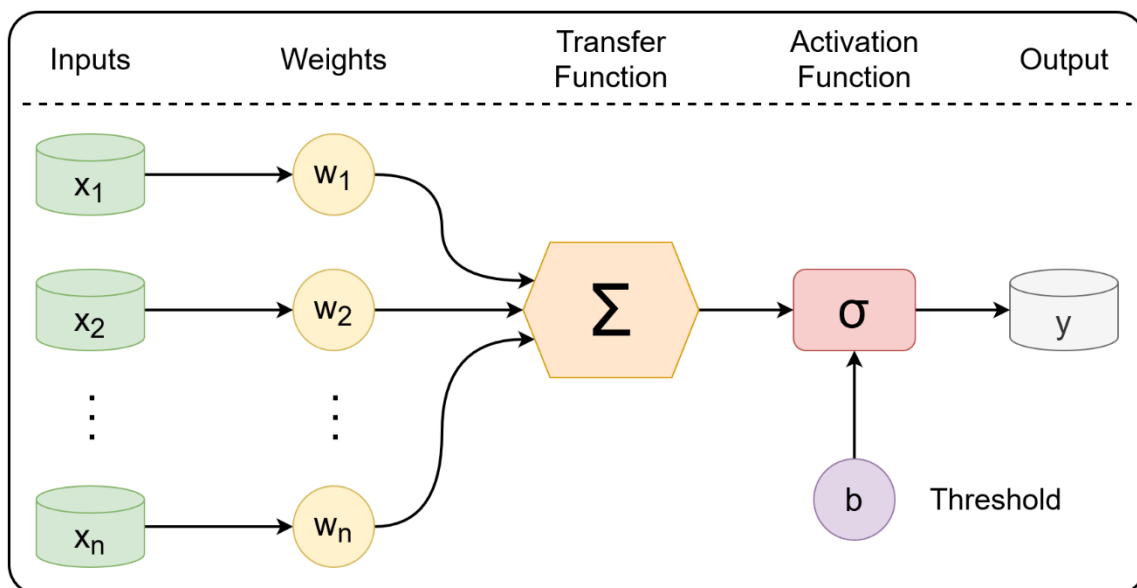


Figure 2.6: Structure of an artificial neuron.

Figure 2.6 shows how the previously discussed internal components of a neuron are arranged. The neuron can accept any number of inputs for which each is multiplied by a weight. The transfer function then creates a network input from the individual results. The activation function, which differs depending on the architecture, now calculates the activation (output). The activation of a neuron can now be summarized as follows:

$$y = \sigma(w^L x + b) \quad (2.3)$$

A neuron can be sensitized to certain inputs by adjusting the weights or the threshold value. If many neurons are linked to form a network, a behavior pattern for complex problems can be achieved [10, 16, 67, 68]. In the following subchapters, it is discussed how neural networks are able to learn and more complex network topologies are introduced like deep neural networks as well as convolutional neural networks.

2.2.2 Supervised Learning

The concept of learning is defined as a process of obtaining a routine for a neural network that solves a problem [68, 69]. Patterns and features are worked out in examples and later applied to unknown data. The aim is to use a learning algorithm to find a mapping that changes the network in such a way that the input is followed by the correct output. These changes in the network can theoretically affect the entire structure, such as adding, deleting or changing connections and neurons [10, 16, 68, 69]. In practice, however, mostly only the weighting of the neuron inputs and slight modification of the activation and output functions are changed during learning [69, 70]. Supervised learning utilizes annotated training data with the associated desired outputs of the network. These training data are passed into the neural network model to be learned after which the resulting output from the model is compared with the desired output. This is done utilizing an error function, which is called the loss function and calculates the quality of the output [10, 16, 68, 69]. An example of an error function is the quadratic error E :

$$E = \sum_{k=1}^n (t_i - y_i)^2 \quad (2.4)$$

In this formula, y is defined as the predicted output, t the actual output, n the number of samples in the dataset, and E the corresponding measure of the error [68]. Based on the error estimation, necessary changes that have to be made to the network can be derived in order to generate the desired output [10, 16, 68, 69]. Commonly, the method of backpropagation is used [10, 16, 68, 69]. This is done by propagating backward through the network and adjusting the neuron weights according to the size of the influence on the error [10, 68]. The algorithm tries to optimize the calculated error of the neuron to a global minimum [10, 68]. However, as common in gradient based techniques, the algorithm may approximate a local minimum [10, 68, 69]. The update of a neuron weight can be represented in the following formula [68]:

$$w_{new} = w_{old} \left(-\eta \frac{\partial E}{\partial w_{old}} \right) \quad (2.5)$$

Here, w is the respective weight, η a constant learning rate that determines the impact strength of the learning process on the network, and E corresponds to the current error [68]. Through repeated training, the model gradually gets closer to the desired results [10, 16, 68, 69].

2.2.3 Deep Learning

A neural network usually consists of many neurons that are connected to one another and the success of training is determined by how the neurons are linked to each other [69]. The layer concept, which arranges the individual neurons in layers, is typical for neural network architecture representation, nowadays [30, 67, 69, 71, 72]. In addition to the input and output layers, there are hidden layers in which the outputs of neurons are often not directly visible or interpretable [10, 69]. Neural network architectures with multiple hidden layers are considered a deep neural network, which is also called a ‘deep learning model’ [30, 67, 69, 71, 72].

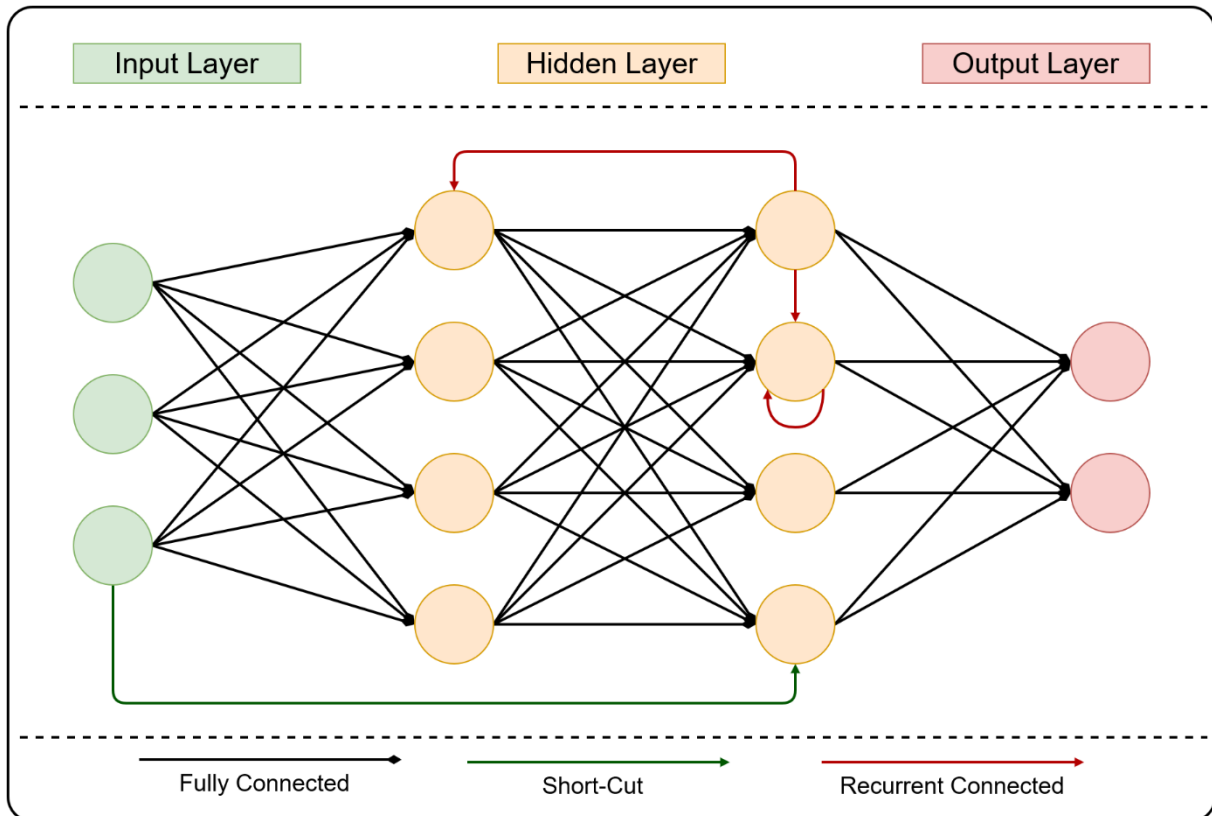


Figure 2.7: Overview of different neural network topologies.

A fundamental deep neural network is the multi-layer perceptron. As input, the multi-layer perceptron receives two sets of parameters $\theta = \{W, B\}$, in which B is a set of threshold values and W is a set of weights of the individual neurons [10, 13, 69]. Layers in between the input and output are referred to as ‘hidden’ layers [10, 13, 30, 67, 69]. The network can be defined as follows [13]:

$$f(x; \theta) = \sigma(w^L \sigma(w^{L-1} \dots \sigma(w^0 x + b^0) + b^{L-1}) + b^L) \quad (2.6)$$

In a feed-forward architecture, each output of a layer is only connected to the inputs of the next layer [30, 69, 70]. An extension of this would be the fully connected topology, which links all neurons of one layer with all inputs of the next layer [30, 69, 70]. Short cuts are connections skipping at least one layer in front of them [30, 69, 70]. The feedback layers, which are also

called recurrent connections, allow neurons to connect to previous layers as well as to themselves [30, 69, 70]. In Figure 2.7, the individual topology types are highlighted as a simple multi-layer perceptron in which a feed-forward concept was assumed for simplification.

2.2.4 Convolutional Neural Networks

Convolutional networks are a special type of deep neural network. The difference to the multi-layer perceptron is the arrangement of the individual neurons in addition to the divided weights as well as local connectivity [30, 69, 70]. These are not aligned one-dimensionally but usually in two to three dimensions within a layer, which is thus referred to as a convolutional layer [30, 69, 70]. This structure is ideal for problems of image processing as an image is often represented by a 3D shape consisting of pixels with one or multiple channels (height, width, and channel)

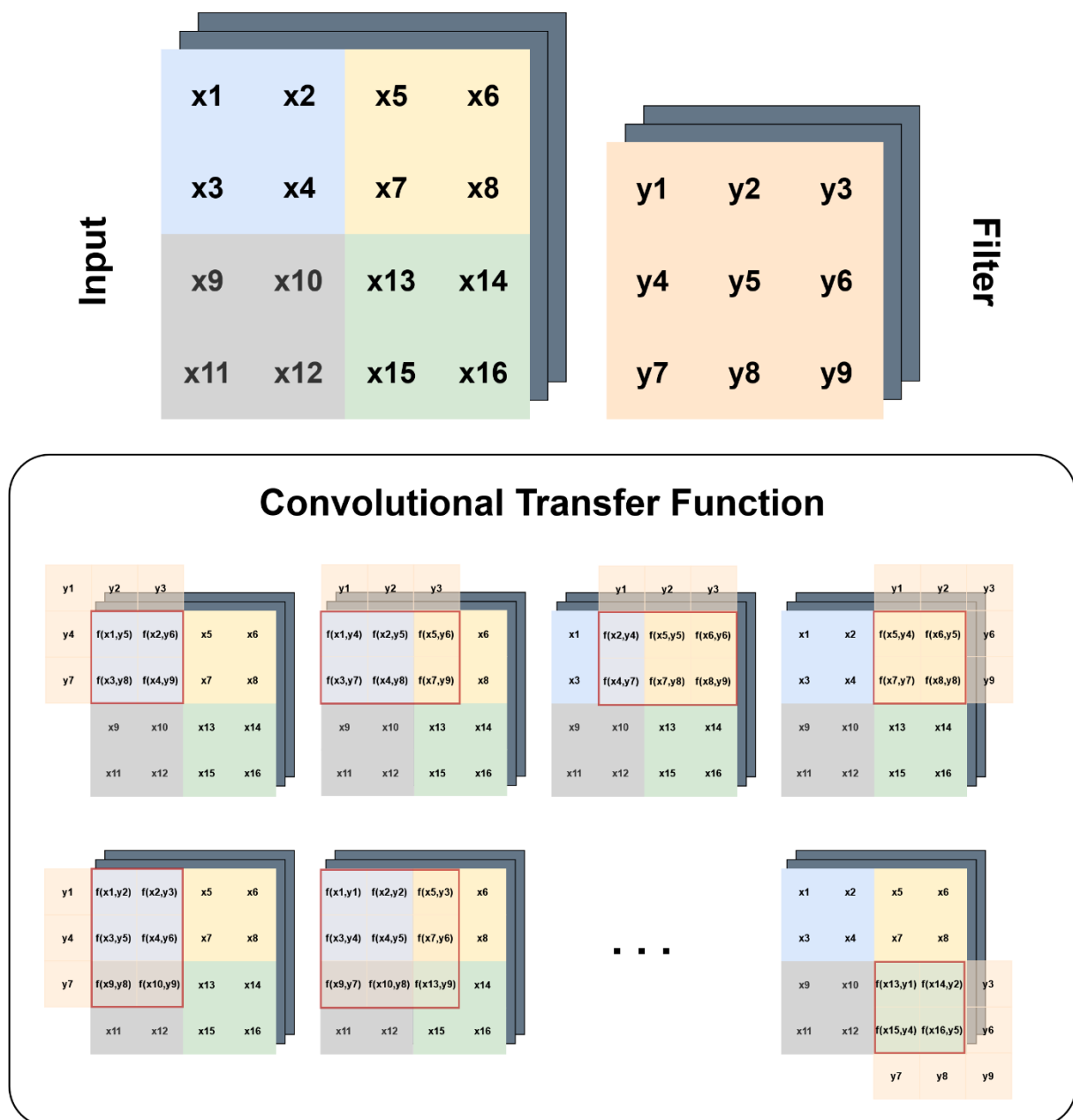


Figure 2.8: Illustration of the convolutional transfer function.

[30, 67, 69, 70]. In a convolutional layer, a filter, which is a stack of multiple kernels, is placed over the input matrix in a sliding window strategy [30, 69, 70]. The filter is a two or three-dimensional matrix (also called a convolution matrix), which dimension is smaller than the input and contains the weights of the neuron [30, 69, 70]. The network input is generated by commonly multiplication through the filter and the area of the input matrix that is ‘covered’ by it [69, 70]. Through this technique, neurons that are located next to each other react to a similar image section. Finally, the activation function is used to calculate the activation output based on the network input as previously defined. In convolutional networks, the rectified linear unit (ReLU) is usually utilized as activation function [30, 67, 69, 70]:

$$f(x) = \max(0, x) \quad (2.7)$$

The ReLU function passes only positive values and maps all negative values to zero [69, 73]. In contrast to the other commonly utilized activation functions like sigmoid [69, 73], the utilization of the ReLU function reduces the processing effort and enables more complex architectures [69, 73]. The output of the ReLU section also forms the activation of the neuron, which is usually also called a feature map [69, 70]. Figure 2.8 demonstrates the operations of a convolutional transfer function. For each movement of the filter kernel, the results of each dimension are added up and the sum of the individual results of the dimensions is formed again. The resulting values form the network matrix for the activation function.

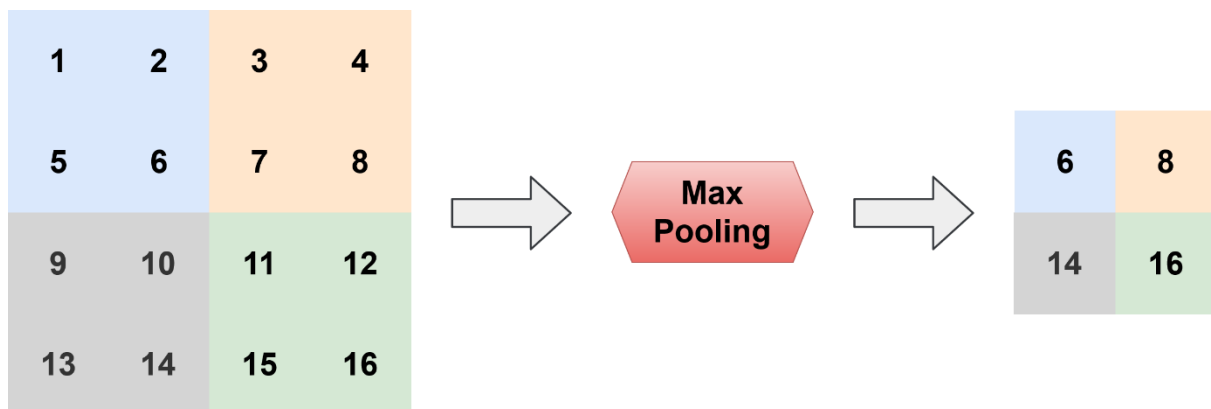


Figure 2.9: Illustration of a maximum pooling operation on a two-dimensional input matrix.

Between the individual convolutional layers, pooling layers are commonly utilized [13, 30, 69, 70]. Pooling layers reduce the information to stabilize the network [13, 30, 69, 70]. Due to convolutional layers being computationally and data-intensive, more complex tasks would not be possible in practice without downsampling through pooling. As a pooling function, max or average operations are commonly utilized for which the maximum pooling is illustrated in Figure 2.9 [13, 30, 69]. Similar to convolutional layers, maximum pooling considers a partial section of the output matrix of the previous layer and only the largest input from the processed area is returned.

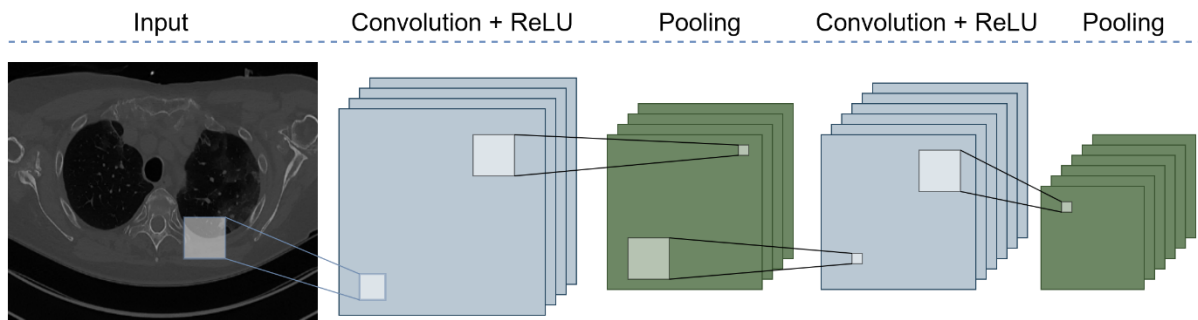


Figure 2.10: Architecture of a convolutional neural network.

Convolutional neural networks, which are also called CNNs, are deep neural networks and their architecture consists of one or multiple layers of convolutional as well as pooling operations [13, 30, 69]. There are various architecture designs for convolutional neural networks which are highly task-specific and utilize further complex strategies [13, 30, 69]. For example, in order to solve a classification problem, mostly multi-layer perceptrons with a fully connected topology are utilized at the end of the network in which the number of neurons for the last perceptron layer corresponds to the number of classes [30, 67]. Nevertheless, convolutional neural networks are the most effective and widely used architectures for computer vision [13, 27, 30, 34, 67, 74, 75].

Throughout this thesis, the terms ‘convolutional neural network’, ‘deep neural network’, ‘deep convolutional neural network’, and ‘neural network’ are referred to as equivalents describing a model based on a modern convolutional neural network architecture.

2.3 Computer Vision

In the last years, computer vision (CV) has seen rapid growth in popularity, application, and research. CV is an interdisciplinary field that deals with how to automatically process, analyze and understand images. The general aim behind computer vision is to build artificial intelligence systems which are able to automate visual tasks with a performance similar to or better than humans. Goodfellow et al. [69] described CV in their work as follows:

“Computer vision is a very broad field encompassing a wide variety of ways of processing images, and an amazing diversity of applications.”

Use cases of CV tasks are widespread starting from robot navigation to clinical decision support for medical imaging [17]. Also, the application tasks of computer vision range from reproducing human visual abilities like object recognition to new forms of visual processing like sound wave diagram analysis [69]. Since the breakthrough of convolutional neural network architectures, neural networks became one of the most accurate and popular machine learning algorithms for automated image analysis [13, 26, 27]. Prevalent subfields of CV are classification, registration, object detection, and segmentation. In this thesis, the focus is set on image classification and segmentation which is why these two subfields are further introduced in the following subchapters. Figure 2.11 illustrates the two different subfields of computer vision on dermatoscopy images by the International Skin Imaging Collaboration (ISIC) [58].

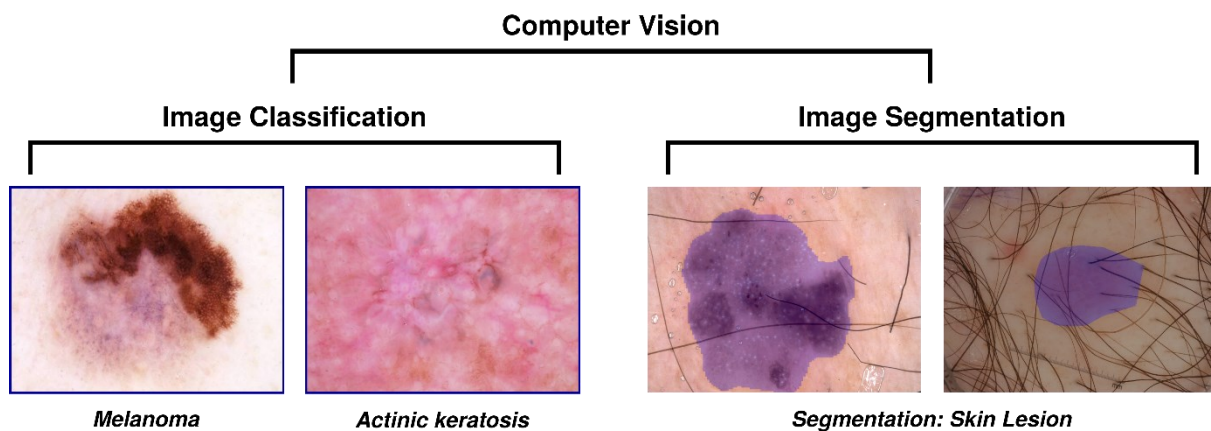


Figure 2.11: Illustration of image classification and segmentation in computer vision.

2.3.1 Image Classification

In the subfield of image classification, complete images are classified into predefined categories or classes according to their visual content. The key characteristic of image classification is that the assertions are made on entire images instead of only parts or pixels within an image. This is why the field focuses on general image understanding and interpretation tasks [17]. Therefore, image classification methods present a solution for tasks about detecting the content of an image. Image classification is a major and one of the most groundbreaking subfields in deep learning based CV with a large number of applications and an active research community [17, 76, 77]. Modern image classification models consist of an architecture combined with a

‘classification head’ at the end of the used architecture [78–80]. Whereas the architecture is often referred to as ‘backbone’ of the model and is responsible for the feature extraction from an image, the classification head is responsible for the classification task based on the identified features provided by the backbone [78]. More information on the classification head can be found in the last subchapter. Popular image classification architectures can also be utilized as backbone for other CV tasks like segmentation. The classification task can be categorized into three types. In a typical scenario, an image is annotated with a single class resulting in a binary or multi-class classification task depending on the number of predefined classes. A special task is multi-label classification in which a single image is annotated with multiple classes.

Scope of Architectures

Since the publication of AlexNet in 2012 [81], which revealed surprisingly accurate results with a convolutional neural network, the research community has been focused on developing novel deep learning architectures based on CNNs [82]. The subfield of image classification in computer vision yielded a large variety of deep convolutional neural networks without a clear single best architecture. Instead, multiple strategies represented through architectures were simultaneously developed and further improved. Even today, the community is not centralized on a single architecture but rather each researcher selects individually their favorite from the current list of top-performing architectures based on the ImageNet dataset [76, 83]. The following paragraph gives a brief overview of architectures that were developed in the last 10 years.

Simonyan et al. [84] further advanced the AlexNet architecture by increasing its depth as well as improving filter configurations [82, 85]. The resulting architecture was proposed by the authors in 2014 as visual geometry group (VGG) [84]. The GoogLeNet or also called InceptionV1 was developed by Szegedy et al. [86] in 2014 and introduced the inception module (‘network in network’). The inception module allowed an improved acquisition of the image content as well as also continued increasing the depth of the model [82]. Since then, the inception-based architecture has been continuously updated and further versions released like InceptionV3 [87]. In the process of continuously increasing the depth of architectures, researchers observed significant vanishing of gradients by normalized initializations resulting in decreasing performance. To solve this issue of degradation, He et al. [80] proposed deep residual learning in 2016 in which residual learning blocks consisting of shortcut connections for identity mapping allow model depths of more than 1,000 layers [82]. The authors named their novel architecture utilizing residual learning blocks ResNet [80]. For the ResNet architecture, multiple updated versions were proposed like the ResNetV2 also by He et al. [82] or the ResNeXt by Xie et al. [88] which introduces cardinality into the ResNet (the size of the set of transformations). After the success of ResNet, the Inception architecture by Szegedy et al. also included residual blocks resulting in the InceptionResNet [89] in 2016. Even so, strong improvements to avoid degradation had been made, large as well as depth architectures still suffered performance due to it. Regarding that, Huang et al. [90] proposed the DenseNet in 2017 which introduced dense blocks allowing the reusing of features of previous layers and

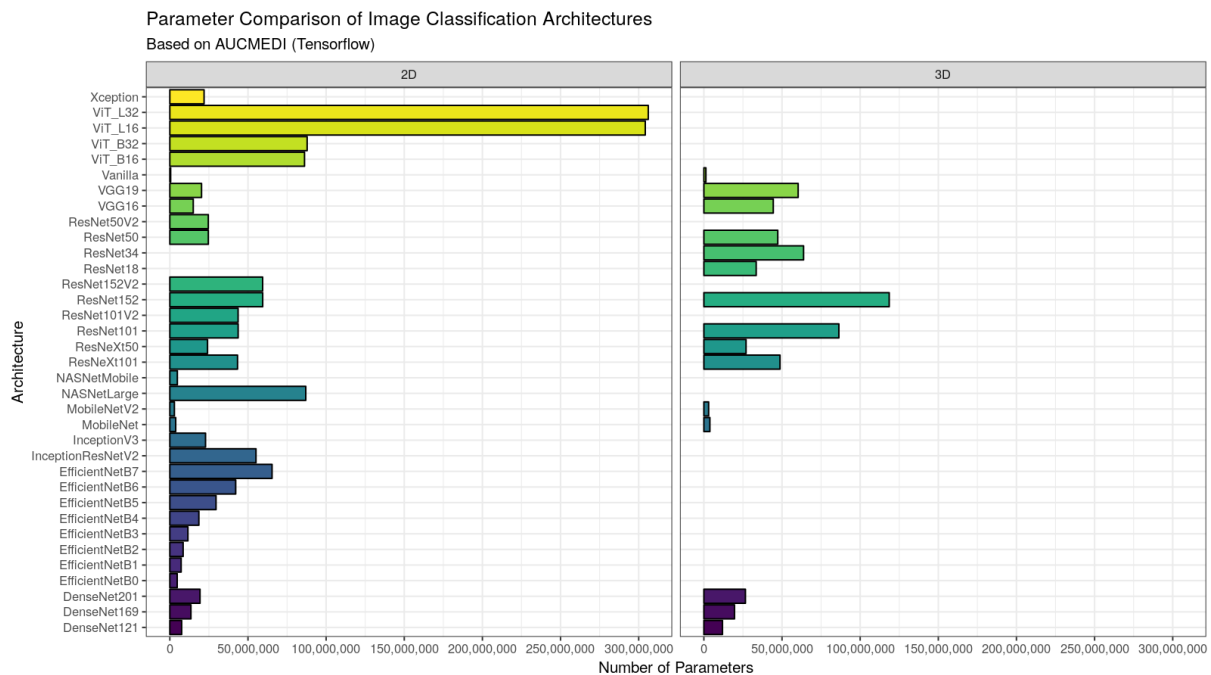


Figure 2.12: Image classification architecture overview (selection) and comparison by number of parameters.

improving feature propagation resulting in a significant reduction of gradient vanishing [82, 85]. This allowed the DenseNet not only to outperform ResNet-based architectures but also needed notably less parameters [82, 85]. In contrast to achieving the highest possible performance with ideal computing resources, Howard et al. [91] developed an architecture for mobile and embedded computer vision applications in 2017. The MobileNet [91] and its successors [92, 93] utilize depth-wise separable convolutions to build the light weighted architectures [91]. In 2019, Tan et al. [79] further studied increasing the depth, width, and resolution of CNNs with a fixed resource budget. The authors proposed the so-called compound coefficient which allowed scaling architectures resulting in top performance with less number of parameters than other architectures [79]. Tan et al. [79] released their series of architectures as EfficientNet. Until then, architectures based on CNNs were unchallenged in computer vision. However, in late 2020, Dosovitskiy et al. [94] published the vision transformer (ViT) architecture which uses pure transformers applied directly on image patches. This approach revealed unsurpassed performance results and a new era of architectures for CV next to CNN-based architectures [94]. Still, the current best architecture design is highly contested as Liu et al. [95] demonstrated with the ConvNeXt architecture in 2022 that CNN-based architectures are still able to surpass ViT. Further architectures which are presented in Figure 2.12 but not introduced are NASNet by Zoph et al. [96], Xception by Chollet et al. [97], and Vanilla which is a minimalistic architecture by the AUCMEDI framework [98] for testing.

Imaging data is commonly represented in 2D which is why the majority of architectures in CV focus on 2D input data. However, architectures specifically designed for 3D imaging data like in medical imaging are limited [99]. Thus, variants of popular 2D deep convolutional neural network architectures were developed to handle the 3D input data. Especially after base frameworks for neural network building introduced corresponding 3D layers for each type of

2D layer (like a 3D convolutional layer with identical usage as the 2D convolutional layer), architectures could be easily transformed to be applicable to 3D data, as well [71, 72]. These architectures demonstrated higher performance than their 2D counterparts due to the increased information available from the additional z-axis [36, 99–101].

Classification Head

As previously introduced, an image classification model consists of a backbone and a classification head. The backbone outputs features encoded in a multi-dimensional matrix [78]. The task of the classification head is to predict the correct classification based on the identified feature matrix from the backbone architecture [78]. In conclusion, the classification head is a neural network part that takes a feature matrix and outputs class probabilities.

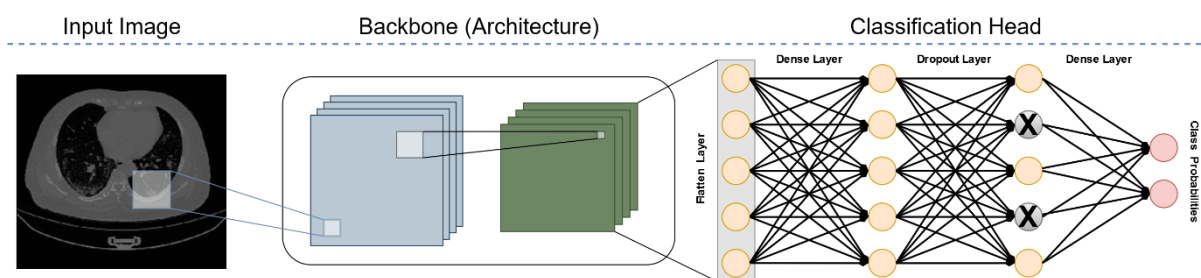


Figure 2.13: Setup of an image classification model including backbone and classification head structure.

The designs of classification heads in computer vision are diverse but commonly defined as a simple fully connected network. The required properties of a classification head are a dimension reduction operation to reduce the multi-dimensional feature matrix into a single dimension, and a final fully connected hidden layer (dense layer) consisting of the same number of neurons as classes combined with a ‘softmax’ or ‘sigmoid’ activation function [78]. As dimension reduction operation, global average pooling or flattening layers are utilized. These two operations represent the minimal as well as the most widely used classification head for image classification. More advanced designs add dense and dropout layers between the dimension reduction and final classification layer. Through the classification head, it is also possible to integrate metadata apart from the image into the classification model. In the literature, the classification head is often not described in detail but is still an important part of an image classification model. An illustration of a classification model setup is shown in Figure 2.13 in which the input CT image originates from Ma et al. [102].

2.3.2 Image Segmentation

Another popular subfield of image processing within the computer vision field is image segmentation. Image segmentation is the process of automated partitioning of an image into multiple segments (clusters of pixels) based on shared features and characteristics [10]. Thereby, each pixel in an image is classified to a label by which the same classified pixels assemble into a segment. In total, the aim of image segmentation is to simplify an image into a more expressive or convenient representation. There are two types of image segmentation:

Instance segmentation and semantic segmentation [17, 103]. Instance segmentation is defined similarly to object detection. Whereas object detection only outputs an approximated bounding box around an object, instance segmentation identifies the exact borders and edges of an object instance [17]. This is possible by performing a decision for each pixel to assign it as part of the object instance or not. Identical to object detection, an image can contain multiple objects of the same class for example the segmentation of eyes in a face image. By comparison, semantic segmentation only assigns each pixel to a class without the need for objectification [17]. Thus, semantic segmentation does not need to differentiate between instances of a class. In contrast to image classification, image segmentation focuses on understanding the entities within an image by training models to differentiate between objects or class types with pixel-precise accuracy.

Scope of Architectures

The range of deep convolutional neural network architectures for image segmentation heavily focuses on fully connected networks [13]. In the beginning, deep learning approaches for segmentation utilized popular classification architectures like AlexNet [81], VGG [84], or ResNet [80] as a base [17]. The idea was to utilize these architectures as a backbone and change only the classification head, which consists of at least two fully connected layers at the end of classification architectures. Instead of the classification head, up-sampling layers were introduced to map the convolutional feature maps back to the original input shape of the image [17, 103–105]. This front-end CNN architecture, also called an encoder, combined with a back-end up-sampling approach to the original input shape, also called a decoder, results in the encoder-decoder architecture [17, 103–105]. The general idea of this structure is to identify and extract image features by encoding these into smaller low-resolution feature maps [17, 103,

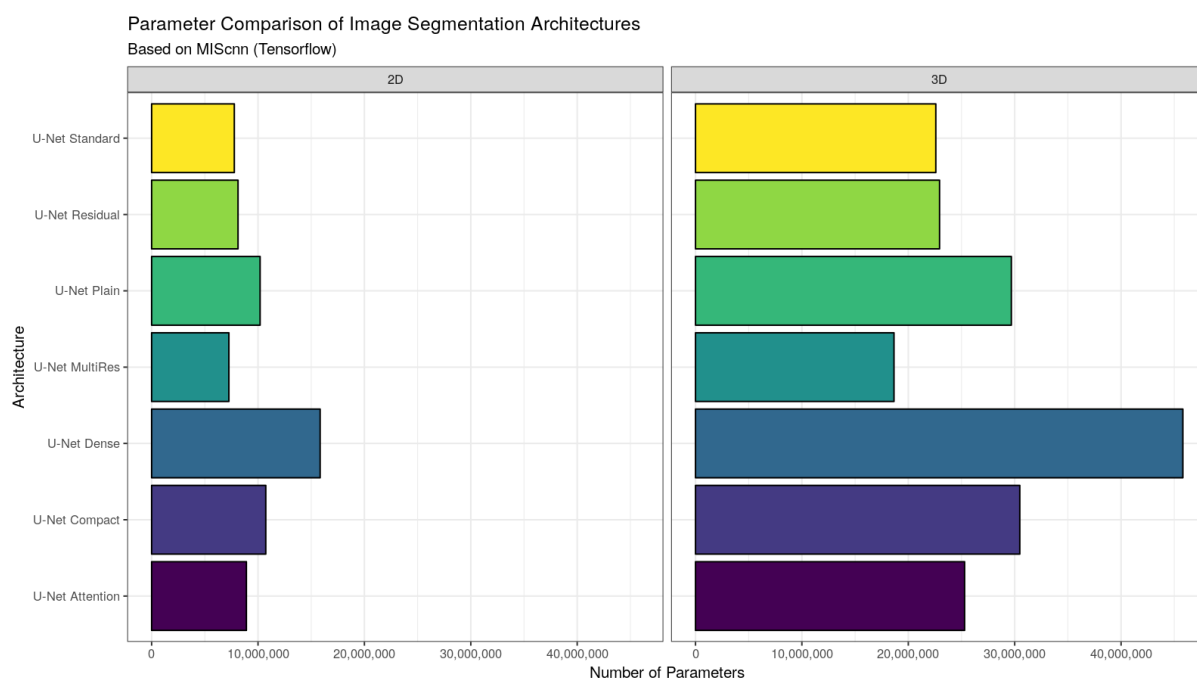


Figure 2.14: Image segmentation architecture overview (selection) and comparison by number of parameters.

104]. Afterward, these low-resolution discriminative feature maps from the encoder are mapped back to high-resolution to perform a classification for each pixel [17, 103, 104]. The encoder-decoder structure is the basis of any state-of-the-art architecture for semantic image segmentation, today [17, 103, 104]. The starting fully connected network architectures were purely based on the common classification architectures combined with deconvolutional layers for up-sampling [17, 103, 104]. However, novel architectures based on the encoder-decoder structure like SegNet (2015) by Badrinarayanan et al. [106] and DeepLab (2016) by Chen et al. [107] were quickly developed. Whereas SegNet introduced a symmetric structure for end-to-end pixel segmentation utilizing improved pooling operations for up-sampling [106], DeepLab focused on not only improving the fuzzy and insensitive up-sampling operation in the decoder, but also on the pixel relationship and spatial consistency [107]. At the same time, Ronneberger et al. [108] published the U-Net architecture (2015), which is still the most widely used architecture in biomedical image segmentation and present in the majority of deep convolutional neural network pipelines [13, 103, 109].

U-Net Architecture

With more than 50,000 citations so far, Ronneberger et al. [108] presented the original U-Net architecture at the 2015 MICCAI conference. The architecture demonstrated excellent performance on various image segmentation datasets and challenges [13, 108, 109]. Furthermore, it showed that it is capable of image segmentation tasks in which only a low number of training images is available and strong class imbalance is common [110]. An illustration of the U-Net structure is shown in Figure 2.15 in which the CT images originate from Ma et al. [102].

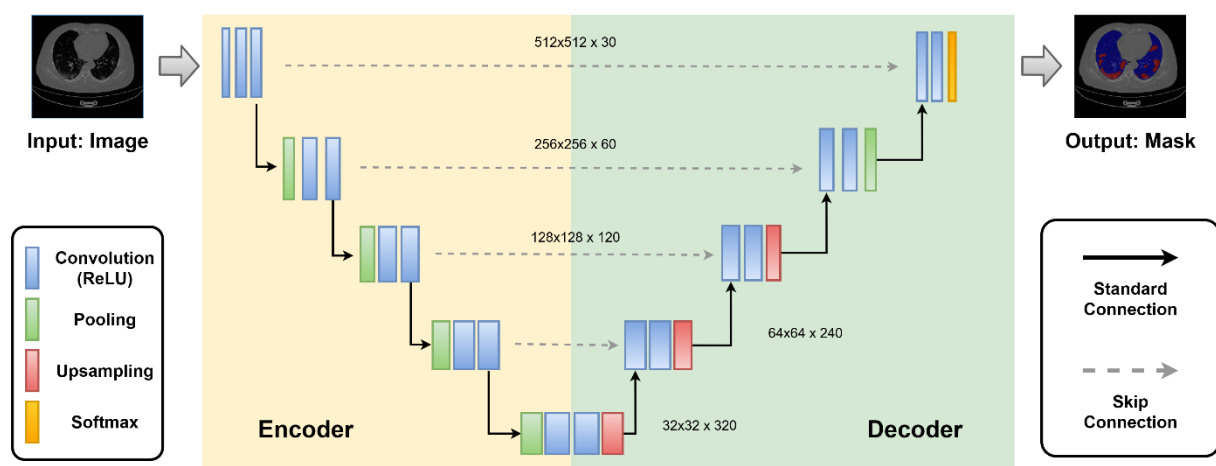


Figure 2.15: Illustration of the U-Net architecture.

The U-Net is built as a symmetric ‘U’ structure with multiple down-sampling and up-sampling levels, representing the encoder-decoder structure. In the original implementation [108], the U-Net is based on four levels, as illustrated in Figure 2.15. Each encoder block contains two convolutional layers with 3×3 ReLU activations and a max pooling layer with a 2×2 kernel size for downsampling. The decoder blocks contain a single up-convolutional layer with a 2×2

kernel size for up-sampling and also two convolutional layers with 3x3 ReLU activations. Additionally, skip-connections link together the down-sampling and up-sampling blocks on the same level. This allows transferring feature information from encoder-levels to decoder-levels with the same resolution [103, 104]. Through its structure, the U-Net extracts low-resolution and high-resolution information for which the low-resolution information is utilized for improving accuracy, whereas the high-resolution information is utilized for improving the extraction of complex features [104]. Through this functionality, U-Net architecture allows individual predictions for each pixel with high performance. In recent years, a large number of novel architectures were developed which are heavily based on the U-Net [13, 17, 103, 104, 111]. The new designs try improving the U-Net with layer changes, slight modifications, or additional modules, but still rely on the proven ‘U’ shape. Zhang et al. [112] and Kolařík et al. [113] introduced further connections inside the convolutional blocks for allowing more complex pattern findings resulting in the Res-UNet [112] and Dense-UNet [113]. Ibtehaz et al. [114] replaced the two convolutional layer blocks with a ‘MultiRes’ block, consisting of four convolutional layers in which one layer is a residual path, and changed the skip-connections to so-called ‘res path’ connections, consisting of further convolutional layers [114]. Another variant is the integration of attention mechanisms into the U-Net by Oktay et al. [115]. The authors introduced attention gates that enhance skip-connections by weighting activation functions for irrelevant regions reducing the number of redundant features [115]. Similar to image classification, 3D variants of segmentation architectures were developed by simply transforming the corresponding layers. Thus, 3D variants of the U-Net were quickly published like the V-Net by Milletari et al. [116] or the 3D U-Net by Çiçek et al. [117]. Further variants which are presented in Figure 2.14 but not introduced are the Compact U-Net, which is a denser version of the Dense-UNet by Kolařík et al. [117], and the plain U-Net, which is a stricter configured variant compared to the more dynamic standard implementation [108, 109, 118].

2.3.3 Hardware Requirements

Modern CV pipelines utilizing the latest architectures for optimal performance require high hardware specifications [14, 119, 120]. Parallelized dense computation and large memory bandwidth are crucial for modern computation-intensive deep neural networks [14]. Central processing units (CPUs) do not fulfill these rising hardware requirements which is why neural network computations are relocated to graphics processing units (GPUs) [14]. GPUs provide next to powerful graphics engines also highly parallelized computation and memory bandwidth capabilities which are ideal for high-performance computing as required in deep neural networks [14]. Next to the core clock speed, the available video random access memory (VRAM) of a GPU determines its capabilities to store a deep learning architecture in its memory [14, 119]. State-of-the-art deep learning architectures often require the usage of current high-end GPUs and, therefore, the available GPU hardware is an essential factor to consider for assessing the research capabilities of a lab [14, 119]. This is why larger labs or industries maintain expensive computing infrastructures or even clusters in order to offer competitive research capabilities. However, it is important to note that for application, drastically less

computation resources are commonly available which is why there is a certain balance to consider between applicability and best-possible performance.

Nevertheless, CPUs are still strongly incorporated for preprocessing in CV pipelines [14, 119]. Modern pipelines utilize real-time image processing as well as augmentation (see Chapters 3.2 and 3.3) leading to high CPU usage during training, as well. During the development of a CV pipeline, it is often required to deal with bottlenecks originating either from the CPU, for which the GPU is waiting to obtain the next processed images, or from the GPU, for which the CPU is waiting to pass the already processed next images.

2.4 Medical Image Analysis

Through the increased availability and usage of medical imaging like magnetic resonance imaging or computed tomography, the need for automated processing of scanned imaging data is quite strong [10, 19, 121]. Currently, the evaluation of medical images is a manual process performed by physicians [19]. Larger numbers of slices require the inspection of even more image material by doctors, especially regarding the increased usage of high-resolution medical imaging [10, 19, 121].

The field of medical image analysis (MIA) describes the computer-assisted processing, analysis, or visualization of medical images to support medical staff in diagnostic, therapy, and documentation [10, 19, 28]. The general idea of MIA is to assist physicians in processing the continuously increasing quantity of medical images and to provide additional information [10, 28]. An important key concept in MIA is the automated analysis of medical images in order to provide clinical decision support (further introduced in Chapter 2.4.2). This can be achieved by integrating image processing algorithms, statistical models, or computer vision methods into clinical workflows [10, 28, 122]. Common application fields of MIA tools are medical disciplines that heavily utilize medical imaging in routine like radiology, pathology, oncology, neurology, and surgery [10, 28, 122]. Through the usage of MIA, clinicians strive to achieve improved healthcare quality as well as a reduction of time-consuming processes [10, 122].

The first ideas for MIA emerged in the year 1969 with workshops about information processing in medical imaging, and peak today with the utilization of artificial intelligence models [122]. The history of MIA can be summarized in four eras according to Duncan et al. [122]:

“

- 1. *pre-1980 to 1984, that one could term the era of 2D image analysis,*
- 2. *1985-1991, when knowledge-based strategies came to the forefront and the advent of Magnetic Resonance Imaging (MRI) changed the landscape,*
- 3. *1992-1998, when the analysis of fully 3D images became a key goal and more mathematical-model-driven approaches became computationally feasible, and*
- 4. *1999 and beyond, where now advanced imaging and computing technology is facilitating work in image-guided procedures and more realistic visualizations.*

“

There are various applications in MIA as well as associated objectives depending on the imaging modality, discipline, medical condition of the patient, and the particular task [10, 122]. Nevertheless, the applications in the field can be grouped in the following universal objectives: Consolidation as well as centralization of information from diverse sources, information gain by analysis, evaluation, as well as interpolation, and information reduction by summarizing as well as relevant selection.

2.4.1 Research Fields

Through the wide definition of MIA, multiple subfields focusing on different tasks in medicine emerged. These subfields differ in individual tasks, objectives, and processing methods. In this subchapter, the five major subfields in MIA are introduced to provide a brief overview of the field.

Visualization

The subfield visualization focuses on the efficient presentation of medical information [10]. The objective in this field is to improve diagnostic, treatment planning as well as execution, and monitoring of patients through supportive information displaying [10]. This can be achieved by an efficient display of the medical image and optional integration of further anatomical or pathology knowledge into the representation. The domain is fundamental in clinical routine as well as research and is present in any aspect of medical imaging [10, 19]. Efficient and smart overview through image viewer and illustrations of relevant medical information, specifically 3D visualization, is an important key domain in MIA [10, 19, 28]. The spatial display of an object like a tumor can provide crucial insights and help surgery and radiotherapy treatment procedures [10]. Another more recent domain is virtual reality in which clinicians can visualize objects in 3D near-eye display for immersive insights which can be helpful for surgery as well as education [10, 28, 120].

Medical Image Registration

The subfield medical image registration (MIR) focuses on the translation of different imaging matrices into a single and uniform coordinate system [10]. The objective of this domain is to combine multiple images into a single one in order to obtain a comprehensive visualization that allows displaying information from both images [10, 123]. A processed image consisting of multiple aligned images is called a multi-modal image while the process of registration is also often called image fusion [10, 123]. This procedure allows the direct comparison of images from either single or even multiple patients. Furthermore, the combination of different imaging modalities is also possible and commonly applied for CT or MRI with PET scans [10, 123]. Another important application domain is the registration of a segmentation atlas in which annotated regions are overlaid on a patient image to incorporate knowledge like the different human brain regions [10, 123]. The general idea behind a registration operation is to transform one image to be optimally aligned with another image. This process can be summarized in four core steps [10]. In the first step, distinctive spots in both images are identified. In the second step, these points are matched between the images. In the third step, the transformation for optimal consensus based on the distance between matching points is computed, and, finally, the computed transformation is applied in which the images are aligned.

Medical Image Segmentation and Object Detection

The subfield medical image segmentation (MIS) and object detection (OD) focus on the identification and partition of objects within a medical image [10, 124, 125]. The objective of this domain is to detect and highlight anatomical or pathological structures in an image [10, 124, 125]. Such a structure or object is also called a region of interest (ROI) [10]. The benefits of MIS and OD range from decision support in diagnosis or treatment, integration of knowledge, and automation up to documentation [10, 124, 125]. Furthermore, there is also a wide range of applications like brain region partition, organ segmentation, tumor detection, and cell counting in histopathology [10, 124, 125]. In general, MIS and OD methods are often implemented for the automated processing of medical images and are commonly the first step in an image interpretation pipeline to define objects within an image matrix [10, 124]. The domain can be grouped into four types of segmentation [10, 124]: The manual annotation by clinicians or radiologists, the atlas-based segmentation with MIR, the automated object detection of ROIs, and the automated semantic segmentation in which each pixel is classified.

As the field of medical image segmentation is one of the core topics in this thesis, an in-detail introduction is presented in Chapter 4.1.

Quantitative Image Analysis

The subfield quantitative image analysis (QIA) focuses on measuring objects and features within medical images [10]. The objective of this domain is to determine quantitative descriptions of relevant image objects in order to assess medical information [10]. Furthermore, the field of QIA allows the quantitative assessment of object characteristics which can be utilized as explicit norms for treatment instructions [10, 28]. Thus, QIA provides essential information for diagnostic, treatment planning, and progression control of medical conditions [10]. Popular applications of QIA are in oncology for tumor analysis and surgery [10]. QIA methods often require a prior MIS for object acquisition and can provide features that are further used in medical image classification [10, 28]. The domain can be grouped into four major types [10, 54]: Elemental image analysis operations like the computation of the diameter, angle, or volume of an object, texture analysis of surfaces for tissue detection, fractal image analysis for contour detection, and morphological image analysis for assessing the form as well as structure of an object. Overall, QIA processes are widespread in all kinds of medical imaging software and corresponding methods can be found in any modern medical imaging viewer [10, 54].

Medical Image Classification

The subfield medical image classification (MIC) focuses on the classification and detection of images on the basis of visual content [10, 13]. The objective of this domain is to structure and categorize the image into predefined medical classes based on diseases or other medical features [10, 13]. Especially for diagnostics, the field of MIC offers essential information by interpreting medical images. MIC methods are commonly the last part of traditional MIA pipelines by combining all prior computed features like quantitative information based on an MIS to

prognose severity or detect diseases [10]. Typical applications of MIC are tumor type classification, Gleason score grading of prostate carcinoma, fracture risk detection, and melanoma detection [10]. In general, MIC can be utilized for assigning images to any classification system which provides numerous application capabilities in the medical context [10, 13]. The domain can be grouped into three types of classification: Binary classification in which images are grouped into two classes like healthy or diseased, ordinal classification in which the classes have a relative order in an arbitrary scale like the Gleason grading system, and nominal in which classification systems consist of distinct labels like types of brain tumors (glioblastoma, acoustic neuroma, astrocytoma, ...).

As the field of medical image classification is one of the core topics in this thesis, an in-detail introduction is presented in Chapter 5.1.

2.4.2 Clinical Decision Support

The practical application of medical image analysis algorithms for computer-based and automatic analysis of medical images is defined as clinical decision support (CDS) [10, 24, 25]. These systems assist clinical decision-making in diagnostic and therapy by providing analysis reports and additional information [10, 24, 25]. Sutton et al. [24] defined CDS systems in their work as follows:

“A clinical decision support system (CDSS) is intended to improve healthcare delivery by enhancing medical decisions with targeted clinical knowledge, patient information, and other health information.”

The range of CDS applications can be categorized into knowledge-based and non-knowledge-based systems [24, 25]. A knowledge-based system utilizes explicit as well as predefined rules, which can also be represented in ‘IF-THEN’ statements, according to gained knowledge from literature or experience [24]. In contrast, a non-knowledge-based system utilizes a machine learning algorithm or statistical model which gains knowledge on its own commonly through a training process instead of using explicit knowledge definitions by experts [24]. Furthermore, Sutton et al. [24] described the types of CDS throughout history as follows:

“A traditional CDSS is comprised of software designed to be a direct aid to clinical-decision making, in which the characteristics of an individual patient are matched to a computerized clinical knowledge base and patient-specific assessments or recommendations are then presented to the clinician for a decision. CDSSs today are primarily used at the point-of-care, for the clinician to combine their knowledge with information or suggestions provided by the CDSS. Increasingly however, there are CDSS being developed with the capability to leverage data and observations otherwise unobtainable or uninterpretable by humans.”

The integration of a CDS system can yield multiple benefits for clinicians [24, 25]. A CDS system can increase decision reliability by providing further insights or additional information, improve the quality of health care in terms of diagnosis as well as treatment, prevention of

potential errors or adverse effects, and increase time and cost efficiency through structured reporting or automated documentation [24, 25].

Establishing a medical image analysis method as CDS in a medical workflow is the ultimate goal in the field [10]. However, the integration of a tool in such a sensitive environment reveals the following crucial challenges: Through the medical context, the applied software must be validated for being functional, fail-safe, and reliable in any possible scenario [24, 25]. Moreover, clinicians need to trust the computed results of the tool for worthwhile incorporation in their decision-making [24, 25]. To achieve this, model transparency and concept understanding are important aspects to factor in next to providing specific training lessons for users in order to enhance user experience [24, 25, 126]. Finally, it is essential to ensure the maintenance of the software for multiple years which requires a standardized and robust implementation [24, 25]. Standardized solutions, which are easier to maintain and are utilized in multiple locations, provide a natural advantage compared to custom implementations based on the larger community and the available knowledge in user experience.

3

Workflow in Medical Image Analysis

The field of medical image analysis is a broad spectrum including subfields like visualization, classification, segmentation, registration, and reconstruction. Still, many aspects and principles of medical image analysis methods can be summarized as extracting information from medical images to support or automate medical processes. Furthermore, the subfields of deep learning based medical image classification and segmentation share fundamental parts and methods in workflows.

A deep learning based medical image analysis workflow can be defined to contain five core steps: Data loading or management which handles the input as well as output processes of the pipeline, data (image) augmentation which is an essential technique in any medical-related deep

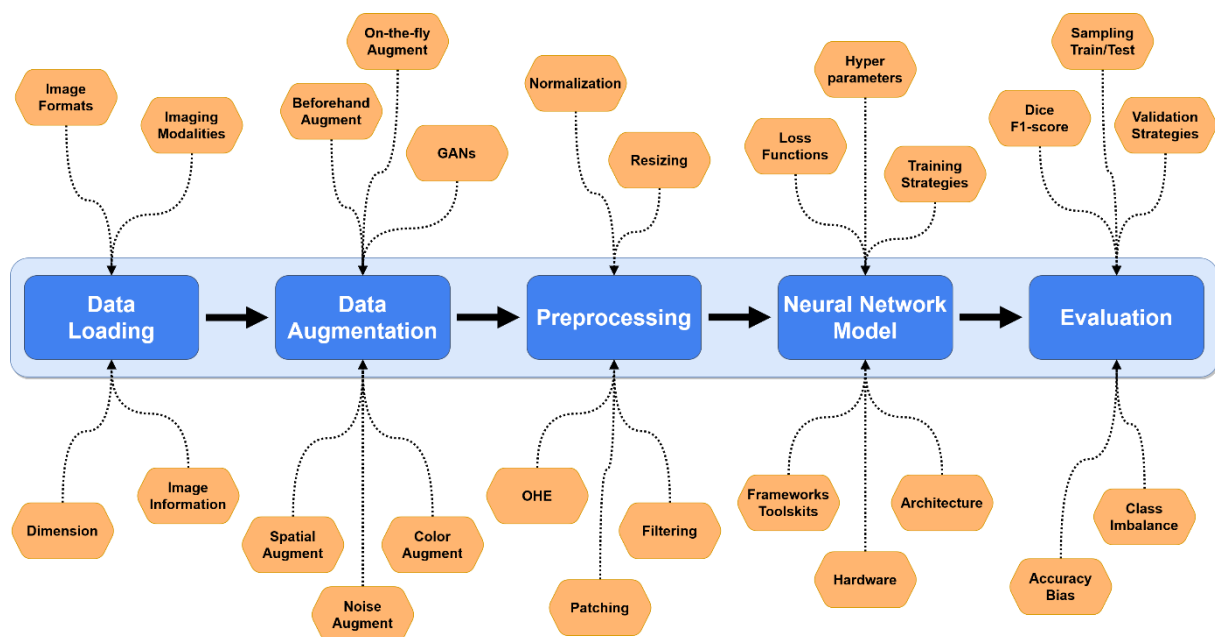


Figure 3.1: Illustration of a typical workflow in deep learning based medical image analysis.

learning approach due to the often limited dataset sizes, preprocessing which describes the applied methods between image loading and passing to the model, the neural network model consisting of a state-of-the-art convolutional neural network architecture for computer vision, and the evaluation process for performance assessment.

As the core topic of this thesis is deep learning based medical image analysis with a focus on image classification and segmentation, the following chapter presents a broad review of the field of medical image analysis based on the Author's gained experience from 15 studies as well as the Author's acquired knowledge from an extensive meta-analysis of the research field including more than 250 publications over the last four years.

3.1 Data Management

The data and information in medical imaging are represented in a wide landscape of different formats and structures. Even though the core of every medical image is a projection of some object or state, the representation of this projection, the storage, the creation technique, the circumstances during creation, the technical artifacts or bias, and naturally the information itself can widely differ. The first step in a medical image analysis pipeline is data loading and parsing out all relevant information. Thus, this chapter summarizes the common and popular data types for medical image analysis. More information about medical imaging modalities can be found in Chapter 2.1.1.

3.1.1 Imaging Formats in Medicine

There are various methods and formats to encode imaging information which are highly dependent on the specific use case: For medical information exchange between physicians and telemedical approaches, as additional information for the patient, or for publication within teaching or research scope [54]. Also, the source of a medical image varies from high-quality imaging originating from clinical devices to low-quality imaging made with smartphones for e-health. In general, image formats differ in the following criteria: Patient information or anonymization, medical imaging metadata, and image quality.

DICOM

The most used format for medical imaging is the DICOM format which is short for Digital Imaging and Communications in Medicine [10, 54, 127]. It is a worldwide standard for the communication and management of medical imaging information and related data [54]. The format allows interoperability between all kinds of devices from medical scanners, workstations, and printers to communication servers and PACS (Picture Archiving and Communication System) [10, 54]. The international uniform format supports fast and effective information exchange for healthcare between hospitals and across borders [127]. Distinctively, the DICOM format contains much more than just the image. It also includes all kinds of medical metadata: Patient information (like id, name, sex, diagnosis) provenance information (like treating physician, time, clinic), modality information (like CT, MRI, endoscopy), and equipment information (like scanner model, scanning configuration, capturing parameters, image meta information) [10, 127]. DICOM images are the typical format to work with in a clinical environment, but are rarely used for public datasets due to the high content of patient identifying information.

NIFTI

The Neuroimaging Informatics Technology Initiative (NIFTI) file format was initially created to speed up the development and enhancement of informatics tools in neuroimaging [128]. The format allows the storage of anonymous 3D imaging data like MRI and CT scans including important metadata like slice thickness, but without the patient identifying data like the DICOM

format. Thus, it is now commonly used for sharing public as well as anonymous MRI and CT scans, instead of only for brain imaging, but also for all kinds of human 3D imaging [129–131].

Regular Image Formats

Mainly for sharing 2D medical images with patients as well as for publication, regular image formats aside from medicine are often utilized [51, 54]. Shortly summarized, there are three important image formats [54]: Joint Photographic Experts Group (JPG) is one of the most commonly used imaging formats which utilizes lossy compression to achieve small file sizes for easier exchange. The more modern Portable Network Graphics (PNG) raster-graphics file format supports lossless data compression and allows more color depths with qualitative better visualization than JPG. Professional medical 2D imaging datasets often use the Tagged Image File Format (TIFF). TIFF offers the best image quality without information loss and is commonly used in professional photography and printing. One of the exceptions to the usage of DICOM in medical imaging is the field of histopathology. Microscopes produce images with a resolution of typically 80,000x60,000 pixels which are stored in the bigTIFF format (an extended version of TIFF for images with more than 4GB) [120, 132, 133]. Nevertheless, regular image formats do not contain any medical metadata and are highly dependent on the use case as well as on the image source which can be an advantage in terms of anonymization and a disadvantage in terms of appropriate preprocessing methods requiring metadata.

Other Formats

Besides commonly accepted formats especially for medical images, there is a variety of other imaging or data storage formats in which medical images can be stored. Especially in research, datasets can be encoded in unusual or custom formats. In the research field of medical image processing, SimpleITK is a toolkit library for multiple programming languages like R and Python [134, 135]. It also provides data loading and writing operations for the *MHA* (MetaImage) format which can be utilized for storing metadata along graphical information [136]. Through the popularity of the SimpleITK framework, more organizations utilize *MHA* for encoding complex medical imaging data like CT scans based on the format capability to dynamically store any annotation as well as mandatory meta information like voxel spacing [137]. An alternative to imaging formats is any type of data storage format. Especially already preprocessed datasets are present in popular data formats like HDF5 [136] or NPY (NumPy file format) [138]. Working with these preprocessed images has to be done with caution due to non-revertable already applied methods which are often not sufficiently documented.

3.1.2 Image Dimensions

The dimension of an image defines the number of axes and, therefore, the mathematical representation of the imaging data. In contrast to regular imaging like photography, medical imaging consists of common 2D images as well as more complex 3D images.

2D Imaging

The majority of computer vision research is based on ordinary 2D imaging [9, 10, 75, 77]. Thus, various frameworks for 2D image processing as well as 2D deep learning libraries and deep convolutional pipeline APIs (application programming interface) are available, which can also be utilized for medical imaging [13, 75, 139, 140]. The resolutions of 2D medical images are quite diverse. Whereas ultrasound has a lower resolution like 0.2 to 1.0 megapixels (5,122, 7,682 or 10,242 pixels), visible light imaging approaches like endoscopy around 1 to 8 megapixels or X-ray imaging like mammography around 20 megapixels [10, 120]. However, it is common to down-scale such high-resolution images to more common resolutions like 2,562, 5,122, or 10,242 pixels in medical image analysis pipelines [74, 118].

3D Imaging

In contrast to regular 2D images, medical imaging also consists of 3D images. However, through the rare occurrence of 3D imaging in regular imaging, there are only a handful of computer vision frameworks available for 3D image processing [134, 135, 141, 142]. Additionally, specific medical imaging frameworks and deep convolutional neural network architectures are needed for handling 3D imaging data, which is explained in more detail in Chapter 2.3. 3D imaging is commonly based on tomography approaches like CT, PET, or MRI. The tomography technique creates 2D cross-sections, also called slices, of a body with the utilized method like X-rays [10, 54]. Afterward, the multiple created 2D slices are assembled into a single 3D image [10]. The slice resolution for such approaches ranges normally between 2,562 and 5,122 pixels with around 200 slices per 3D image (e.g. 512x512x200) [10, 120].

2.5D Imaging

Various medical image analysis studies integrated an intermediate alternative by utilizing popular 2D frameworks as well as architectures for 3D imaging which is defined as 2.5D imaging [11, 13]. 2.5D pipelines split the 3D images into slices and process these 2D slices individually [11, 100, 101]. Afterward, the slices with their individual predictions are collected and assembled back into the 3D matrix [11, 100, 101]. This approach allows not only the application of 2D imaging methods but also avoids the required extensive hardware resources for processing 3D data [100]. However, processing each slice of a 3D scan individually leads to inevitable information loss. The presence of an interesting pattern in another slice and only cross-sectional ROI visibility could be missing or be not enough information for efficient model learning, which could have a drastically negative impact on model performance.

3.1.3 Image Information: Representation, Annotation, and Metadata

Medical imaging generates data units that consist of multiple different information types besides the actual image. In a medical image analysis pipeline, there are three important information categories defining each individual image:

Representation

The actual image can be stored and represented via different techniques which are highly dependent on the modality. In general, the pixel intensity, also called pixel information, is encoded via one or multiple integers. The number of integers per pixel is also defined as the number of channels for this image. For pixel intensity encoding, there are diverse formats that are different in their number of channels and possible integer ranges. The most commonly known encodings in regular 2D imaging are grayscale and RGB (Red-Green-Blue) [10, 54]. Whereas grayscale is a single-channel encoding describing the gray intensity with an integer range between 0 to 255, the RGB encoding is defined by three channels describing the red, green, and blue intensity with each channel using an integer range between 0 to 255 [10, 54]. Nevertheless, medical imaging techniques, which do not rely on visible light imaging, compute different measurements as their intensity values. However, besides some exceptions, often these intensity values are automatically converted to grayscale. An exception to this rule is CT intensity values which are often represented through the Hounsfield scale. Hounsfield Units (HU) are units on a quantitative scale for describing the radiodensity and have fixed value ranges for specific human tissues as shown in Table 3.1 [10, 54, 143–145]. Such intensity encodings can be exploited for MIA detection of specific tissues, which is commonly applied in clipping based preprocessing (described in detail in Chapter 3.3.5) [10, 146].

Table 3.1: Overview of Hounsfield unit ranges for selected tissue types.

Tissue Type	Hounsfield Value Interval
Air	−1,000
Lung tissue	−900 to −170
Fat tissue	−120 to −50
Water (H ₂ O)	0
White matter	20 to 30
Kidney	20 to 45
Grey matter	37 to 45
Muscle	35 to 55
Liver	45 to 65
Bone	700 to 3,000

Furthermore, multi-modal encoding-based architectures are prevalent in many MIA pipelines [13, 147–149]. Multi-modal image representations are the pixel-wise combination of multiple modalities by registration of multiple modality images in the same image coordinate system. Thus, a single pixel can have a channel for each modality. Commonly, multi-modal encodings are often utilized in pipelines based on PET-CT imaging data or multiple MRI sequence imaging data like T1-T2 [13, 147–149]. Instead of analyzing modality images individually by separate models, multi-modal encodings allow utilizing and learning from multiple modality information at the same time. This can result in significantly increased performance, especially in the detection of false positives which appear as an abnormality in one modality but can be differentiated in another modality [147, 150].

Annotation

Next to the image, an MIA pipeline requires an annotation or mask for the image in order to train a model for identifying the correct classification or annotated pixel of the ROI. For image classification, annotations are text-based classes, also called labels, which are assigned for a complete image. For multi-label classification, a single image can also have multiple classes assigned. In contrast, a mask for image segmentation has an identical shape as the actual image but the pixel values consist of a class integer instead of an intensity value. Therefore, each pixel is labeled to commonly one specific class. Overall, there are three types of annotations: Binary, in which each pixel or image can only belong to one of 2 classes (normal or cancer), multi-class, in which each pixel or image can belong to one of the n classes (background or lungs or cancer ...) and multi-label, in which each pixel or image can belong to multiple n classes (0 or 1+3 or 2+3 ...). Most of the time, the first class acts as a background or normal (control) class in contrast to the target classes. Further information on classification theory can be found in the excellent works of Zhou et al. [151] and Sorower [152].

Besides the different types of annotation for MIA, the quality of annotation can highly vary depending on several factors. The first one is the expertise of the annotator, which can range

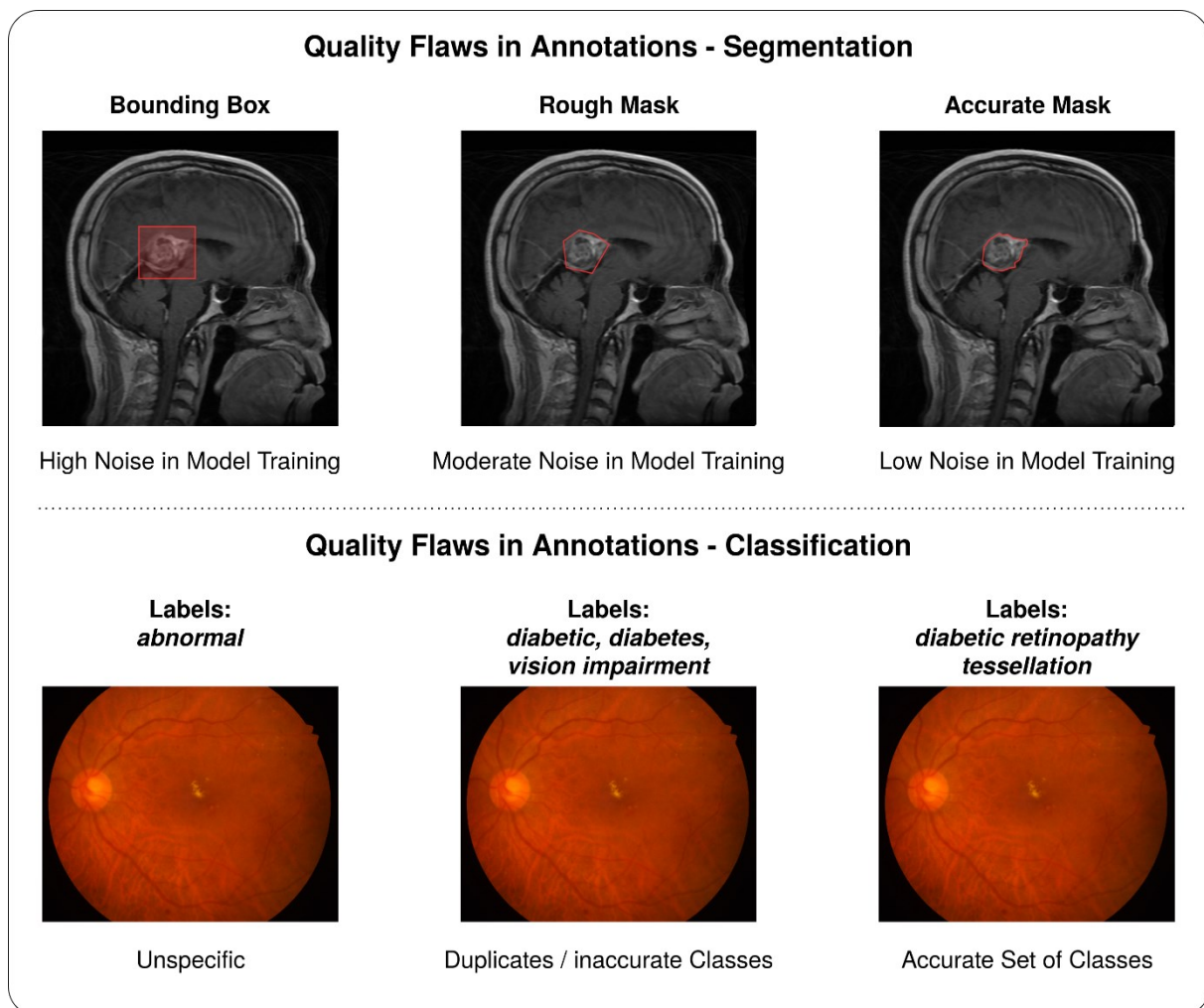


Figure 3.2: Illustration of quality flaws in annotations for medical image analysis.

from student research assistants to experts like radiologists for CT images [153, 154]. However, not only the field but also the experience from years of practice can have a large impact on the annotation quality as studies revealed [8, 155]. The second factor is annotation processing. Common techniques are applying filters or other refinement techniques for smoothing manually drawn ROIs with possible sloppy lines [129, 153, 156]. The last one of the three important factors is the annotation generation process. For image segmentation, masks can be either drawn per hand, automatically generated via thresholding from ROIs, or by just cutting out the complete area of ROIs [153, 156]. For image classification, annotation labels can be manually assigned by a predefined set of classes, automatically identified by using structured reports with standardized terminologies, or predicted by natural language processing techniques on medical notes [156]. All methods result in valid annotations but can drastically differ in quality [156]. In total, manually drawn and afterward refined annotations masks or manually assigned annotation labels from a skilled expert in the corresponding field can be seen as a gold-standard for MIA annotation.

Common quality flaws in the annotation of medical imaging are illustrated in Figure 3.2. The figure demonstrates the annotation quality based on noise presence in masks for image segmentation and labels for image classification. As datasets, MRI brain scans for tumor segmentation from Cheng et al. [60, 157] and fundus photography for disease classification from Pachade et al. [158] were used.

Metadata

In addition to the two core information of an MIA pipeline, the image representation, and the annotation, medical images often include metadata. Metadata is information describing supplementary details on the patient, the disease status, the image generation process, or technical details. As already stated in Chapter 3.1.1, DICOM files regularly contain various patient-related information, however also anonymized datasets can contain descriptive information about the patient like age, sex, or disease status and severity [54, 127]. This information can be utilized in an advanced pipeline build like cascading pipelines based on multiple models [132, 159, 160]. Another important piece of information is the pixel spacing, also called slice thickness, which defines the size of a pixel in 3D images and is required for the resampling method during the preprocessing [127, 161]. For multi-modal approaches, registration of images from different modalities is needed. This requires the origin of the image coordinate system (final image with the patient in the center) in order to convert the image back into a world coordinate system (coordinates showing scanner and patient position which highly varies depending on setup and device) [123, 162]. Furthermore, technical details like device information and configuration can be used for advanced image preprocessing and refinement. In order to avoid unnecessary bias, it is always recommended to check for information about image generation and already or automatically applied preprocessing methods.

3.2 Image Augmentation

The process of data augmentation can be defined as the technique of artificially increasing the dataset by creating modified or simulated images based on the available information. Typically, images of the dataset are transformed or utilized to create new images [163, 164]. The medical image analysis field often lacks adequate-sized datasets because of the labor and time-intensive annotation process by physicians [13, 110]. Therefore, data augmentation is present in any state-of-the-art MIA pipeline and has a huge impact on the model performance [163–166]. There are several advantages in the utilization of data augmentation.

The training of a model has always the risk to ‘overfit’ on the training data, which means that non-relevant information or complete memorization of training samples is used for the prediction [163]. This scenario is called overfitting and leads to increased or even optimal performance on the training data. However, through overfitting, the model is not able to perform reasonable predictions on unseen or new data, which significantly reduces the model’s usefulness [163]. Increasing the dataset size by data augmentation lowers the risk of overfitting [163–166]. It follows the assumption that through image transformations, the data usage efficiency can be increased by extracting more information from the dataset. Another perspective of data augmentation is the variance enlargement in the dataset. The model performance is determined by its detection and generalization ability [163]. Medical imaging datasets are known to have a small sample size and similar images with a low variance which leads to low generalizability [110, 118]. Especially in medical images in which a high number of features is shared, data augmentation is able to introduce reasonable variance by creating spatial transformation of abnormalities or noise addition. In addition, data augmentation is able to tackle class-imbalanced datasets by either augmenting complete samples with an overall underrepresented class or by cropping or zooming specifically in segmentation areas with the underrepresented class [163, 164, 167].

Still, performing data augmentation in an MIA pipeline also reveals challenges. In contrast to common augmentation methods for image classification, image segmentation also requires the corresponding augmentation of the segmentation mask as well. Transformations have to be applied equally to the representation image itself as well as the segmentation mask to be valid. Particularly in 3D imaging, this can be a challenging task that is only supported by a niche of data augmentation frameworks [141]. Also, not all commonly used data augmentation methods on regular imaging are reasonable for medical images in which introduced discrepancies are either non-logical or misleading in the medical context. Popular data augmentation frameworks for medical image segmentation are batchgenerators by Isensee et al. [142] and torchio by Pérez-García et al. [141]. Furthermore, it is important to note that data augmentation is purely based on increasing the information usage efficiency of images. It is not possible to create new information through data augmentation (data processing inequality) which is why it can, correctly applied, significantly decrease the risk of overfitting, but also significantly increase the risk if applied thoughtlessly.

3.2.1 Types of Image Augmentation

Artificial images from image augmentation can be generated with diverse methods. These methods can be categorized as regular image augmentation and more complex generative adversarial networks. A demonstration of image augmentation techniques can be seen in Figure 3.3. In the figure, the techniques are categorized into spatial augmentations, color augmentations, noise augmentations, and generative adversarial networks. For the generative adversarial network, a super-resolution generative adversarial network (SRGAN) architecture by Ledig et al. [90] was utilized. The example images originate from the RFMiD dataset [158].

Regular Image Augmentation

The majority of applied image augmentation methods can be defined as functions which are modifying a single and specific feature. These augmentation methods are based on simple image operations or mathematical functions which are usually not computationally intensive and do not require preprocessing. The aim of regular image augmentation is to generate valid new images in a medical context as well as alter image-capturing perspectives or simulate technical variance like Rician noise in MRI scans [163]. This kind of image augmentation is a wide and established scientific field in which methods are utilized in the vast majority of state-of-the-art MIA pipelines [163]. The augmentation functions can be grouped into three types: Spatial augmentations, which modify position-dependent information like image translations or rotations; color augmentations, which alter the intensity values of the image like contrast or brightness; and noise augmentations, which introduce artifacts in an image like Gaussian Noise or Rician Noise [142, 168, 169].

Generative Adversarial Networks

In contrast to regular image augmentation, generative adversarial networks, short GANs, have been a popular alternative for several years [163, 167, 170, 171]. GANs aim to generate new

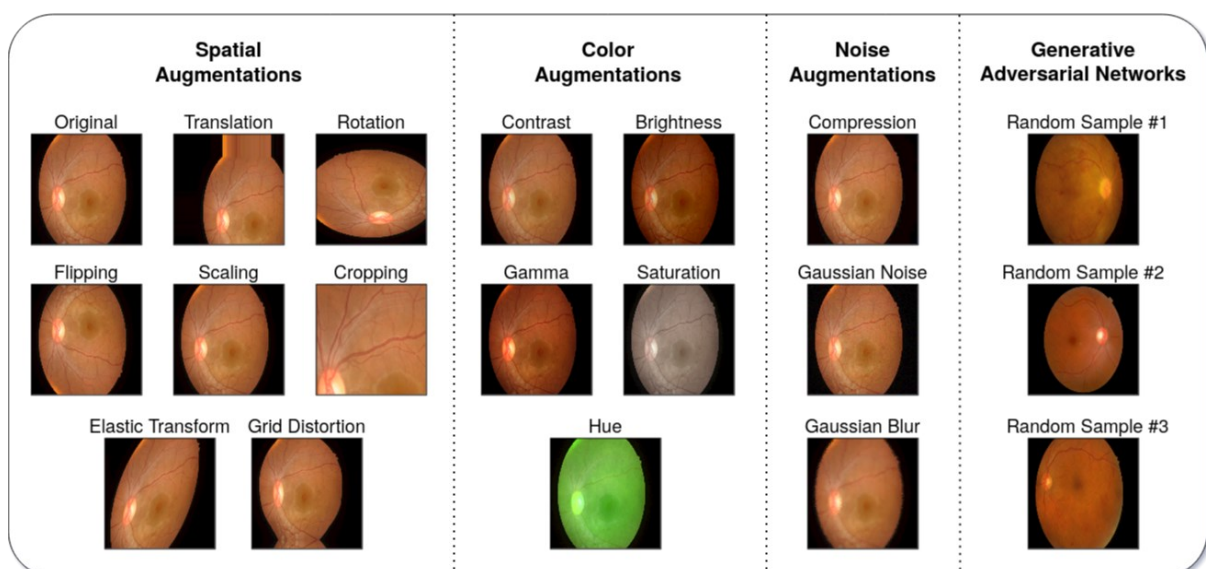


Figure 3.3: Overview on common image augmentation techniques.

images which are reasonable and not distinguishable as artificial. Instead of image creation by modifying specific features, GANs are separate neural network models which try to learn and then generate similar probability distributions for specific regions of interest [163, 167]. This allows the generation of augmented images with more complex random variable modification. However, in medical image classification and segmentation challenges, which allow direct comparison of methods on the same dataset, pipelines based on GAN augmentation are often inferior to the other data augmentation techniques [101, 129, 163, 171–173]. Thus, GANs in modern competitive MIA pipelines have become rare in recent years.

3.2.2 Application Techniques

Besides the transformation types, data augmentation can be applied with two different techniques.

On-the-fly Augmentation

The majority of high-performing MIA pipelines are utilizing on-the-fly augmentation, which is also called online or real-time augmentation [109, 110, 118, 159, 174]. The principle of this technique is to take the original dataset as templates and compute randomly transformed variants of it during run-time. Simplified, instead of using the original image for the training, a novel augmented image is created and fed into the model. The aim of on-the-fly data augmentation is that a model never sees the same image twice in the complete training process. In the last few years, this technique became the most favored data augmentation technique in medical image analysis [49, 109, 118, 142, 175]. Still, applying real-time data augmentation besides the actual training process is a computationally expensive task. If no sufficient hardware is available, on-the-fly data augmentation can become a computational bottleneck that drastically increases training time.

Prior Augmentation

The traditional approach for utilizing data augmentation is to apply transformations a single time as preprocessing technique [163]. The aim is to artificially increase the small dataset size to the individually preferred distribution and then start the training process on the new larger dataset. Thus, the new dataset consists of the original and the augmented images. Whereas this approach is still popular for up-sampling purposes in datasets with strong class imbalance, experiments showed that the efficiency and performance gain of prior augmentation is inferior compared to on-the-fly augmentation [118, 163, 176]. This can be explained by the larger variance which is introduced by augmenting an image each time the image is used.

3.3 Image Preprocessing

The preprocessing of medical images is a broad but imprecise defined procedure of data conversion and refinement before feeding it to the computer vision algorithm. The two main purposes of preprocessing in machine learning pipelines are making the data machine-readable and simplifying the task. Therefore, preprocessing is an important step in a machine learning pipeline as the quality of data and the useful information that can be derived from it, directly affects the learning ability of the model [161, 177]. Identical to a machine learning pipeline, a deep learning based MIA pipeline also requires comprehensive preprocessing to gain machine readability and task simplification. Because the application of preprocessing techniques is highly dataset and task-specific, there is no strict guideline on which to use [10]. Instead, the wide landscape of preprocessing methods can be distinguished into two types depending on each individual use case: Necessary methods to be fed into a deep convolutional neural network model. Recommended methods to possibly improve performance, reduce bias, lower required hardware resources, or training time. In the following, a short description, relevance, and use case of the 10 most important and popular preprocessing techniques for medical image analysis are presented. As an overview, Table 3.2 summarizes all presented preprocessing techniques.

Table 3.2: Overview of preprocessing techniques including their types, uses, and application references.

Preprocessing Technique	Type	Influence on	Used in
One Hot Encoding	Possibly Necessary	Machine Readability	[105, 160, 178]
Normalization	Recommended	Task Simplification	[49, 109, 161, 179]
Image Filtering	Recommended	Bias Reduction & Task Simplification	[129, 177, 180, 181]
Bias Correction in Medical Imaging	Recommended	Bias Reduction	[169, 182, 183]
Clipping	Recommended	Task Simplification	[109, 110, 184, 185]
Padding	Possibly Necessary	Architecture Compatibility	[146, 186–189]
Resizing	Possibly Necessary	Lowering Hardware Requirements	[49, 109, 110, 190]
Resampling	Possibly Necessary	Lowering Hardware Requirements & Task Simplification	[49, 109, 110, 118]
Patching	Possibly Necessary	Lowering Hardware Requirements	[49, 109, 110, 141]
Cropping	Recommended	Task Simplification	[109, 160, 191, 192]

For demonstration purposes, the presented image preprocessing methods (except bias correction) are visualized in Figure 3.5 and Figure 3.6. The utilized image is a thorax CT scan with a diagnosed COVID-19 pneumonia from the STOIC project (Revel et al.) [137].

3.3.1 One Hot Encoding

The one hot encoding technique describes the process of converting a single categorical non-machine-readable variable into multiple binary machine-readable variables [193]. This kind of conversion is also called creating dummy variables in the field of statistics [161]. The training of a model for semantic segmentation tasks requires a mask in which each pixel in the image is

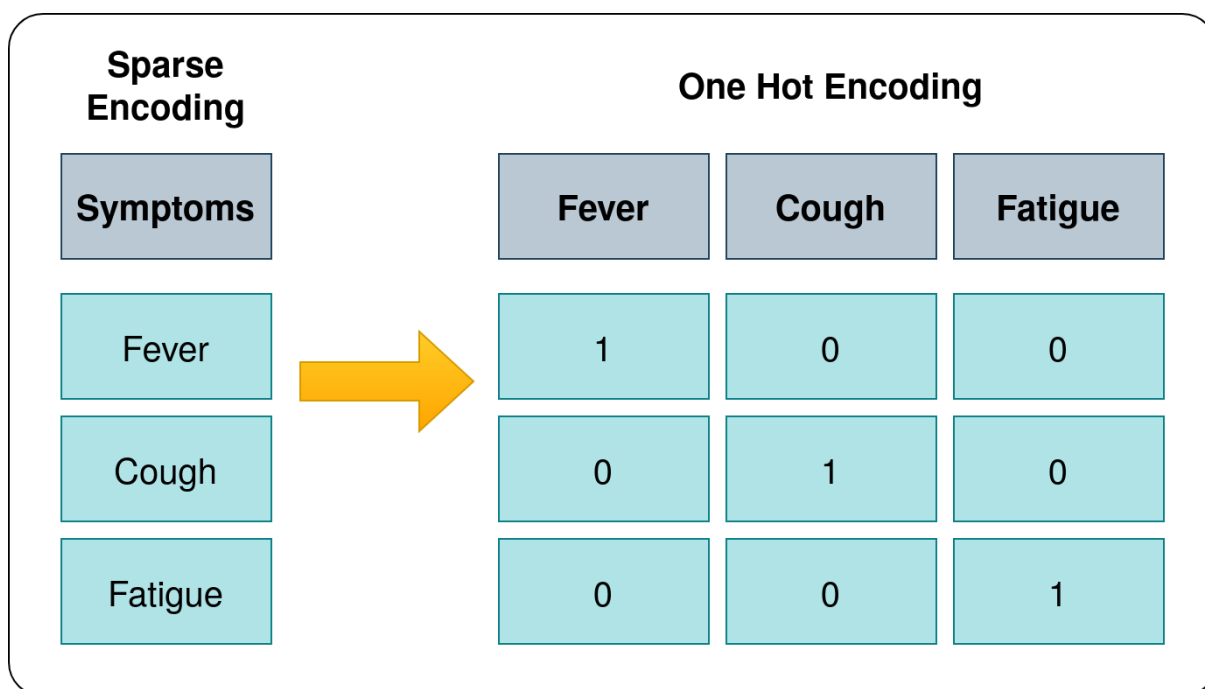


Figure 3.4: Visualization of the one hot encoding procedure.

annotated with one or multiple classes. In a binary segmentation task like a simplistic cancer segmentation (normal/cancer), a neural network model requires a quite simple mask by using a single variable with two states, zero and one. As output, a single probability score is predicted for each pixel ranging from 0 (normal) to 1 (cancer). However, a multi-class segmentation model (e.g. normal/lung/cancer) requires a binary mask for each class as well as outputs a probability score for each class. Therefore, a pixel in a multi-class mask requires the same amount of binary channels as the number of classes. This scenario gets even more complicated in a multi-label segmentation task, in which a single pixel can be assigned with not only a single class but also multiple at once. These scenarios and proceedings are analogical in image classification. In order to solve the multi-class issue, it is necessary to apply the one hot encoding technique to create binary dummy variables for each class.

3.3.2 Normalization

The normalization process of images describes a technique that changes the range of pixel intensity values [168, 177]. Especially in biomedical imaging, pixel intensity value ranges are highly inconsistent and vary due to different hardware (scanners), technical discrepancies, biomedical variation (skin color), or image formats (compression) [10]. Inconsistent pixel intensity ranges in an imaging dataset can have a drastic impact on the model learning ability and, therefore, classification as well as segmentation performance [146, 161, 177]. Furthermore, machine learning algorithms like deep convolutional neural networks usually perform better on normally distributed feature vectors [193]. In order to achieve pixel intensity range consistency, there are two popular normalization techniques: The MinMax normalization which scales the original value range to a predefined range usually between 0 and 1, whereas

the Z-Score normalization standardizes the values to a normal distribution by computing their zero-mean and variance [193].

$$\text{MinMax: } x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (3.1)$$

$$\text{Z-Score: } x' = \frac{x - \bar{x}}{\sigma} \quad (3.2)$$

Besides the normalization method, medical images can have multiple feature vectors according to their modality and representation like grayscale or RGB. Thus, it has to be noted that normalization should be applied to each feature vector channel individually [118]. Furthermore, image normalization can be applied to each image individually or computed on the basis of the complete dataset. Specifically, the Z-Score normalization can drastically vary between the individual image computation and the dataset-wide computation. Whereas individual image normalization highlights the focus of each image by allowing intensity value variance, uniform dataset normalization highlights value consistency between images.

3.3.3 Image Filtering

The field of image filtering, also known as image processing, is a collection of methods with the general aim of image enhancement [10, 146, 177, 194]. Image filtering methods try to counter or remove undesirable characteristics of images like blur, noise, poor contrast, or variation in intensity and illumination [10]. Particularly non deep learning based biomedical analysis approaches like thresholding, clustering, or edge detection are based on image filter utilization [10]. Popular image filtering techniques are smoothing linear and non-linear filters like Gaussian and Median filters, sharpening filters like Laplacian and Gradient filters, or histogram-based filters like Histogram Equalization [10, 146, 168, 169, 177]. However, image filtering as preprocessing method does not find use in the majority of modern deep convolutional neural network pipelines [118]. Interestingly, image filtering approaches are beforehand applied in many publicly available datasets for overall image quality enhancement, variance bias reduction, and task simplification [129]. Because the majority of medical image analysis research is built on these public datasets, this introduces a possible bias in these pipelines, which has to be noted when applying a classification or segmentation pipeline on clinical raw data. In general, the utilization of image filtering has to be under careful balance between enhancement or simplification of contextual information and possible information loss.

3.3.4 Bias Correction in Medical Imaging

Aside from image filtering techniques for standard imaging, often medical modalities require specific filters for medical images. Theoretically, medical imaging like MRI, CT, X-ray, or

OCT should have constant intensity values for the same tissue without big variance in the location of the tissue in the image [146, 183]. However, in reality, technical noise and artifacts distort the signals and result in a smooth variance in intensities [10, 146, 183]. This phenomenon is referred to as the bias field. Even if this issue is hard to notice by the human eye, it drastically impacts the learning ability of MIA models [146]. Thus, specific medical image filtering methods are required. There exist various kinds of approaches to solve this issue: Histogram based methods (Intensity Distribution based, Gradient Distribution based), Segmentation based methods (EM-based, Fuzzy C-Means based), Surface Fitting based methods (Intensity-based, Gradient-based), and Filtering based methods (Homomorphic Filtering, Homomorphic Unsharp Masking) [10, 146, 183, 194]. Nevertheless, in the scientific community, the most popular technique for MRI bias correction is the N4 Bias Field Correction via B-spline approximation by Tustison et al. [183].

3.3.5 Clipping

As already stated, ideally, medical imaging features constant intensity values for the same tissue. This feature can be exploited to simplify the ROI detection and segmentation task. For example, computed tomography scans using Hounsfield units as intensity values [10, 121, 146]. These have fixed and known value ranges for organs or other regions of interest, e.g. lymph nodes with a HU range of +10 to +20 [10, 54, 121, 146]. Clipping the intensity values outside of a defined minimum and maximum range close to the desired ROI range results in labeling every pixel as the background which is not close to this desired ROI range. This technique of focusing on only a specific value range associated with the ROI drastically reduces the model search space and increases the performance by avoiding false positives. However, cautious usage is highly recommended because clipping images to an incorrect minimum or maximum leads automatically to undetectable ROIs and a maladaptive model.

3.3.6 Padding

The method of padding an image describes the techniques for increasing a smaller image to a fixed larger shape [118]. In contrast to resizing which stretches an image to the desired shape, padding creates new artificial pixel intensity values along an axis until the image reaches the desired size. In deep convolutional neural networks, the convolutional and pooling layers of an architecture reduce the feature maps of an image which results in decreased image size by a factor of commonly 2 [108, 118]. This concludes that an image shape has to be dividable one or multiple times by a factor of 2 depending on the pooling depth of the used architecture. For ensuring this divisibility, smaller images can be padded to the minimum size to be successfully fed into such models. There are various techniques to pad an image like cloning the last axis row, mirroring, or using a specific value, the mean, median, minimum, or maximum of an image. Selecting the right padding technique has to be done carefully. The usual goal of padding is to synthetically expand the image background. Still, the risk of the unintentional creation of artificial features or an ROI has to be acknowledged and avoided.

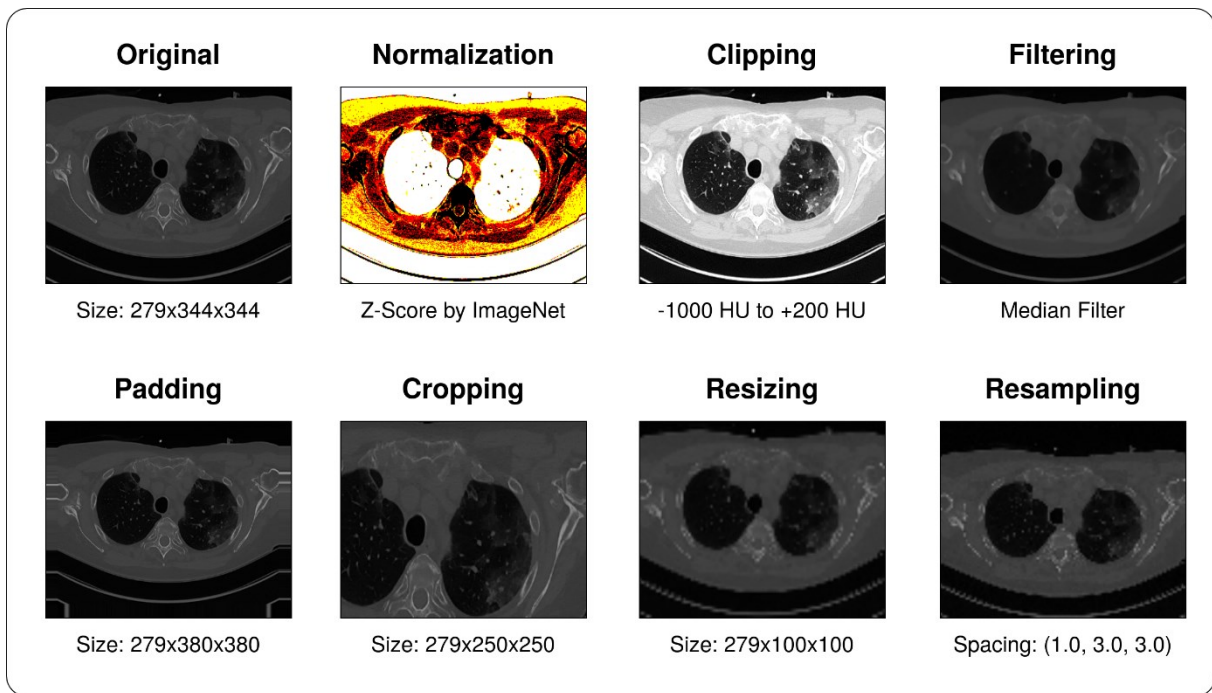


Figure 3.5: Application overview of multiple image preprocessing techniques.

3.3.7 Cropping

The removal of unwanted outer areas in an image is defined as cropping. In MIA pipelines, cropping is a popular technique to remove background or non-interesting regions [49, 118, 159, 177]. The idea behind cropping is to remove unnecessary information for task simplification. Excluding information from an image by cropping, is a semi-supervised preprocessing which requires prior knowledge of the task in order to differentiate between important and unimportant image areas. This technique automatically causes the model to focus more on the desired ROI via search space reduction. The benefits of cropping conclude with the avoidance of possible false positives and the model parameter number reduction which, thus, lowers the GPU memory requirements. In practice, cropping is performed either with fixed bounding box coordinates or with a binary mask specific to each image. For example, in lung cancer segmentation, cropping of the lung regions, which can be based on a previous segmentation model or an atlas registration, is commonly performed as preprocessing for the lung cancer segmentation model [191, 192, 195, 196].

3.3.8 Resizing

Similar to padding, the resizing technique increases the size of an image to a predefined value but instead by stretching or shrinking the image to its desired shape. This stretching and shrinking process can be computed via various interpolation algorithms like nearest-neighbor, bilinear, pixel area relation, bicubic and Lanczos interpolation [177, 197]. Next to normalization, resizing is one of the most popular preprocessing methods in image analysis and is integrated into the majority of imaging data augmentation frameworks [49, 140–142]. However, resizing an image always leads to an artificial bias and variance compared to the original data. Therefore, this technique reveals a double-edged dagger mentality between

resizing the image too small, which can result in information loss and blurriness, and resizing the image too big, which results in an unnecessarily large number of model parameters and excessive training time. Because medical images are usually in high resolution nowadays, the most common reason for resizing is down-scaling the image until it is possible to feed the image into the GPU [110, 197]. Overall in practice, up-scaling is quite rare, but ensuring a shape, which is dividable by factor 2 accordingly to the used model architecture, is also often solved via resizing. In direct comparison with padding, resizing interpolates the missing pixel whereas padding commonly fills the missing pixel with predefined dummy data (in practice with the background or minimum value). From the neural network model perspective, both methods are valid and commonly utilized. The model has to either learn from a stretched image or an image with larger background and a transformed ROI position.

3.3.9 Resampling

The resampling technique can be defined as a normalization attempt to obtain uniform pixel (voxel) sizes via resizing the images in a dataset. Regularly, magnetic resonance or computer tomography scans can have different pixel sizes which are called pixel spacing (for all axis) or also slice thickness (for the z-axis) [54, 146, 177]. A 3D model of a medical scan is computed by joining multiple 2D slices together [10, 54]. The z-axis size of a single 2D slice is represented through the slice thickness. Comparing two images, one with low slice thickness and one with high slice thickness but both with identical pixel resolution e.g. 512x512x200 pixels, the following conclusions can be drawn: The image with higher slice thickness is computed with fewer 2D slices and therefore has a worse spatial resolution or image quality. The image with lower slice thickness is computed with more 2D slices and therefore has a better spatial resolution or image quality. The pixel and slice thickness of a medical image, which value range is commonly presented in millimeters, results from the configuration of the individual medical scanner device [54]. However, inhomogeneous voxel spacings between images in a dataset are a challenging task for deep neural network models [118]. Therefore, volumes in an imaging dataset are resampled to homogeneous voxel spacing (also called target spacing) which drastically reduces the complexity of the task. This can be achieved by resizing each image according to the individual ratio between their current pixel spacing (a, b, c) and their desired target pixel spacing (a', b', c').

$$\text{Resampling: } x', y', z' = x \cdot \left(\frac{a}{a'}\right), y \cdot \left(\frac{b}{b'}\right), z \cdot \left(\frac{c}{c'}\right) \quad (3.3)$$

In conclusion, resampling directly affects the image sizes and the contextual information, which the neural network model is able to capture. This highly influences the required GPU memory of the model and has a huge impact on the final performance.

3.3.10 Patching

The resolution of a medical image can heavily determine the model's learning ability and performance [197]. Nevertheless, it is not possible to fully fit a complete high-resolution

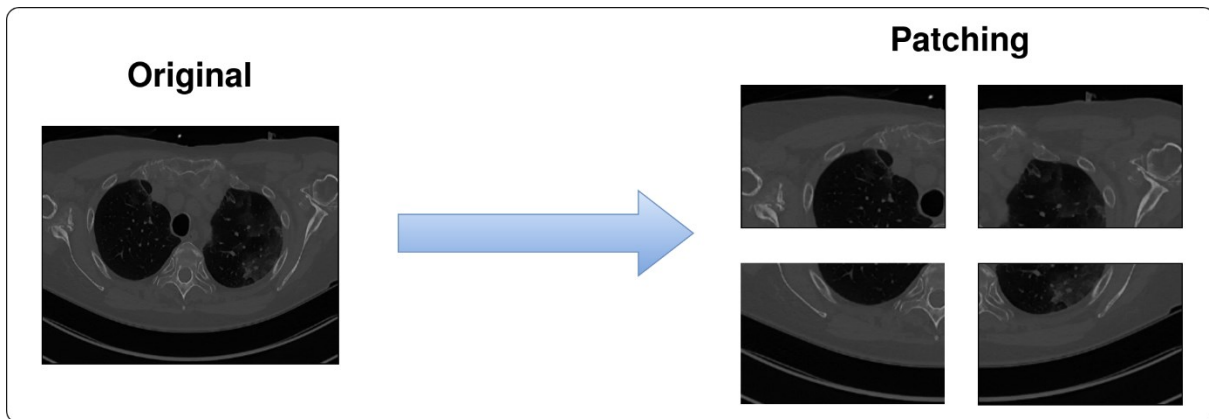


Figure 3.6: Illustration of a simple patching application on a CT scan.

medical 3D volume in state-of-the-art deep convolutional neural network models with current GPU hardware. An alternative to resizing or cropping methods in order to reduce the enormous GPU memory requirements due to the large data size is a patching approach. Patching describes the technique of slicing a large volume into multiple smaller commonly cuboid patches [49, 118, 146, 175, 198]. These smaller patches are able to fit and be processed in the neural network model. This allows the processing of high-resolution images without information loss like through resizing. Thus, patching became popular in the scientific community and widely utilized in recent MIA pipelines [27, 118, 198]. Popular patch sizes are 64^3 and 128^3 pixel cuboids or $160 \times 160 \times 80$ pixel patches depending on the available GPU hardware [49, 118]. For model training, there are two common approaches for patching: Fitting on all patches from a volume or fitting only on a single randomly selected patch (similar to a random cropping approach) from the volume [49, 109, 175]. This random selection of a single patch/crop technique for each epoch introduces another layer of data augmentation and, additionally, decreases the risk of overfitting. For inference, all sliced patches are processed by the model and, afterwards, ensembled back to the original volume shape. Recent pipelines also introduced an overlap between patches, because predictions for pixel close to the patch center seemed more reliable than predictions for pixel at the edge of a patch [49, 109, 175]. By overlapping patches, multiple predictions for a single pixel are computed and can be combined to a single prediction by simply averaging or weighting according to their position in the patch.

3.4 Neural Network Model

After preprocessing and formatting the data into suitable chunks, the images can be passed into a machine learning model. Neural networks are one of the most popular artificial intelligence or machine learning algorithm, which are used to automatically solve complex problems like computer vision tasks [13]. This chapter assumes that a model is a deep neural network in a medical image analysis pipeline and describes important hyperparameters, usage techniques as well as related topics for application. The theory and further information on the principles of a deep learning based neural network model can be found in Chapter 2.2.

3.4.1 Hyperparameters

A hyperparameter is defined as a parameter for a machine learning model which has to be selected prior to training start [16, 199]. In practice, a model should ideally approximate a similar predictor providing the same data and architecture. In recent years, neural network models get more similar as well as standardized in terms of application [200]. Thus, next to the data itself, the hyperparameters of a neural network model are one of the most significant parts of an MIA pipeline. An overview of the hyperparameters which will be discussed in the next sections is illustrated in Figure 3.7.

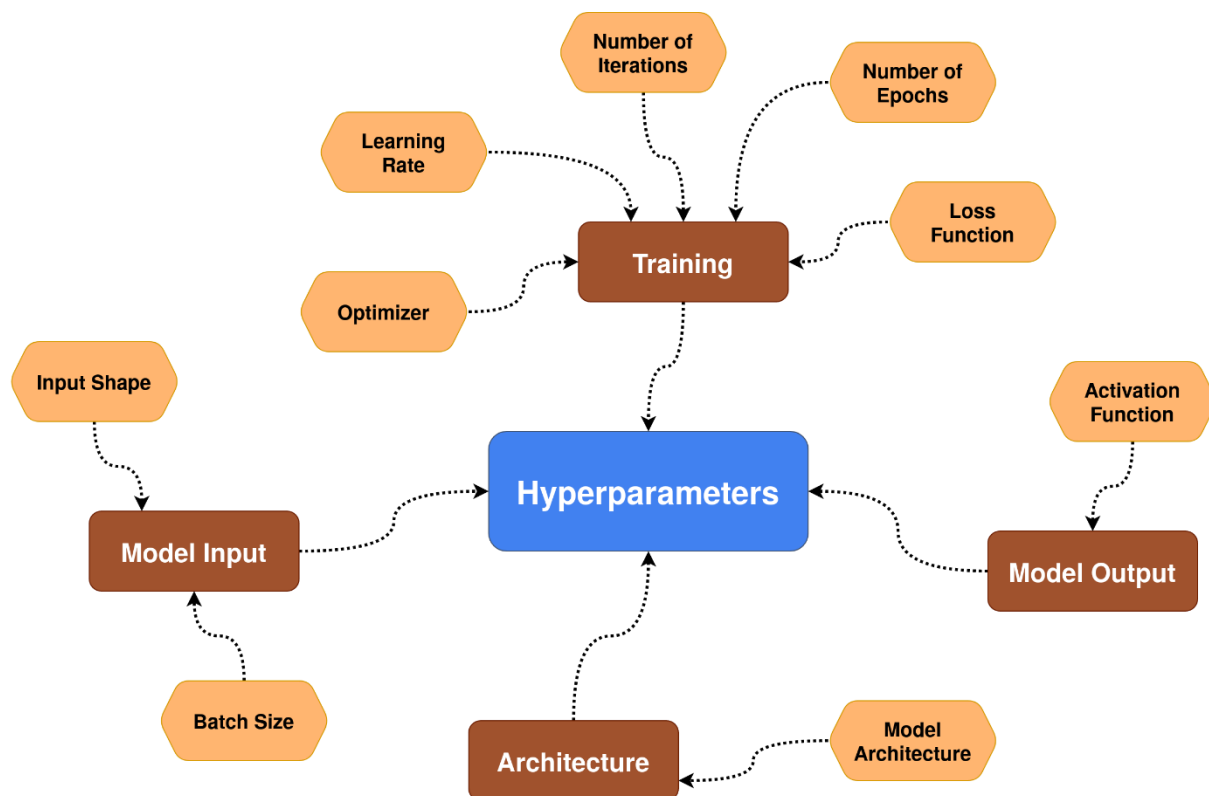


Figure 3.7: Overview of important hyperparameters in a MIA pipeline.

Training related Hyperparameters

For the training process of a neural network model, various hyperparameters are essential. The training process of a model is substantially framed by its learning rate, optimizer, and loss

function [67]. Whereas the learning rate is responsible for how much the neural network weights are adjusted in response to the loss function (see Chapter 2.2 for more information about neural network theory) and is thereby one of the most important hyperparameters for training success, the optimizer refines this process of weight updating. A too high learning rate yields more drastic weight updates which can result in ‘jumping’ across the global performance optimum, while a too low learning rate yields smaller weight updates which can result in being stalled at a local optimum. The usage of optimizers like SGD (Stochastic gradient descent) or Adam (adaptive moment estimation) [201] with a learning rate of $1E^{-3}$ to $1E^{-6}$ are popular in modern MIA pipelines [67, 118, 202]. The provided loss function of a model is utilized for error estimation in order to decide on necessary weight adjustments. Further details on loss functions can be found in Chapter 3.5.2.

Another important hyperparameter is the training time itself. In a traditional neural network training process, the training time is pre-defined as the number of ‘epochs’. An epoch is defined as the complete processing of all samples in the dataset [16, 199]. An epoch can further be divided into ‘iterations’ which is defined as the processing of a single batch [16, 199]. Thus, the number of iterations of an epoch is determined by the number of samples and the batch size, which is shown in Equation (3.4).

$$n_{iterations} = \frac{|samples|}{batch_size} \quad (3.4)$$

However, various modern MIA pipelines utilize a fixed iteration number per epoch instead of the traditional epoch definition [118]. By randomly drawing only a fixed number of batches during an epoch, it is possible to allow improved model convergence, especially for limited-sized datasets which are common in medical imaging. A rule of thumb recommendation would be to ensure an epoch to have at least 150 iterations for a well-converging training process. However, this recommendation requires further research and experimentation.

Model Input

After preprocessing the imaging data, it is required to pack image samples together in a so-called ‘batch’. A batch is a data unit that can be processed by the neural network model and consists of one or multiple samples. The number of samples in a batch is defined as ‘batch size’ and determines how many samples are processed by the neural network model in a single step. The batch size can affect multiple model capabilities. Kandel et al. [203], Radiuk [204], and Golmant et al. [205] concluded in their experiments that the batch size has a significant impact on model performance and effectiveness in optimization steps for convergence. Especially with a high learning rate, large batch sizes yield strong performance results compared to models with smaller learning rates [203]. However, Golmant et al. [205] stated that the impact on the performance with an increasing batch size is highly dependent on the neural network architecture and computer vision task, as well as only possible to a certain point. Due to hardware requirements (GPU VRAM), high-resolution images or even 3D images, which are

common in medical imaging, drastically limit the maximum batch size for state-of-the-art neural network architectures. Thus, batch sizes of 16 to 64 2D images and of 2 to 4 3D images are common in modern MIA pipelines [118, 203, 206].

Another hardware limited and maximum batch size determining hyperparameter is the input shape of the images. During the preprocessing, an image has to be resized, resampled, cropped, or padded to a uniform input shape for all samples. These fixed input shapes for an image are often determined by the utilized neural network architecture of the MIA pipeline [74]. In the medical context with a prevalence of high-resolution and 3D imaging, this often leads to the need for image size reduction in order to fulfill the fixed input shapes predetermined by the architecture or hardware limitations [118].

Model Output

The output of a neural network model is always defined by the applied activation function at the end of the architecture. Whereas the majority of neural network nodes utilize ridge activation functions like linear activation or ReLU [73], the output activation function is determined by the classification task [75]. Binary as well as multi-label classification and segmentation tasks require a ‘sigmoid’ activation function as output while multi-class classification and segmentation tasks require a ‘softmax’ activation function. More about activation function theory can be found in Chapter 2.2.1.

Architecture

The structure of a neural network model is defined by its architecture and is the most important choice in MIA pipeline building. The architecture represents the network design by neurons and their edges to each other. More about neural network architecture theory can be found in Chapter 2.2 and Chapter 2.3. Due to the standardization and wide usage of established architectures in image classification and segmentation [13, 74, 75], custom implementation or designing of novel architectures tends to become a separate research field. For application, it is common to utilize already implemented architectures like the U-Net for segmentation or ResNet for classification [13, 74, 75]. Thereby, the architecture of a neural network model can be defined more like a hyperparameter in modern MIA pipelines. Selecting a suited architecture is highly task-dependent. Important decision factors are the available hardware and intended application. Large and complex architectures like EfficientNet [79] require modern high-performant GPUs like an NVIDIA TITAN RTX, whereas smaller architectures like MobileNet [91] are designed to run on mobile devices like smartphones or tablets.

3.4.2 Training and Prediction Process

After the setup of a neural network model including data loading, image augmentation, and preprocessing, it is possible to start the training process of the model. In a training process, the neural network model is fitted by minimization of its error (loss) function resulting in an increase in its predictive capabilities. In supervised learning, medical images as well as

annotations are required for the training process in which the model repeatedly attempts to predict the annotation of the images and evaluates its performance. After a predefined number of epochs, the training process stops and yields a fitted model. Next to the internal model evaluation for the training based on a loss function, it is also possible to provide the model with additional validation data for which metric scores can be computed at the end of each epoch. This validation allows not only a finer interpretation of the training process quality for hyperparameter tuning but also the application of advanced validation monitoring techniques which are explained in detail in Chapter 3.6.2.

A fitted neural network model can be used for predicting an annotation of unknown and new images. These images have to be preprocessed identically as the images for training in order for the model to make reasonable inferences. The prediction process can be categorized into two usage cases: Either by predicting large batches from a testing set for performance assessment in terms of an experiment, or by predicting a single sample in terms of a service point for an application. The prediction for a testing set is a key part of any MIA pipeline to robustly evaluate model performance and predictive capabilities. A finalized MIA pipeline can be utilized in a production environment in which new samples can be passed to and automatically processed by the MIA pipeline resulting in a predicted classification or segmentation. The range of applications highly varies from deployment in a mobile App to integration in a clinical workflow.

Transfer Learning

The repeating of the training process of a neural network model is defined as transfer learning [11, 74, 207]. The idea of transfer learning is to utilize information from another dataset, which can be related as well as unrelated to the actual task, in order to compensate for insufficient training data or reduce generalization complexity [34]. In deep learning based computer vision, there are three types of transfer learning:

- Prior model training from scratch on another dataset
- Utilizing weights from a pre-fitted model and starting a usual training process
- Utilizing weights from a pre-fitted model and starting a transfer learning training process

The most intuitive transfer learning technique is the beforehand training from scratch in which a prior training process is applied with a separate dataset. Often the separate dataset is a similar task on the same medical imaging modality [208]. Instead of individually training the neural network model on a separate dataset, it is time- as well as hardware-saving (if no access is available to large GPU clusters) to simply reuse the weights of an equivalent neural network model with comparable preprocessing. Afterward, the training process on the target data is applied as usual. Ker et al. [34] defined this type of transfer learning as ‘deep-tuning’. The last type of transfer learning is to optimize the training process based on reused weights from a pre-fitted model. This type of transfer learning can be divided into the ‘shallow-tuning’ and ‘fine-tuning’ phases [34]. For the shallow-tuning phase, the neural network model starts an initial training process based on weights from a pre-fitted model. For this initial training process, all

layers except for the classifier are frozen, a high learning rate is selected (for example $1E^{-04}$), and the model is fitting for a small number of epochs (commonly 5-15 epochs) [11, 207, 209]. The concept of shallow-tuning is that the model classifier can adapt the fixed architecture weights to the task. After this initial adaption phase, the architecture weights get unfrozen and the second training process is started with regular hyperparameters but a smaller learning rate than for the shallow-tuning phase (for example $1E^{-05}$) [11, 207, 209]. In this phase, the complete neural network model fine-tunes all weights for the task to obtain optimal performance.

Compared to general computer vision, medical image analysis lacks large annotated datasets due to the increased required expertise for the annotation process [34]. Thus, MIA pipelines commonly apply transfer learning strategies with unrelated data to medical imaging in order to overcome this obstacle [34]. The significant performance benefit of the usage of transfer learning has been proved by several authors like Krizhevsky et al. [81] in one of the key publications of computer vision, Shin et al. for interstitial lung disease classification in CT scans [207], Ravishankar et al. [210] for kidney localization in ultrasound imaging, and Tajbakhsh et al. [211] in a specific study about transfer learning effectiveness in multiple medical imaging modalities.

Even though transfer learning is heavily utilized in MIA pipelines, there is a large difference in transfer learning type preference in the MIA subfields. In medical image classification, the shallow- and fine-tuning transfer learning strategy is heavily utilized as well as it is the accepted gold-standard in the research field [210]. The pre-fitted weights are universally based on the ImageNet dataset which consists of non-medical-related 15 million labeled high-resolution images categorized in over 22,000 classes [76, 210]. However, medical image segmentation heavily relies on prior model training from scratch on a similar dataset [208]. Even so, the field of image segmentation is widely based on the U-Net architecture in recent years, the main reason for the preferred transfer learning type is that there is no large annotated image segmentation dataset in computer vision. Thus, transfer learning strategies in MIS pipelines are often neglected in favor of other performance improvement strategies due to the time-consuming process of prior model training from scratch. Nevertheless, the transfer learning strategy of utilizing weights from pre-fitted models is a popular technique in general MIA for initializing weights in novel architectures like 3D variants of popular 2D architectures [34].

Fitting Process Monitoring

Apart from the performance assessment, evaluation of the training process is crucial for robust model building. In order to identify optimization potential and overfitting presence, the effectiveness of the model fitting has to be validated. A powerful but simple method for fitting effectiveness analysis is the visualization of the ‘fitting curve’. During the training process, the loss function score is logged after each epoch for the training and a validation set. The fitting curve is a line plot comparing the loss function (or additional metrics) progression against the epochs and is showing the curve for the training as well as validation. In terms of training effectiveness, the fitting curve can present multiple indicators. An overall non-decreasing or

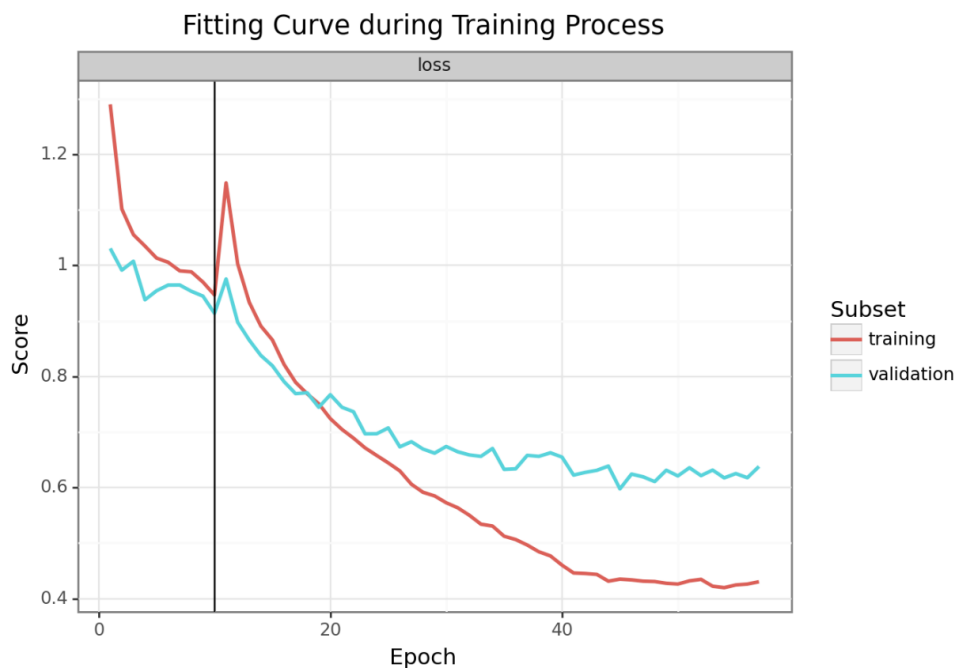


Figure 3.8: Fitting curve visualizing of a typical loss function trend during a training process.

wave-looking curve indicates that the model is not able to learn any features for generalization and yields a model without any or only low predictive capabilities [212]. The interpretation of a plateau in a fitting curve depends on its position. A plateau at the beginning of the training process reveals no learning effect of the model and is an indicator of various possible issues in the pipeline ranging from a too small learning rate to faulty preprocessing techniques. In contrast, a plateau at the end of the fitting curve with a prior loss decrease is expected and shows successful learning of feature generalization until an optimum in predictive capability is achieved [212]. Whether the optimum is a local or global loss minimum is evaluated in the performance assessment. For overfitting presence analysis, the distance of the fitting curves between the training and validation set has to be evaluated. An increasing discrepancy between the training and validation loss reveals that the model gradually overfits on the training data whereas its predictive capabilities decrease on the unseen data from the validation set [212]. Usually, the discrepancy is small at the beginning and grows with an increase in epochs [212]. When reaching a learning plateau, an increased discrepancy between training and validation loss can be typically observed [212].

A fitting curve example is illustrated in Figure 3.8 which shows a transfer learning enhanced (indicated by the horizontal black line at epoch 10) training process for a skin lesion classifier based on the International Skin Imaging Collaboration dataset [57–59] and fitted with a class weighted focal loss function [213].

3.4.3 Frameworks

In computer science, a software framework is a construct, platform, or API which provides generic functionality for the development or deployment of specific software applications. As key feature, a framework allows universal and standardized development. The building process

of an MIA pipeline involves expertise in various scientific fields: Low-level hardware communication for complex computation, machine learning, deep learning based neural networks, image processing, and medical imaging [13]. For building an MIA pipeline from scratch, it would be necessary to be an expert in all mentioned fields which would result in an excessively time-consuming developing process. For this reason, MIA pipelines are created with different types of frameworks that allow an effective building and application process. There are three types of frameworks that are relevant for building MIA pipelines: Base frameworks, toolkits, and frameworks for automated machine learning (AutoML).

Table 3.3: Overview of available medical image analysis frameworks for research.

Software Name	Type	Features	Publication	Popularity	Activity
Tensorflow [72]	Base FW	Static Graph-Creation, Industry-focus	2015	167,474	2022
Pytorch [71]	Base FW	Dynamic Graph-Creation, Research-focus	2016	58,384	2022
JAX [214]	Base FW	High-performance ML research	2018	20,048	2022
NiftyNet [43]	Toolkit	Segmentation & Classification	2017	1,314	2020
MONAI [175]	Toolkit	Segmentation & Classification	2020	3,303	2022
MedicalTorch [215]	Toolkit	Segmentation	2018	788	2019
EISEN [216]	Toolkit	Segmentation	2020	42	2020
DeepNeuro [215]	Toolkit	Segmentation	2018	111	2020
nnU-Net [109]	AutoML	Segmentation	2019	2,889	2022
nnDetection [217]	AutoML	Object Detection	2021	324	2022
Nobrainer [218]	AutoML	Segmentation & Classification	2020	118	2022
ivadomed [218]	AutoML	Segmentation, Classification & Object Detection	2020	139	2022
MIScnn (own) [49]	Toolkit	Segmentation	2019	325	2022
AUCMEDI (own) [98]	Toolkit & AutoML	Classification	2022	10	2022

In order to give an overview of frameworks for deep learning based MIA, relevant software packages are summarized and presented in Table 3.3. The frameworks are categorized according to their framework type, features or computer vision task, publication date (official release year if available or manuscript publication), popularity measured by GitHub [219] stars of the official Git repository, and activity measured by the date of the last commit in the associated Git repository (measurement date: 01.09.2022). The activity coloring also indicates if the project is still maintained (green) or deprecated (orange).

Base Frameworks

According to Litjens et al. [13], one of the main contributions to the rise of deep learning based solutions in MIA are the GPU computing libraries as well as the open-source software packages for building deep learning models. Well-maintained GPU computing libraries like CUDA [220] or OpenCL [221] allow utilizing GPUs for high-performance computing with complex neural network models. Accordingly, base frameworks, which are based on GPU computing libraries,

are essential for building modern neural network models and are widely popular in the fields of general deep learning based machine learning, computer vision, and MIA. Base frameworks (Base FM) provide efficient implementations for robust and rapid building of neural network models. These implementations include model management in terms of weight data structure as well as model storage, functions for training as well as prediction processes, and a library of neural network layers for architecture building like convolutions, pooling, or normalization. Furthermore, Base FMs include implementations for popular optimizer and activation functions. These high-level frameworks allow researchers to focus on experimentation and application rather than worrying about the efficient implementation of neural network foundations [13].

Toolkits

For building MIA pipelines, various methods for processing medical images have to be implemented. However, instead of re-implementing the same processing methods or similar MIA pipelines each time, researchers have started to establish community projects providing efficient implementations of popular methods and essential parts of MIA pipelines. These community projects can be defined as ‘toolkits’ which are often high-level APIs and built on top of Base FMs. Toolkits commonly consist of image loading functions for medical formats like DICOM or NIfTI, libraries for preprocessing methods, and neural network architectures for computer vision [43, 175, 222, 223]. In recent years, the usage of toolkits for building MIA pipelines drastically increased [200, 224]. This can be explained due to the higher availability of such toolkits as well as the growing interest in the application of deep learning based computer vision models outside of the computer science research field. As toolkits offer the possibility to quickly setup a functional MIA pipeline without time-consuming foundation implementations, deployments and applications in clinical environments became easier to manage and, thus, more popular.

Automated Machine Learning

Even though, toolkits provide powerful capabilities for efficient MIA pipeline building, the development and application process for such pipelines still needs deep learning experts to setup robust and high-performing pipelines. Nevertheless, the majority of modern MIA pipelines share a high similarity with only marginal differences depending on the task and imaging modality. For the purpose of usability and simplification, automated machine learning frameworks provide a high degree of automation for the deployment and application of modern MIA pipelines [98, 109, 217, 223]. An AutoML framework manages all relevant steps from loading medical imaging data to the neural network model. Moreover, adequate preprocessing and hyperparameters are often automatically selected or adjusted according to the input data [109, 217]. This allows also non-experts in the field of deep learning to build, apply as well as maintain deep learning based MIA pipelines.

3.5 Performance Assessment

Evaluation of the model performance and predictive power is an essential step in every machine learning pipeline. In medical image analysis with deep convolutional neural networks, there are two types of performance scores. The evaluation metrics, which are used for external model scoring after the inference as well as follow in general the principle ‘if higher than better’, and the loss functions, which are used for internal model scoring during the training process and follow the principle of ‘if smaller than better’.

3.5.1 Evaluation Metrics

Evaluation of MIA can be quite complex because it is required to measure classification accuracy as well as additional localization correctness for segmentation tasks. Over the last 30 years, a large variety of evaluation metrics can be found in the MIA literature [225]. However, only a handful of scores have proven to be appropriate and are used in a standardized way [225, 226].

In the following subchapters, each metric will be defined and discussed in terms of possible issues. Nearly all presented metrics, except Hausdorff distance, are based on the computation of a confusion matrix for a binary classification task, which contains the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions. Except for Cohen’s Kappa and Hausdorff distance, the value ranges of all presented metrics span from zero (worst) to one (best).

F-measure based Metrics

F-measure, also called F-score, based metrics are one of the most widespread scores for performance measuring in computer vision as well as in the MIA scientific field [105, 125, 225, 227, 228]. It is calculated from the Sensitivity and Precision of a prediction, by which it focuses on scoring the overlap between prediction and ground truth. Still, by including the precision, it also penalizes false positives, which is a common factor in highly class-imbalanced datasets like MIA [225, 227, 228]. Based on the F-measure, there are two popular utilized metrics in MIA: The Intersection-over-Union (IoU), also known as the Jaccard index or Jaccard similarity coefficient, and the Dice Similarity Coefficient (DSC), also known as F1-score or Sørensen-Dice index (in the MIC field mostly called F1-score and in MIS field DSC).

$$IoU = \frac{TP}{TP + FP + FN} \quad (3.5)$$

$$DSC = \frac{2TP}{2TP + FP + FN} \quad (3.6)$$

Besides that, the DSC is defined as the harmonic mean between Sensitivity and Precision, the difference between the two metrics is that the IoU penalizes under- and over-classification more than the DSC. Even so, both scores are appropriate metrics, the DSC is the most used metric in the large majority of scientific publications for MIS evaluation [125, 225, 227].

Sensitivity and Specificity

Especially in medicine, Specificity and Sensitivity are established standard metrics for performance evaluation [225, 227]. For classification, the Sensitivity (Sens), also known as Recall or True Positive Rate, focuses on the true positive detection capabilities, whereas the Specificity (Spec), also known as True Negative Rate, evaluates the capabilities for correctly identifying true negative classes. In MIA evaluation, Sensitivity is a valid and popular metric, but still less sensitive to F-score based metrics for exact evaluation and comparison of methods [225, 227]. However, the Specificity can result in an improper evaluation metric if not correctly understood. In MIA tasks, the Specificity often indicates the model's capability to detect the non-ROI or control class in an image. In segmentation, due to the large fraction of pixels annotated as background compared to the ROI, Specificity ranges close to 1 are standard and expected. Thus, Specificity is a suited metric for ensuring model functionality, but less for model performance estimation.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (3.7)$$

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (3.8)$$

Accuracy

Accuracy (Acc), also known as Rand index or pixel accuracy, is one or even the most known evaluation metric in statistics [225, 228]. It is defined as the number of correct predictions, consisting of correct positive and negative predictions, compared to the total number of predictions. However, it is strongly discouraged to use Accuracy due to the strong class imbalance in MIA. Because of the true negative inclusion, the Accuracy metric will always result in an illegitimate high scoring. Especially in image segmentation, predicting the mask of an entire image as background class, Accuracy scores are often higher than 90% or even close to 100%. Therefore, the misleading Accuracy metric is not suited for MIA evaluation and using it is highly discouraged in scientific evaluations. This issue is discussed in detail in Chapter 7.2.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \quad (3.9)$$

Receiver Operating Characteristic

The ROC curve, short for Receiver Operating Characteristic, is a line plot of the diagnostic ability of a classifier by visualizing its performance with different discrimination thresholds [225, 229]. The performance is shown through the True Positive Rate (TPR) against the False Positive Rate (FPR). In particular, ROC curves are widely established as a standard metric for comparing multiple classifiers and in the medical field for evaluating diagnostic tests as well as clinical trials [230]. As a single-value performance metric, the area under the ROC curve (AUC) was first introduced by Hanley and McNeil in 1982 for diagnostic radiology [231]. Nowadays, the AUC metric is a common method for the validation of machine learning classifiers. It has to be noted that an AUC value of 0.5 can be interpreted as a random classifier. The following AUC formula is defined as the area of the trapezoid according to David Powers [47]:

$$AUC = 1 - \frac{1}{2} \left(\frac{FP}{FP + TN} + \frac{FN}{FN + TP} \right) \quad (3.10)$$

Cohen's Kappa

The metric Cohen's Kappa (Kap), introduced by Cohen in 1960 in the field of psychology, is a change-corrected measure of agreement between annotated and predicted classifications [225, 232, 233]. For interpretation, Kap measures the agreement caused by chance like the AUC score and ranges from -1 (worst) to +1 (best), whereas a Kap of 0 indicates a random classifier. Through its capability of application on imbalanced datasets, it has gained popularity in the field of machine learning [233]. However, a recent study demonstrated that it still correlates strongly to higher values on balanced datasets [233, 234]. Additionally, it does not allow comparability on different sampled datasets or interpretation of prediction accuracy.

$$f_c = \frac{(TN + FN)(TN + FP) + (FP + TP)(FN + TP)}{TP + TN + FN + FP} \quad (3.11)$$

$$Kap = \frac{(TP + TN) - f_c}{(TP + TN + FN + FP) - f_c} \quad (3.12)$$

Average Hausdorff Distance

In contrast to other confusion matrix-based metrics, the Hausdorff Distance (HD) is a spatial distance-based metric that can be utilized for MIS evaluation [225]. The HD measures the distance between two sets of points, like ground truth and predicted segmentation, and allows scoring localization similarity by focusing on boundary delineation (contour) [225, 235, 236]. Especially in more complex and granular segmentation tasks, exact contour prediction is highly important which is why HD based evaluations have become popular in the field of MIS [225].

However, because the HD is sensitive to outliers, the symmetric Average Hausdorff Distance (AHD) is utilized in the majority of applications instead [225, 235, 237]. The symmetric AHD is defined by the maximum between the directed average Hausdorff Distance $d(A,B)$ and its reverse direction $d(B,A)$ in which A and B represent the ground truth and predicted segmentation, respectively, and $\|a-b\|$ represents a distance function like Euclidean distance [225]:

$$d(A, B) = \frac{1}{N} \sum_{a \in A} \min_{b \in B} \|a - b\| \quad (3.13)$$

$$AHD(A, B) = \max(d(A, B), d(B, A)) \quad (3.14)$$

Other Metrics

In the field of MIA, various other metrics exist and can be applied depending on the research question and interpretation focus of the study. For further details on the theory of previously presented metrics, the Author refers to the excellent studies for general classification evaluation by Lever et al. [228] and for MIS evaluation by Taha et al. [225]. Additionally, Nai et al. provided a high-quality demonstration of various metrics on a prostate MRI dataset [237].

3.5.2 Loss Functions

The fitting process of a neural network model is an optimization problem in which the adjustment of model weights has to be scored for each iteration. The optimization aim of the training process is to find the model weight combination which results in the best scoring. By running an internal prediction on the training data with the adjusted weights, the current performance can be evaluated and scored. In contrast to regular evaluation metrics, neural networks require a decreasing metric in which high performance results in a low value whereas low performance in a high value. Such kind of metric for optimization minimization is defined as a loss function [238–240].

F-measure based Loss

An intuitive approach for choosing a loss function is to utilize a favored evaluation metric. For this reason, F-score based loss functions like the DSC are also highly popular in MIA [240]. In order to obtain a loss function, it is required to create the antipode of the desired metric. This can be simply achieved by subtracting the metric from the value 1, which is how the majority of DSC loss functions are implemented [239, 240].

$$DSC_{Loss} = 1 - \frac{2TP}{2TP + FP + FN} \quad (3.15)$$

Another popular implementation of F-score based loss functions is the Tversky index, also called Tversky loss [241].

$$Tversky = 1 - \frac{TP}{TP + \alpha \cdot FN + \beta \cdot FP} \quad (3.16)$$

Here, α and β are defined as free parameters. The Tversky loss acts as a generalization for all F-score based loss functions [239–241]. For example, if defining α and β as 1, the resulting Tversky loss will be equal to the IoU. If defining α and β as 0.5, the loss will be equal to the DSC. Furthermore, the parameters can be used for custom weighting of the false negative and false positive impact on the score.

Cross-Entropy

For training deep convolutional neural networks, cross-entropy (CE) is the most popular loss function in general computer vision [238–240, 242, 243]. Thus, it is also widely used in MIA with performance commensurate to F-score based loss functions [239, 240, 242]. In general, the cross-entropy measures the difference between two probability distributions for a random variable (a sample) and events (class labels). The predicted class probability is compared to the ground truth and a score is computed based on penalizing how far it is from the actual class.

$$CE(p_t) = -\log(p_t) \quad (3.17)$$

In this formula, p_t is defined as the model's predicted correct class probability for the observation (sample) t . The loss minimization is achieved by the negativity of the entropy estimation.

Focal Loss

Originating from the object detection field in computer vision, the focal loss has become more and more popular in MIA [239, 240]. Lin et al. introduced the focal loss as a reshape or enhancement of the standard cross-entropy loss function, in which well-classified examples are down-weighted for the scoring [213]. Especially in MIS, the focal loss is particularly useful for extreme foreground-background class imbalance or an ROI consisting of only a few pixels in an image. This challenge is widely present in all kinds of medical imaging which is why the focal loss showed significant effectiveness for training MIA models [213, 239, 240].

$$Focal(p_t) = -(1 - p_t)^\gamma \log(p_t) \quad (3.18)$$

In this formula, p_t is defined as the model's predicted correct class probability for the observation (sample) t and γ is defined as a tunable focusing parameter. The focusing parameter determines the extent of down-weighting easy samples. Adjusting the γ to 0, results in the

standard cross-entropy loss function, whereas adjusting γ to 2 showed high-performing model training capabilities on highly imbalanced data [213].

Class Weighting

Weighting evaluation metrics and especially loss functions for model training became a standardized method for state-of-the-art MIA pipelines [244]. Next to specifically designed loss functions with tunable focusing parameters like the focal loss, it is often possible to multiply a loss for a single sample with an associated weight based on its class frequency. Commonly, this approach is applied to the cross-entropy loss function in which the associated class weights are prior calculated through the total number of samples divided by the multiplication of the number of classes and the occurrences of each class [239, 240].

3.6 Sampling and Validation Strategies

In statistical application and machine learning, models are evaluated by an untouched dataset separated from the training set which is used for model building [199, 245, 246]. This type of dataset is also called a testing set or hold-out set. The testing set is commonly used to give an unbiased estimate of the fitted model performance and its prediction capabilities [199, 246]. This testing evaluation is usually performed after model training. However, for measuring model performance on unseen data for hyperparameter tuning and model selection, another subset separated from the training data is recommended. This set is called a validation set, is usually utilized during or at the end of the model training and supports the model-building process [245, 246].

The usage of a validation set reveals several advantages. In general, performance evaluation of a model on data, which was used for model fitting and is already ‘seen’ by the model, results always in a bias towards seemingly better performance than it actually is [199, 245, 246]. Such overfitting can lead to the model losing usability on its actual task resulting in low up to catastrophic performance on unseen data [245, 246]. The difference between the performance on training and unseen data or generalization capabilities of a model is called the generalization error [16]. The process of counteracting and minimizing the generalization error resulting from overfitting is defined as regularization in the mathematical and statistical field [16]. In general, any data which influences the model-building process is biased toward higher performance. For example, selecting model hyperparameters based on the best performance on the testing set still contains the risk that the rejected models are better for new unseen data [16, 199, 245]. This is why, it is highly recommended to hold-out an untouched testing set for evaluation, a separate set specifically for hyperparameter tuning or model selection, and any remaining data for the model training. In conclusion, utilizing validation and testing sets allow performing more robust and reliable evaluations by avoiding possible overfitting and reducing variance bias.

3.6.1 Sampling Strategies

When dividing a dataset into subsets, like training, validation, and testing, it is recommended to consider the class frequency and difficulty variance [245, 246]. For handling multi-class classification or segmentation tasks, it is required to ensure that all classes are present at least in the training dataset. It is a general rule of thumb that the overall class frequency across the dataset is properly represented in the resulting subsets, as well, which is called stratified sampling. Similar to class frequency, the task difficulty of images can be varying, which also should be considered and equaled across subsets.

Percentage Split

The minimal approach for performance evaluation of an MIA model on a dataset is the train/test split. By randomly selecting and excluding samples from the dataset, a testing set can be created. Whereas the model-building process uses the training data, the resulting model is

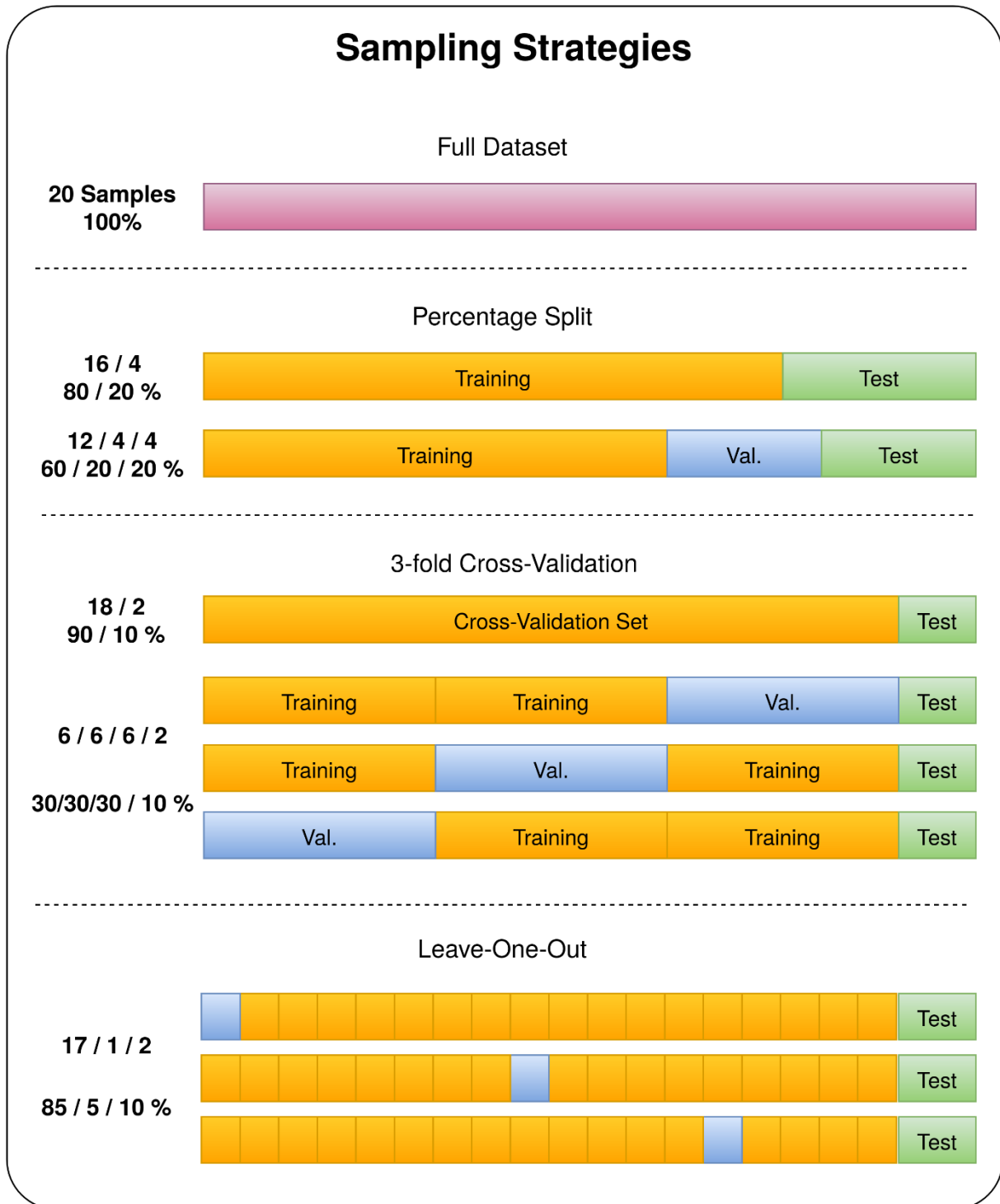


Figure 3.9: Illustration of sampling strategies.

evaluated on the testing set [199, 246]. Commonly, split ratios of 90-60% training and 10%-40% testing are favored [246]. However, the suited splitting ratio is highly dependent on the dataset and task.

The utilization of an additional validation set beside the testing set is the classical and most popular approach in machine learning as well as medical image analysis [199, 246]. The split ratios are often distributed in that the training dataset is the largest while the validation and testing set is smaller as well as equally sized [246].

K-fold Cross-Validation

In contrast to the classical percentage ratio splits like the train/val/test split, the k-fold cross-validation technique is more complex by multiple dataset partitioning. By splitting the dataset into k equally sized folds, it is possible to use each fold one time as a validation set and the remaining folds as training [199, 246]. Therefore, the cross-validation results in k models and is able to utilize the complete dataset for training as well as validation. Commonly, the cross-validation approach is applied to the training set after dividing the dataset via a train/test percentage split. Cross-validation is a powerful technique to significantly reduce the risk of overfitting and selection bias [110, 246]. A popular number of folds (k) in MIA are 3, 5, and 10 [157, 247–249]. These numbers result from finding a suitable balance between variance reduction and training time by increasing the number of folds.

Leave-One-Out Validation

The leave-one-out technique is a special subgroup of cross-validation, in which k equals the number of samples [199, 246]. Therefore, the validation set contains only a single sample whereas the remaining are used for training. Mostly, this approach is utilized during the developing process of a pipeline in order to obtain fast results at the start. Even if this approach is practical to validate pipeline functionality, it is highly inadvisable due to the high subset variance of a single sample and the extreme training time for computing N models.

3.6.2 Validation Monitoring

There are multiple advanced techniques for utilizing information from a validation set during the training process. These methods automatically adjust the corresponding model training hyperparameters based on regularly evaluating the model performance with current configurations. Commonly, this hyperparameter adjustment is based either on the loss or another metric computed for each epoch for the validation set.

Early Stopping

The Early Stopping technique has the goal to stop the training process at the moment when no more loss reduction or minimization is possible [250]. Usually, the model performance on the validation set is monitored every epoch. If a model validation loss is no longer decreasing for a predefined number of epochs, the training process will be terminated. Early Stopping not only helps to eliminate unnecessary training time but also reduces the risk of overfitting by stopping the training when there is no more performance gain on the validation set (even when the training loss would be still decreasing).

Dynamic Learning Rate

A fixed learning rate of a deep convolutional neural network often either slows down the fitting process by being too small or is not able to achieve precise weight adjustments by being too high [250]. The dynamic learning rate method reduces the learning rate by a factor of commonly

2 up to 10 if no improvement is monitored on the validation set for a predefined number of epochs. This allows utilizing a high learning rate at the beginning of the fitting process to find suited weight distributions, whereas also allows weight fine-tuning to achieve optimal performance.

Model Checkpoints

Even if the risk of overfitting can be minimized, there could be still generalization errors due to general variance between the training set and unseen data [16]. This concludes that a model, which is the best-resulting model on the training set, is not necessarily the best-generalized model from the fitting process. By saving the model after each epoch, all models can theoretically be utilized instead of just the final model with the lowest training loss. Commonly, the model with the lowest loss on the validation set is selected for further processing.

4

Medical Image Segmentation

Image segmentation has an essential role in medical image processing [10, 124, 125]. The aim of medical image segmentation (MIS) is the automated detection, labeling, and extraction of regions of interest. Thus, an MIS pipeline is able to divide medical images into segments that correspond to different tissue classes, organs, pathologies or abnormalities, and other biologically relevant structures [10, 124, 125]. These resulting segments can be utilized by clinicians or additional AI approaches to obtain easier or further diagnostic insights [10]. The automatic highlighting of abnormal features and ROIs allow image segmentation to aid radiologists in various tasks like diagnosis, treatment monitoring, and time-consuming inspection processes [10, 124]. Popular applications today are cell counting, organ or tumor measurements, lesion tracking, and sub-region identification like brain atlas [10, 124]. Nevertheless, more complex tasks like clinical decision support systems for abnormality detection as well as identification are currently highly popular research topics for clinical trials and get slowly integrated into the clinical workflow of modern hospitals [11, 62].

In this chapter, the Author proposes the framework MIScnn which is a software package for the effective as well as standardized setup of state-of-the-art convolutional neural network and deep learning models for MIS. Furthermore, three studies by the Author utilizing the MIScnn framework are presented. The studies not only demonstrate the wide applicability of MIScnn but also contribute with their outcomes to the field of MIS, pathology, radiology, as well as COVID-19 research.

4.1 History and Current State

Since about 1980, automatic image segmentation methods are available [251]. Throughout this time, a wide range of algorithms was proposed for segmentation in medical imaging. Nevertheless, because of the complexity of medical images, the application of classic approaches are suboptimal as well as limited on specific imaging modalities and task [10]. This subchapter provides a short overview of methods for MIS throughout the last decades and current challenges in the field.

4.1.1 Methods

Thresholding

The binary partitioning of an image based on the image intensity values is called thresholding and is one of the basic operations in standard image processing [10, 124]. Thresholding is performed by classifying each pixel based on whether the pixel intensity is inside a defined limit (minimum and maximum intensity value) or not. A thresholding approach is often simple but effective for imaging modalities in which medically relevant structures have different pixel intensity ranges like in CT imaging [124]. A demonstration of thresholding-based segmentation in combination with Sobel filtering [252] on retinal imaging data acquired from Staal et al. [55] is illustrated in Figure 4.1. In recent years, thresholding declined to be present only in preprocessing methods for automated MIS and is commonly defined as clipping (see Chapter 3.3.5). MIS pipelines exclusively based on a thresholding method are rare and limited for special cases [10, 124]. Nevertheless, interactive thresholding is a standardized operation in radiology and is available in any modern PACS viewer.

Region Growing

The region growing technique is a semi-automatic method for MIS. The algorithm requires a manually selected seed point which is used to iteratively analyze neighboring pixels [124]. Starting from the seed point, if a neighboring pixel complies with predefined criteria (also called homogeneity criteria), it is added to the segmentation set and all its bordering pixels are also queued for criteria analysis [10, 124]. This process is repeated until there are no more neighboring pixels queued. Typically, homogeneity criteria are based on pixel intensity information, but can also define more complex features [124]. Nowadays, region growing is rarely applied exclusively in MIS pipelines but is similar to thresholding often integrated as a preprocessing method [124]. However, the application of region growing for annotation refinement is still a popular technique to increase annotation quality [156].

Atlas-based Segmentation

The concept of atlas-based segmentation approaches is to exploit prior knowledge by reusing already created segmentations from similar patient cases or tasks [10, 124]. An atlas acts as a

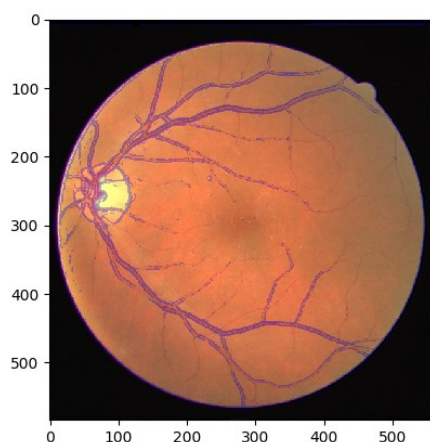


Figure 4.1: Thresholding approach for vessel segmentation in retinal imaging.

template for the searched object which is created manually by an expert based on its anatomy and spatial form. The resulting atlas can be overlaid on a new image in which the template provides the segmentation for the object. This overlay process is a medical image registration problem and, thus, algorithms for image registration are applied (often non-linear, non-parametric registration) [10, 253]. Because of the method requirement of similar cases, atlas-based segmentation can only be applied to uniform ROIs with a consistent form and location [124]. This excludes all types of MIS for abnormal feature detection like cancer or lesions. Still, atlas-based segmentation is typically applied for organ as well as brain region segmentation and is widely used in clinical workflows [10, 124].

Feature Classifier

The field of feature classifiers is derived from machine learning methods. Feature classifiers are general pattern recognition techniques based on supervised learning [124]. This means that annotated data is required for training a feature classifier model. The feature space of a classifier can be a representation of any information obtained from a medical image, whereas the most common feature space is the pixel intensity values of an image [124]. A typical implementation is the representation of an image as pixel intensity histogram in which the classifier is trained to identify pixel intensity patterns for specific structures to segment [124]. Due to the need for distinct quantifiable features and the disregard for spatial information, feature classifiers often perform inferior to other MIS methods [254].

Clustering

Similar to feature classifiers, clustering algorithms are pattern recognition techniques that try to differentiate a provided feature space to generate a segmentation [10, 124]. However, in contrast to feature classifiers, clustering algorithms are unsupervised which means that the methods do not require prior annotations for training. According to Handels [10] and Pham [124], one of the most popular clustering algorithms for MIS is k-means [255, 256]. Due to the feature space representation of an image, clustering algorithms also do not incorporate spatial

information. Nevertheless, based on the need for fast and uncomplicated segmentation in the presence of no available annotation data, unsupervised clustering approaches are still in the field of research, today [257–259].

Deformable Models

For accurate segmentation of structures in images, the utilization of spatial information is crucial. The concept of deformable models is to apply physically inspired algorithms for object boundary detection by approximating dynamic and plain rings to an object perimeter until the rings represent an optimal object boundary [10, 124, 260]. The approximation process can incorporate not only spatial image information but also prior knowledge about the structure and form of an object [10]. Still, deformable models are a semi-automatic segmentation method due to the need of initializing the model with a starting contour. Studies demonstrated that the resulting segmentation performance of deformable models is highly accurate and robust to noise [124, 260]. Before deep learning based MIS, deformable models were a highly popular segmentation method and an active research field [10, 124, 260].

Neural Networks and the State-of-the-Art

Despite the large number of methods for MIS, state-of-the-art accuracy is accomplished by convolutional neural networks and deep learning models [13, 26, 27, 36, 105, 261], which are used extensively today. The theory of neural networks can be found in Chapter 2.2. The newest convolutional neural networks are able to exploit spatial as well as pixel intensity value information [13, 74], local as well as global features in images [108, 109, 262], and can be trained to use 3D image information [109, 113, 117]. Instead of developing novel neural network architectures specifically for a medical imaging modality, it is common in MIA to reuse popular general computer vision architectures due to predictive capabilities, implementation availability, as well as their high adaptability to different tasks [75, 261]. Nevertheless, as presented in Chapter 2.3.2, the central architecture for modern general as well as medical image segmentation is the U-Net from Ronneberger et al. [108], which was introduced in the context of biomedical image segmentation. The U-Net has become the most widely used and best-performing architecture in MIS, which is present in any state-of-the-art MIS pipeline, today [13, 74, 105, 109, 125]. Although the majority of applications of deep learning based MIS are still focused on research, the field of segmentation with neural networks proved remarkable adaptability in terms of ROI type and imaging modality. In recent years, various studies using deep learning based pipelines demonstrated accurate segmentation capabilities for uniform structures like organs as well as for abnormalities like cancer or lesions [13, 103, 105, 125, 263]. Furthermore, applications in all kinds of medical imaging demonstrated its adaptability regardless of imaging modality [13, 103, 105, 125, 263].

Deep learning based MIS is the core method in all discussed studies of this thesis. The terms ‘convolutional neural networks’, ‘deep learning based MIS’ as well as ‘neural network models’ will be used as equivalent terms and be referred to as the main method for MIS in this work.

4.1.2 Challenges

The complexity of medical images represents a significant challenge for automatic image processing methods. The field of medical image segmentation is a particularly difficult task.

In terms of imaging complexity, medical images often have ambiguous image quality with a high amount of distortion and low contrast compared to regular imaging [182]. This is also augmented by the inconsistency and diffusion of pixel values in medical images [124, 182]. Furthermore, technical noise as well as artifacts can be present in images that handicap ROI detection [182]. Medical images also have high variability. This includes not only the human anatomy itself or the variance between individuals, but also the different modalities like microscopy, X-ray, or MRI [129].

Another challenge is the difficulty of distinguishing between benign and malignant abnormalities. Even for clinicians, it is a hard task that requires years of experience to develop highly accurate expertise. The associated challenge to clinician experience is the available annotated data to train a neural network model. Creating annotated data for supervised learning algorithms is a tedious and time-consuming task [13, 105]. Especially in MIS, the annotation process must be done manually by drawing exact masks on ROIs. This has to be done by experienced clinicians which have to detract their time from treating patients [13]. Moreover, low performance is not acceptable in the context of MIS. There is a constant need to minimize errors in any field, still, in the medical field, incorrect conjectures can directly lead to a life quality reduction or fatal consequences for patients [264].

Withal, implementations of MIS pipelines are usually independent software specialized on a single task and dataset [35, 265]. Consequently, MIS pipelines lack reproducibility in terms of replicating the pipeline with the same data, reusability in terms of utilizing the pipeline for another dataset in an equal context, and generalizability in terms of utilizing the pipeline for another context [32, 45]. Even so, the foundations of an MIS pipeline are equivalent, building such pipelines from scratch requires extensive knowledge in the field of MIS ('reinventing the wheel'). Because of that, the challenge of non-reproducible and non-standardized MIS pipelines drastically hampers practical usability and slows down progress in clinical research.

4.2 MIScnn: a Framework for Medical Image Segmentation with Convolutional Neural Networks

In recent years, medical image segmentation models with a convolutional neural network architecture have become quite powerful and achieved similar results performance-wise as radiologists [13, 15]. Nevertheless, these models have been standalone applications with optimized architectures, preprocessing procedures, data augmentations, and metrics specific to their dataset and corresponding segmentation problem [109]. Also, the performance of such optimized pipelines varies drastically between different medical conditions. However, even for the same medical condition, evaluation and comparisons of these models are a persistent challenge due to the variety of the size, shape, localization, and distinctness of different datasets. In order to objectively compare two segmentation model architectures from the sea of one-use standalone pipelines, each specific for a single public dataset, it would be required to implement a complete custom pipeline with preprocessing, data augmentation, and batch creation.

Frameworks for general image segmentation pipeline building cannot be fully utilized. The reason for this is their missing interfaces for medical imaging, their preprocessing methods, as well as their lack of handling highly unbalanced class distributions, which is standard in medical imaging. Recently developed medical image segmentation platforms, like NiftyNet [43], are powerful tools and an excellent first step for standardized medical image segmentation pipelines. However, they are designed more like configurable software instead of frameworks. They lack modular pipeline blocks to offer researchers the opportunity for easy customization and to help develop their own software for their specific segmentation problems.



Figure 4.2: Logo of the proposed framework MIScnn.

In this chapter, the Author proposes an intuitive and easy-to-use framework for fast setup of state-of-the-art convolutional neural network and deep learning models for medical image segmentation. The aim of the proposed framework MIScnn (pronunciation MIZ-C-N-N), short for **Medical Image Segmentation with Convolutional Neural Networks**, is to provide a complete pipeline for preprocessing, data augmentation, patch slicing, and batch creation steps to start straightforward with training and predicting on diverse medical imaging data. Instead of being fixated on one model architecture, MIScnn allows not only fast switching between multiple modern convolutional neural network models, but also provides the possibility to easily add custom model architectures. Additionally, it facilitates a simple deployment and fast usage of new deep learning models for medical image segmentation. Still, MIScnn is highly configurable due to adjustable hyperparameters, general training parameters, preprocessing procedures, including or excluding data augmentations, and evaluation techniques.

4.2.1 Implementation

The open-source Python library MIScnn is a framework for setup medical image segmentation pipelines with convolutional neural networks and deep learning models. MIScnn is providing several core features, which are also illustrated in Figure 4.3:

- 2D/3D medical image segmentation for binary and multi-class problems
- Data I/O, preprocessing, and image augmentation for biomedical images
- Patch-wise and full image analysis
- State-of-the-art deep learning model and metric library
- Intuitive and fast model utilization (training, prediction)
- Multiple automatic evaluation techniques (e.g. cross-validation)
- Custom model, data I/O, pre-/postprocessing, and metric support

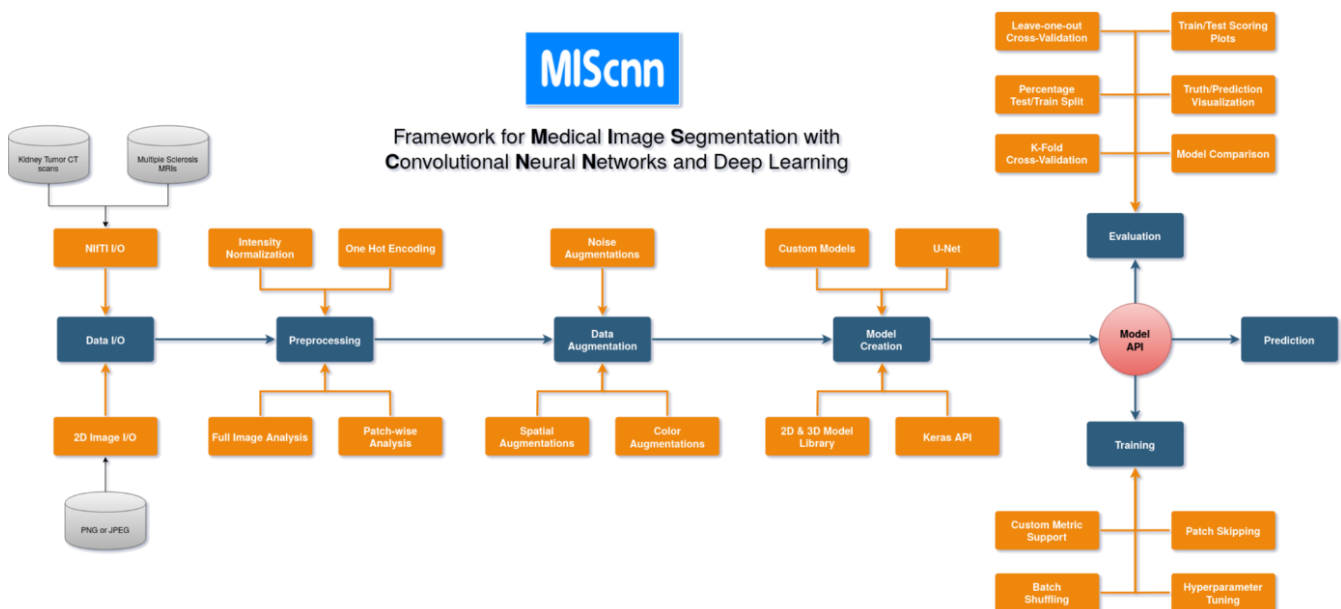


Figure 4.3: Flowchart diagram of the MIScnn pipeline.

Data Loading

An MIS pipeline performs several types of data input and output (I/O) operations. In a minimalistic pipeline, the loading of images as well as annotations as model input and the storage of predicted segmentation masks are needed. In more efficient pipelines processing large datasets, temporary data like preprocessed images, model weights, and evaluation results are also stored. The complete data flow and storage handling is done by the Data IO class in MIScnn. The Data IO class is the gateway into the MIScnn pipeline and therefore the first of four core classes to define a pipeline in MIScnn.

Furthermore, the Data IO module enables MIScnn to handle a wide variety of imaging formats by integrating a switchable I/O interface and offers the possibility to integrate any user-required image format or data structure into the MIScnn pipeline. The aim of an I/O Interface is to handle

the loading and saving process of images in their specific formats and file structure. The user is only required to create an instance of the Data IO class with the desired specifications and I/O interface for the correct format. MIScnn provides already implemented I/O interfaces for the following data formats: Regular image formats (PNG, JPG, TIFF) based on the Pillow package [266], NIfTI based on the nibabel package [267], DICOM based on the SimpleITK [134, 135] and pydicom [268] packages, and images encoded as NumPy [138] matrices in a Python dictionary to support native usability.

Next to the implemented I/O interface, MIScnn allows the usage of custom data I/O interfaces for non-common imaging data formats. This open interface enables MIScnn to handle specific biomedical imaging features (e.g. MRI slice thickness), and therefore it avoids losing these feature information by a format conversion requirement. A custom I/O interface must be committed to the preprocessing function, and it has to return the medical image as a 2D or 3D matrix (in a NumPy data structure [138]) for integration in the workflow. It is advised to add format-specific preprocessing procedures (e.g. MRI slice thickness normalization) in the format-specific I/O interface, before returning the image matrix into the pipeline.

Preprocessing

The aim of preprocessing methods is to extensively increase performance due to simplification of information as well as to reduce image information in order to be passable into the neural network model. Thus, it is commonly required to perform extensive preprocessing on medical images. To provide a wide variety of preprocessing methods, MIScnn offers Subfunction modularity. The user is able to create a list of desired preprocessing functions (in MIScnn called Subfunctions) and pass them to the Preprocessor class, which is one of the four core classes in a MIScnn pipeline. MIScnn provides the following preprocessing techniques: Pixel intensity normalization (Z-Score, MinMax, and grayscale), clipping, resizing, resampling and padding. For more details about preprocessing methods, see Chapter 3.3.

MIScnn is able to handle binary (background/cancer) as well as multi-class (background/kidney/liver/lungs) segmentation problems. The representation of a binary segmentation is made quite simple by using a variable with two states, zero and one. But for the processing of multiple categorical segmentation labels in machine learning algorithms, like deep learning models, it is required to convert the classes into a more mathematical representation. This can be achieved with the One Hot Encoding method by creating a single binary variable for each segmentation class. MIScnn automatically One Hot encodes segmentation labels with more than two classes.

Patch-wise and Full Image Analysis

Depending on the resolution of medical images, the available GPU hardware plays a significant role in 3D segmentation analysis. Currently, it is not possible to fully fit high-resolution MRIs with an example size of 400x512x512 pixels into state-of-the-art convolutional neural network models due to the enormous GPU memory requirements. Therefore, the 3D medical imaging

data can be either sliced into smaller cuboid patches or analyzed slice-by-slice, similar to a set of 2D images [13, 27, 269]. In order to fully use the information of all three axes, MIScnn slices 3D medical images into patches with a configurable size (e.g. 128x128x128 pixels) by default. Depending on the model architecture, these patches can fit into GPUs with RAM sizes of 4GB to 24GB, which are commonly used in research. Nevertheless, the slice-by-slice 2D analysis as well as the 3D patch analysis are supported in MIScnn. It is also possible to configure the usage of full 3D images in case of analyzing uncommonly small medical images or having a large GPU cluster. By default, 2D medical images are fitted completely into convolutional neural network models. Still, a 2D patch-wise approach for high-resolution images can be also applied which is essential for pathology imaging.

Image Augmentation

In the machine learning field, data augmentation covers the artificial increase of training data. Especially in medical imaging, commonly only a small number of samples or images of a studied medical condition are available for training [13, 164–166, 181]. Thus, an image can be modified with multiple techniques, like shifting, to expand the number of plausible examples for training. The aim is to create reasonable variations of the desired pattern in order to avoid overfitting in small datasets [165]. For state-of-the-art image augmentation, MIScnn integrated the batchgenerators package from the Division of Medical Image Computing at the German Cancer Research Center (DKFZ) [142]. It offers various on-the-fly image augmentation techniques for model training and was used by the winners of the latest medical image processing challenges [109, 200, 270]. It supports spatial translations, rotations, scaling, elastic deformations, brightness, contrast, gamma, and noise augmentations like Gaussian noise. The Data Augmentation class represents an optional core class for building a MIScnn pipeline.

Sampling and Batch Generation

The large unbalance between the relevant segments and the background in medical images results in an extensive amount of parts in an image purely labeled as background and without any learning information [13, 181]. Especially after image augmentation, there is no benefit to multiplying these blank parts or patches [271]. Therefore, in the patch-wise model training, all patches, which are completely labeled as background, can be excluded to avoid wasting time on unnecessary fitting.

After the data preprocessing and the optional image augmentation for training, sets of images or patches are bundled into batches. One batch contains a number of prepared images which are processed in a single step by the model and GPU. Sequential for each batch or processing step, the neural network updates its internal weights accordingly with the predefined learning rate. The possible number of images inside a single batch highly depends on the available GPU memory and has to be configured properly in MIScnn. Every batch is saved to disk in order to allow fast repeated access during the training process. This approach drastically reduces the computing time due to the avoidance of unnecessary repeated preprocessing of the batches. Nevertheless, this approach is not ideal for extremely large datasets or researchers without the

required disk space. To bypass this problem, MIScnn also supports on-the-fly generation of the next batch in memory during runtime.

During model training, the order of batches, which are going to be fitted and processed, is shuffled at the end of each epoch. This method reduces the variance of the neural network during fitting over an epoch and lowers the risk of overfitting. Still, it must be noted, that only the processing sequence of the batches is shuffled and the data itself is not sorted into a new batch order.

MIScnn also supports the usage of multiple GPUs and parallel CPU batch loading next to the GPU computing. Particularly, the storage of already prepared batches on disk enables fast and parallelizable processing with CPU as well as GPU clusters by eliminating the risk of batch preprocessing bottlenecks.

Neural Network Model

The selection of a deep learning or convolutional neural network model is the most important step in an MIS pipeline and represents the last core class in MIScnn. But there is a large variety of model architectures, and each has different strengths and weaknesses [13, 27, 36, 118, 200, 261, 269]. MIScnn features an open model interface to load and switch between provided state-of-the-art convolutional neural network models like the popular U-Net model [108]. Models are represented with the open-source base framework TensorFlow [72] which provides a user-friendly API via Keras [272] for commonly used neural network building blocks. The already implemented models are highly configurable by a definable number of neurons, custom input sizes, optional dropout or batch normalization layers, and enhanced architecture versions like the Optimized High-Resolution Dense-U-Net model [113]. Additionally, MIScnn offers architectures for 3D as well as 2D MIS. This model selection process is visualized in Figure 4.4. Besides the flexibility in switching between already implemented architectures, the open architecture interface enables the ability for custom deep learning architecture implementations and simple integration of these custom models into the MIScnn pipeline.

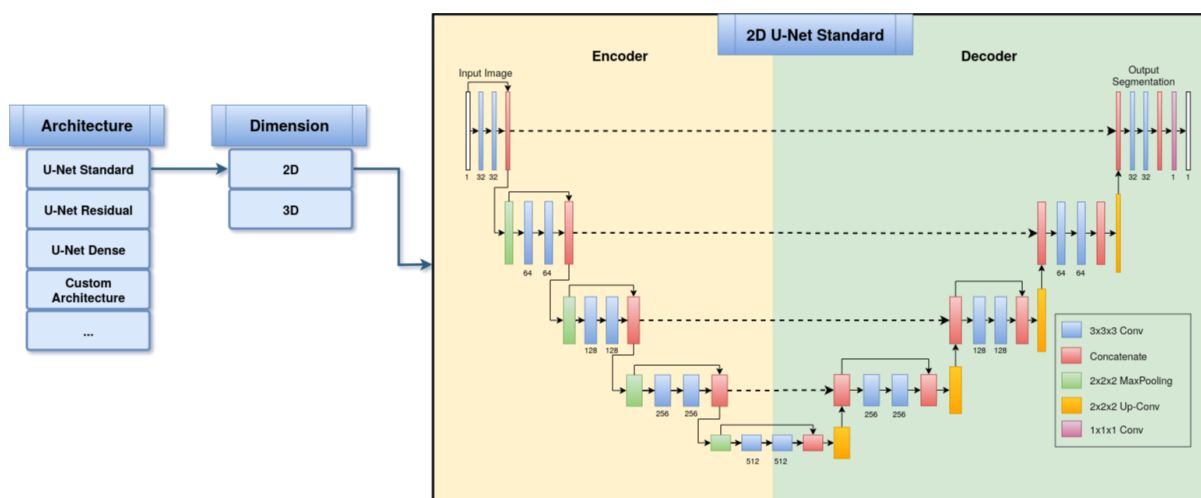


Figure 4.4: Flowchart visualization of the deep learning model selection process.

Loss Functions and Metrics

MIScnn offers a large quantity of various metrics which can be used as loss function for training or evaluation in figures and performance analysis. The Dice Similarity Coefficient is one of the most popular metrics for medical image segmentation. Depending on the segmentation task (binary or multi-class), there is a simple and class-wise DSC implementation in MIScnn. Whereas the simple implementation only accumulates the overall number of correct and false predictions, the class-wise implementation accounts for the prediction performance for each segmentation class which is strongly recommended for commonly class-unbalanced medical images. Still, to compensate for the class bias in versatile situations, the sum of the Tversky index and the categorical cross-entropy is utilized as a loss function for model fitting by default in MIScnn. For more details on the individual loss functions, see Chapter 3.5.2. Thus, the default loss function ensures good performance on binary, as well as multi-class segmentation.

$$L_{\text{total}} = L_{\text{Tversky}} + L_{\text{Cross-Entropy}} \quad (4.1)$$

Furthermore, all standard metrics which are included in TensorFlow, like accuracy or cross-entropy, can be used in MIScnn. Next to the already implemented metrics or loss functions, MIScnn offers the integration of custom metrics for training and validation. A custom metric can be implemented as defined in TensorFlow (Keras), and simply be passed to the deep learning model.

Model Utilization

With the initialized deep learning model and the fully preprocessed data, the model can now be used for training on the data to fit model weights or for prediction by using an already fitted model. Alternatively, the model can perform an automatic evaluation, as well, by running a cross-validation for example, with multiple training and prediction calls. The model API allows saving and loading models in order to subsequently reuse already fitted models for prediction or for sharing pre-trained models.

In the process of training a convolutional neural network or deep learning model, diverse settings have to be configured. At this point in the pipeline, the image augmentation options of the dataset, which have a strong influence on the training in medical image segmentation, must be already defined. Sequentially, the batch management configuration covered the settings for the batch size, and also the batch shuffling at the end of each epoch. Therefore, only the learning rate and the number of epochs are required to be adjusted before running the training process. The learning rate of a neural network model is defined as the extent to which the old weights of the neural network model are updated in each iteration or epoch. In contrast, the number of epochs defines how many times the complete dataset will be fitted into the model. Sequentially, the resulting fitted model can be saved to disk. During the training process, the underlying TensorFlow base framework provides insights into the current model performance with the predefined metrics, as well as the remaining fitting time. Additionally, MIScnn offers the usage

of a fitting curve evaluation functionality in which the fitting scores and metrics are stored in a tab-separated file or directly plotted as a figure.

For the segmentation prediction, an already fitted neural network model can be directly used after training or it can be loaded from a file. For every pixel, the model predicts a Softmax value for each class. The Softmax value represents a probability estimation of this pixel for the associated label. Sequentially, the argmax of the One Hot encoded class is identified for multi-class segmentation problems and then converted back to a single result variable containing the class with the highest Softmax value. If using the overlapping patch-wise analysis approach during the training, MIScnn supports two methods for patches in the prediction. Either the prediction process creates distinct patches and treats the overlapping patches during the training as purely image augmentation, or overlapping patches are created for prediction. Due to the lack of prediction power at patch edges, computing a second prediction for edge pixels in patches, by using an overlap, is a commonly used approach. In the following merge of patches back to the original medical image shape, a merging strategy for the pixels is required, in the overlapping part of two patches and with multiple predictions. By default, MIScnn calculates the mean between the predicted Softmax values for each class in every overlapping pixel. The resulting image matrix with the segmentation prediction, which has an identical shape as the original medical image, is saved into a file structure according to the provided data I/O interface.

MIScnn supports multiple automatic evaluation techniques to investigate MIS performance: k-fold cross-validation, leave-one-out cross-validation, percentage-split validation, hold-out sets for testing (dataset split into test and train set with a given percentage), and detailed validation in which it can be specified which images should be used for training and testing. Except for the detailed validation, all other evaluation techniques use random sampling to create training and testing datasets. During the evaluation, the predefined metrics and loss function for the model are automatically plotted in figures and saved in TSV (Tab Separated Vector) files for possible further analysis. Next to the performance metrics, the pixel value range and segmentation class frequency of medical images can be analyzed in the MIScnn evaluation. Also, the resulting prediction can be compared directly next to the ground truth by creating image visualizations with segmentation overlays. For 3D images, like MRIs, the slices with the segmentation overlays are automatically visualized in the Graphics Interchange Format (GIF).

4.2.2 Open-Source Development and Deployment

To establish a standardized and community-accepted framework, it is essential to utilize state-of-the-art development and deployment techniques for open-source projects. The main idea of open-source projects is to encourage users to build a community around the software package. An adequate open-source project community has a strong interest in keeping the project ‘alive’ resulting in a community effort to provide consistent software updates and support [273]. To achieve this, it is vital to create a robust foundation in which new users can contribute without obstacles [273]. Furthermore, there are several features that can significantly help an open-source project flourish. The development process should be naturally open-source either already

at the start or after a certain release to the public. Next to the development, the software package should already provide high usability in terms of API simplicity, functionality in terms of error-free execution, and stability in terms of robustness in multiple environments as well as simple installation processes [273]. Moreover, APIs with open interfaces for custom contributions are particularly useful in research.

The following subchapter points out the efforts of the Author to establish MIScnn as a standardized and community-accepted framework.

Philosophy of MIScnn

Based on the open-source principles, MIScnn has been developed to comply with the following philosophies:

➤ *User friendliness*

MIScnn is an intuitive API designed for human beings, not machines. With a stronger growing interest in medical imaging processing, building MIS pipelines should not be like ‘inventing the wheel’ for every new user. MIScnn offers consistent and simple APIs for minimizing the number of user actions required for common use cases.

➤ *Modularity*

The general steps in an MIA pipeline are identical in the majority of projects. Nevertheless, switching to another neural network architecture or dataset format breaks most publicly available medical image processing software, today. MIScnn attempts to change this situation. In particular, data I/O, pre-/postprocessing functions, metrics, and model architectures are standalone interfaces that should be easily switchable for customization.

➤ *Easy extensibility*

New interfaces should be simple to integrate into the MIScnn pipeline. Existing interfaces are documented and robust. Additionally, MIScnn provides abstract base classes (class templates in Python) for all interfaces which help define the structure and setup of custom implementations. The aim is to ensure easy integration of user contributions or adapt MIScnn to new data structures.

➤ *Working with Python*

MIScnn can be used in Python, which is compact, easier to debug, allows easier deployment and integration into workflows due to its high popularity in deep learning based computer vision [273]. The utilization of an intuitive framework guides to better understanding and customization instead of a standalone ‘black box’ software.

Open-Source Development

The Git software is a version control system and the gold-standard for any state-of-the-art development of small as well as large programming projects [274, 275]. Git allows not only high-quality data assurance and to setup staging areas (multiple programming environments) but also efficient collaborated programming through branching and merging strategies [275]. These features make Git the backbone of any open-source project [274].

For hosting a Git repository (which is a programming project in Git), there are multiple platforms that offer distributed version control for programming development and other features. The most popular and widely used platform is GitHub [219] in which MIScnn is also hosted. Other popular Git based platforms are GitLab [276] and BitBucket [277]. Next to Git based version control, GitHub also offers various integrated features like an issue and pull request system as well as further infrastructure for hosting development operations (DevOps),

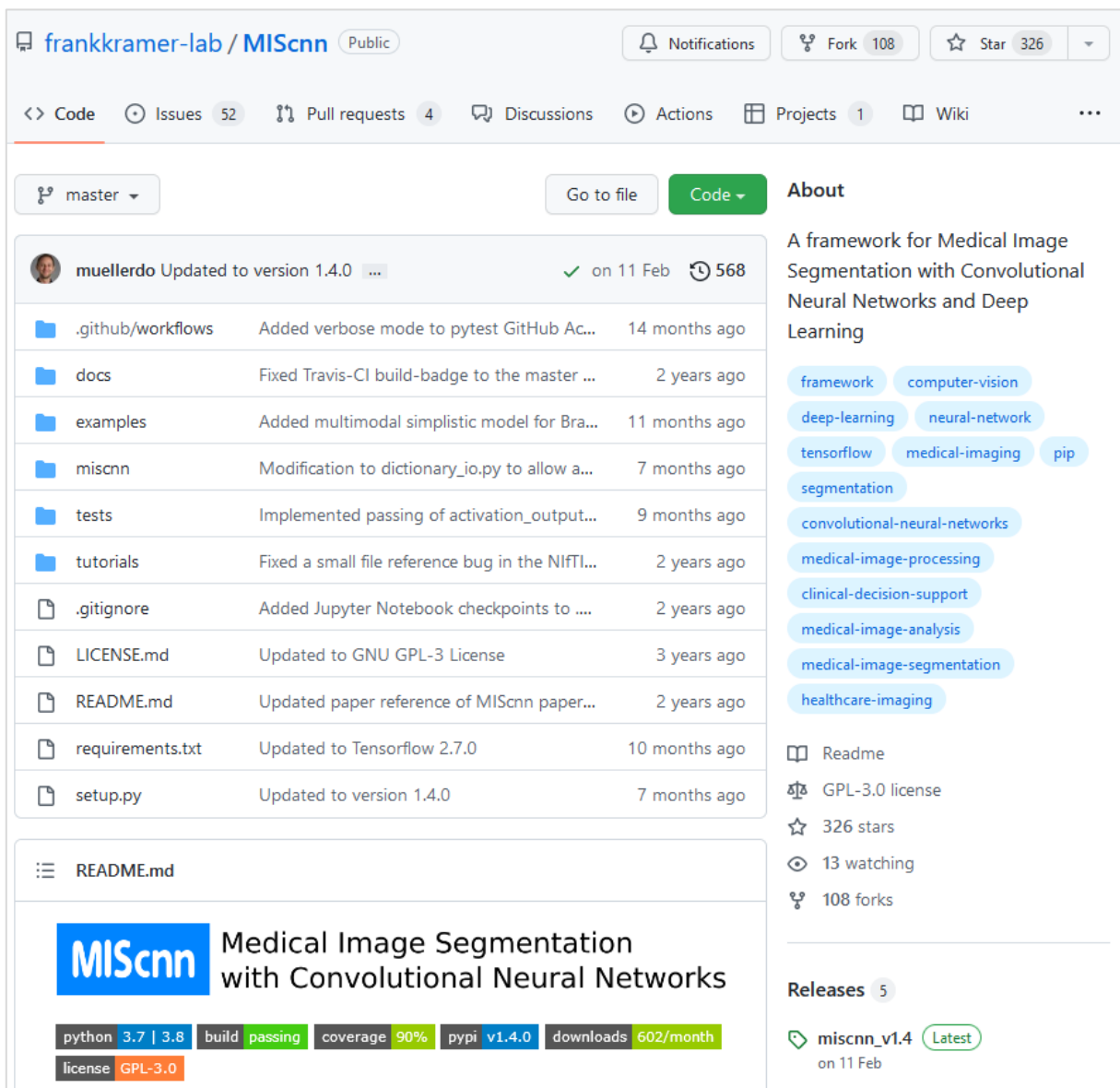


Figure 4.5: Starting page of the MIScnn repository on GitHub.

project management techniques, and structured package releases. Especially for open-source projects, the option for a user to directly communicate to the developers for questions, bug reporting or feature requests is essential. Furthermore, the powerful interface of GitHub for reviewing community contributions (pull requests) strongly improves efficiency in collaboration.

The proposed framework MIScnn is centralized in a GitHub repository. The complete and still ongoing development process has been done in a private Git repository since the project started in May 2019. The repository was publicly available with the first release (version 0.1) published in July 2019. The development of the MIScnn code is sorted into three types of branches. The master branch is the main code stage which is reserved for stable releases. The development branch contains the current working state of MIScnn but can house non-functional features which are still a work in progress. The remaining branches are for individual feature development until the implementation is ready to be merged in the development branch with a pull request.

Besides the version control system, GitHub also offers a full issue and pull request system. Whereas the issue system allows users to report bugs, feature requests as well as open questions, contributors can utilize pull requests to enhance MIScnn features. The repository is still actively maintained through this communication system. Another feature for the development process is the task organization by the integrated project management system on GitHub which is called the ‘development tracker’ in the MIScnn project. The management system is based on an intuitive canvas board illustrating active issues and notes which are sorted into the custom-defined groups: Support/Bugs, Development, In Progress, Done, Ideas, and Future Directions. This public task management ensures a transparent roadmap of the project and informs the community on which features are going to be worked on in the future.

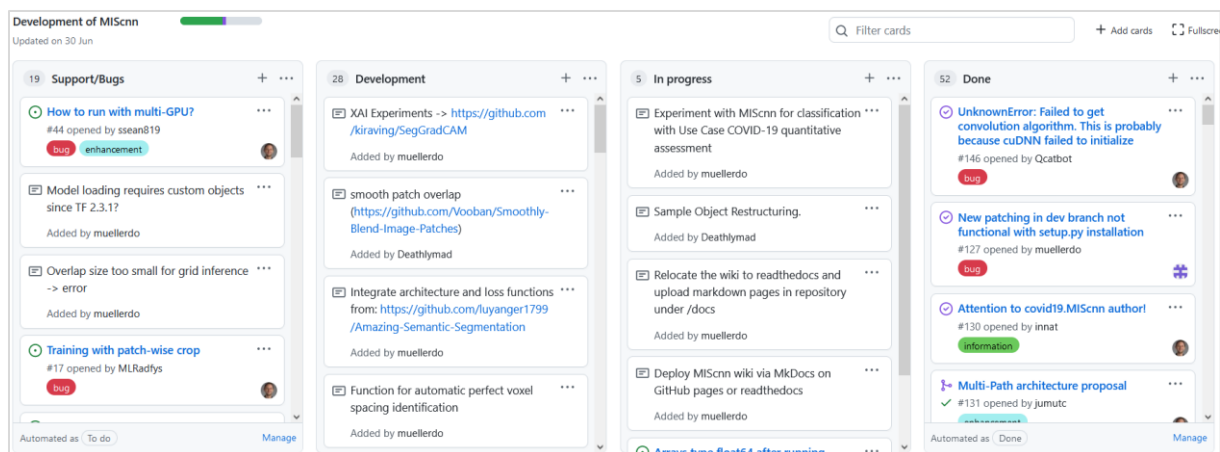


Figure 4.6: Extract of the task management system on GitHub for MIScnn.

Still, the repository does not only provide functionality for the development process but also acts as a hub for the complete MIScnn project. Next to formal information like software licensing, reference to the published article, and the address of the corresponding author for

further collaboration studies, the repository also provides important resources like getting started information, documentation, examples as well as tutorials. More details about the documentation resources of MIScnn can be found in the next subchapter.

Documentation

Next to development-related features, the MIScnn repository also provides extensive documentation. Well-written documentation is fundamental for any open-source project by allowing users to quickly understand the software appliance, utilize functions without reading the source code, and provide further information on pitfalls as well as starting points for problem-solving [274]. There are different types of software documentation ranging from manuals, technical information about functions or interfaces, and software requirements, to provenance information like changelogs. Whereas a version control system automatically provides the majority of provenance information in software development, common forms of documentation in machine learning are wikis for APIs and example applications [278].

All core features of MIScnn are documented in a wiki. The wiki is hosted also on GitHub and is directly associated with the Git repository. It is represented through another hidden Git repository containing the wiki pages encoded in the markdown file format. An example page of the MIScnn wiki is illustrated in Figure 4.7. The API functions like core classes and interfaces are all documented including the function parameters, a short usage example, as well

Home Edit New page
muellerdo edited this page on 30 Jul 2020 · 8 revisions

MIScnn Medical Image Segmentation with Convolutional Neural Networks

The open-source Python library MIScnn is an intuitive API allowing fast setup of medical image segmentation pipelines with state-of-the-art convolutional neural network and deep learning models in just a few lines of code.

MIScnn provides several core features:

- 2D/3D medical image segmentation for binary and multi-class problems
- Data I/O, preprocessing and data augmentation for biomedical images
- Patch-wise and full image analysis
- State-of-the-art deep learning model and metric library
- Intuitive and fast model utilization (training, prediction)
- Multiple automatic evaluation techniques (e.g. cross-validation)
- Custom model, data I/O, pre-/postprocessing and metric support
- Based on Keras with Tensorflow as backend

Philosophy of MIScnn

User friendliness:
MIScnn is an intuitive API designed for human beings, not machines. With a stronger growing interest in medical imaging processing, building medical image segmentation pipelines

Pages 17

Getting Started

- What is MIScnn?
- Install
- Basic Usage
- Tutorials
- Examples

Documentation - Core

- Data I/O
- Preprocessor
- Data Augmentation
- Neural Network Model
- Validation Techniques

Documentation - Interfaces

- Architectures
- Metrics
- Subfunctions
- I/O Interfaces

Documentation - Extras

- Sample

Figure 4.7: Extract of the MIScnn wiki starting page on GitHub.

as further details or warnings. API documentation examples are illustrated in the subchapter Core Classes from Chapter 4.2.3.

Table 4.1: Overview of available examples for MIScnn.

Example Name	Clinic Department	Description
KiTS19	Radiology	3-fold cross-validation on 3D CT scans for kidney tumor segmentation.
ISBI-CTC15	Pathology	Leave-One-Out validation on 2D microscopy imaging for cell segmentation and tracking.
COVID-19	Radiology	5-fold cross-validation with limited 3D CT scans for COVID-19 segmentation.
LCTSC	Radiology	Percentage-Split validation with lung CT scans in DICOM for organ segmentation.
RetinalSeg	Ophthalmology	Segmentation of blood vessels (retinal vascular tree) in retinal images.
BraTS2020	Neurology	Segmentation of brain tumors (glioma) in multi-modal MRI scans.

Alongside the wiki, complete pipelines of MIScnn are presented and explained in separate examples. Studying an implementation example or open-source project which applies the proposed software is often the first step for new users or researchers [278]. An example can provide various types of information, whether it is how to apply the software or in what context the software can be used. Thus, to offer quick familiarization with the API, MIScnn provides multiple documented examples which can be seen in Table 4.1. The examples are either larger projects with separate Git repositories, or smaller studies on simple datasets as well as implemented in Jupyter Notebooks [278] with explanatory sections. Furthermore, MIScnn provides three tutorials which are short examples demonstrating only a single feature instead of a complete MIS pipeline.

Continuous Integration

Another fundamental concept of modern open-source projects is continuous integration (CI) strategies. The concept of continuous integration describes principles for automatic as well as permanent rebuilding, testing, and evaluation of the software to ensure functionality and quality [279]. In a Git environment, this process is often triggered at certain release stages or ideally at each commit push (uploading of new programming code). Primarily, immediate detection of errors when introducing new code is highly important for code stability and robustness, especially for medical software. The testing process of software is called regression and unit testing. Karl Fogel describes the terms in his work as follows [274]:

“Regression testing means testing that working software stays working. Its purpose is to reduce the chances that code changes will break the software, particularly in ways the software has been broken before. Many projects have a regression test suite, a separate program that invokes the project's software with the expectation of particular inputs causing particular behaviors. If the test suite succeeds in causing a different behavior to happen, this is known as a regression, meaning that someone's change unexpectedly broke something else.”

Unit testing means testing the software's module boundaries using their documented APIs. Its purpose is both to reduce the chance that modifications will break existing functionality, and to prove that the intended functionality exists as claimed. If regression testing is retrospective ("What has broken in the past?"), unit testing is prospective ("What do we need to ensure continues to work into the future?"). As with regression tests, many projects have a unit test suite."

In this thesis, regression and unit testing will be used simultaneously by the term 'unit testing'. MIScnn integrates extensive unit testing to provide consistent functionality and high software quality. This allows not only a more efficient reviewing and integration of community contributions, but also significantly improve overall development quality. The unit testing consists of general pipeline testing with multiple types as well as formats of imaging data, core class testing with various parameter options, individual function testing, and overall edge case testing for the complete MIScnn API. In total, 152 unit tests were implemented. An overview of unit tests grouped by submodules in MIScnn is summarized in Table 4.2. The server

```
build (3.8) ✓
succeeded on 22 Jun in 2m 44s

> ✓ Set up job 25s
> ✓ Run actions/checkout@v2 35s
> ✓ Set up Python 3.8 0s
> ✓ Install dependencies 1m 13s
▼ ✓ Test with pytest 1m 24s

1 ▶ Run pytest -v
7 ===== test session starts =====
8 platform linux -- Python 3.8.12, pytest-7.1.2, pluggy-1.0.0 -- /opt/hostedtoolcache/Python/3.8.12
  /x64/bin/python
9 cachedir: .pytest_cache
10 rootdir: /home/runner/work/MIScnn/MIScnn
11 plugins: typeguard-2.13.3
12 collecting ... collected 152 items
13
14 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_attention PASSED [ 0%]
15 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_attention_residual PASSED
  [ 1%]
16 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_compact PASSED [ 1%]
17 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_dense PASSED [ 2%]
18 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_multires PASSED [ 3%]
19 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_plain PASSED [ 3%]
20 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_residual PASSED [ 4%]
21 tests/test_architectures.py::architectureTEST::test_ARCHITECTURES_UNET_standard PASSED [ 5%]
22 tests/test_cli.py::cliTEST::test_CLI_setup PASSED [ 5%]
23 tests/test_cli.py::cliTEST::test_CLIbinningAnalysis PASSED [ 6%]
```

Figure 4.8: Extract of the CI pipeline logs for MIScnn on GitHub.

infrastructure for automatically running the unit testing was initially hosted at the commercial platform TravisCI [280] and integrated into the GitHub environment of the MIScnn repository. However, due to service changes of TravisCI (ending special support for open-source projects) in November 2020, the majority of CI workflows of MIScnn were relocated to the GitHub infrastructure. Thus, all resources of MIScnn were centralized at a single infrastructure host which drastically reduced integration complexity.

Table 4.2: Distribution of unit tests for the MIScnn framework.

Submodule	Unit Tests
Command Line Interface	9
Data I/O	13
IO Interfaces	16
Data Augmentation	16
Patch Operations	4
Subfunctions	19
Preprocessor	15
Data Generator (interface between Preprocessor and Neural Network)	9
Neural Network	15
Deep Learning Architectures	8
Loss Functions and Metrics	17
Multi-Model Module (Ensemble Learning)	7
Evaluation	4

In order to further evaluate the code quality, the software metric unit testing coverage was computed with the Codecov platform [281]. The code coverage can be defined as the number of code lines that are covered in the unit testing. A low coverage indicates that only individual parts of an API or framework are tested indicating that a large percentage of the software is vulnerable to undetected errors, whereas a high coverage indicates software robustness and functionality. For MIScnn, 3,753 code lines from a total of 4,166 lines (without documentation and formatting lines) are unit tested (state: 13.09.2022) which results in coverage of 90.09%.

Package Availability

The source code and further resources like documentation are available in the Git repository: <https://github.com/frankkramer-lab/MIScnn>. MIScnn is licensed under the open-source GNU General Public License Version 3.

For deployment in a Python environment, MIScnn is also available in the Python Package Index (PyPI) which is the official package store for Python. The Python Package Index allows not only installing already built ‘wheels’ (allowing faster installation) but also the automatic building from the source code of the package. Therefore, the MIScnn framework can be directly installed as a Python library using *pip install miscnn*.

4.2.3 API Usage

The framework MIScnn is a toolkit for medical image segmentation and, thus, can be utilized as an API in a Python environment for building MIS pipelines. This subchapter describes the installation process, the documentation of the four core classes of MIScnn, and a simple example of how to use MIScnn.

Installation

There are two options to install MIScnn in a new Python environment. The recommended way of installation is using PyPI (more information on PyPI can be found in the *Package Availability* section of Chapter 4.2.2). The corresponding installation manager for PyPI is the *pip* package. It is the official preferred installer program of the Python language (since version 3.4) [282]. The strongest advantage of utilizing *pip* for installation is the fully automatic and simplistic installation of package dependencies. The alternative installation option is the manual installation from the source code. This can be achieved by downloading MIScnn from the Git repository (GitHub) and installing it by calling the installation script *setup.py*. Similar to *pip*, the manual installation script also supports simplistic dependency installation, but can not handle more complex dependency issues. Furthermore, *pip* offers additional virtual environment support compared to the manual *setup.py* installation [283] and does not require manual source code download beforehand. Both installation options are demonstrated in Code Snippet 4.1.

```
1 # Installation via PyPI
2 pip install miscnn
3
4 # Installation from Source
5 git clone https://github.com/frankkramer-lab/MIScnn
6 cd MIScnn/
7 python setup.py install
```

Code Snippet 4.1: Installation process of the proposed MIScnn framework.

Despite the automatic dependency installation, it is still recommended to setup a TensorFlow distribution on the system prior MIScnn installation in order to ensure full library functionality. The reason for this is that TensorFlow relies on a large amount of operational systems drivers like CUDA [220]. After the installation process, the MIScnn API can be utilized in any associated Python environment.

Core Classes

The MIScnn pipeline is represented by four core classes as already described in Chapter 4.2.1:

- Data I/O
- Data Augmentation (optional)
- Preprocessor
- Neural Network Model

These classes handle all required steps for medical image segmentation and can be extensively customized. All classes, except the Data Augmentation class, use switchable interfaces which results in high configurability and customizability offering simple integration of user-defined solutions.

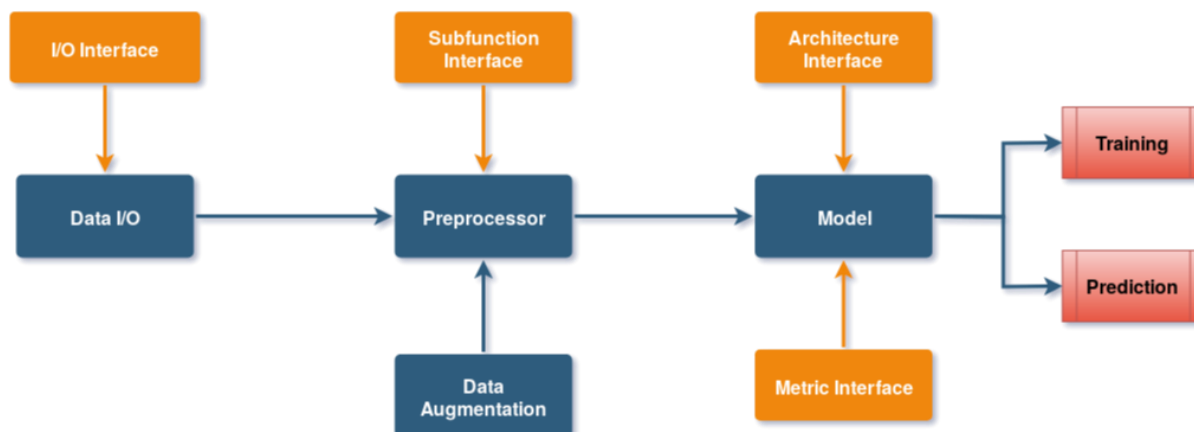


Figure 4.9: Workflow of the four core classes in the MIScnn API.

The first core class of the MIScnn API is the Data IO class. This class provides functionality for handling all input and output processes of the imaging data, as well as the temporary backup of batches to the disk. A user is only required to create a single instance of the Data IO class with the desired specifications and IO interface for the correct format. It is possible to create a custom IO interface for handling special data structures or formats. The docstring and argument descriptions are summarized in Figure 4.10.

```
Data_IO(interface, input_path, output_path="predictions",
        batch_path="batches", delete_batchDir=True)
```

Initialization function for creating an object of the Data IO class.

Arguments:

- **interface:** Data I/O interface object.
- **input_path:** Path to the input image data directory (passed to Data I/O interface object).
- **output_path:** Path to the output directory for predictions (passed to Data I/O interface object).
- **batch_path:** Path to the directory for temporary files like batches or preprocessing files.
- **delete_batchDir:** Boolean, whether the temporary batch directory should be deleted, afterwards.

Returns:

A Data_IO class object. Have to be passed to the Preprocessor class and can be used to access the sample list.

Example:

```
# Create a Data I/O interface for kidney tumor CT scans in NIFTI format
from miscnn.data_loading.interfaces import NIFTI_interface
interface = NIFTI_interface(pattern="case_000[0-9]*", channels=1, classes=3)

# Initialize data path and create the Data I/O instance
data_path = "/home/mudomini/projects/KITS_challenge2019/kits19/data.original/"
data_io = miscnn.Data_IO(interface, data_path)
```

Figure 4.10: Extract of the documentation entry for the Data IO class from the MIScnn wiki.

```
Data_Augmentation(cycles=1, scaling=True, rotations=True,  
                  elastic_deform=False, mirror=False, brightness=True,  
                  contrast=True, gamma=True, gaussian_noise=True)
```

Initialization function for creating an Data Augmentation object.

Figure 4.11: Docstring of the Data Augmentation class from the MIScnn wiki.

Data augmentation on images is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. The point of data augmentation is, that the model will learn meaningful patterns instead of meaningless characteristics due to small dataset size. The Data Augmentation class in MIScnn is the second as well as an optional core class. The following Preprocessor, which is the third core class, automatically initializes a Data Augmentation class with default values, but by manually initializing the optional class, it is possible to adjust which augmentation techniques should be applied to the dataset. The docstring and argument descriptions of the Data Augmentation class are summarized in Figure 4.11 and Figure 4.12.

Arguments:

- **cycles:** Number of augmented image copies that should be created.
- **scaling:** Boolean, whether scaling should be performed as data augmentation.
- **rotations:** Boolean, whether rotations should be performed as data augmentation.
- **elastic_deform:** Boolean, whether elastic deformation should be performed as data augmentation.
- **mirror:** Boolean, whether mirroring should be performed as data augmentation.
- **brightness:** Boolean, whether brightness changes should be added as data augmentation.
- **contrast:** Boolean, whether contrast changes should be added as data augmentation.
- **gamma:** Boolean, whether gamma changes should be added as data augmentation.
- **gaussian_noise:** Boolean, whether Gaussian noise should be added as data augmentation.

Returns:

A Data_Augmentation class object. Have to be passed to the Preprocessor class.

Example:

```
# Initialize  
data_aug = Data_Augmentation(cycles=2,  
                              scaling=False, rotations=False,  
                              elastic_deform=False, mirror=False,  
                              brightness=False, contrast=False,  
                              gamma=True, gaussian_noise=True)  
  
# Further configurations  
data_aug.config_p_per_sample = 0.35  
data_aug.config_contrast_range = (1, 2)  
  
# Pass to Processor  
pp = Preprocessor(data_io, data_aug=data_aug)
```

Figure 4.12: Documented argument descriptions of the Data Augmentation class from the MIScnn wiki.

The third core class in the MIScnn API is the Preprocessor class which provides functionality for handling all preprocessing methods in the pipeline. This includes diverse optional processing subfunctions like resampling, clipping, normalization, or custom subfunctions. This

```
Preprocessor(data_io, batch_size, subfunctions=[],
             data_aug=Data-Augmentation(), prepare_subfunctions=False,
             prepare_batches=False, analysis="patchwise-crop",
             patch_shape=None, use_multiprocessing=False)
```

Initialization function for creating a Preprocessor object.

Arguments:

- **data_io**: Data IO class instance which handles all I/O operations.
- **batch_size**: Number of samples inside a single batch.
- **subfunctions**: List of Subfunctions classes which will be SEQUENTIALLY executed on the data set.
- **data_aug**: Data Augmentation class instance which performs diverse data augmentation techniques. If no Data Augmentation is provided, an instance with default settings will be created. Use `data_aug=None`, if you want no data augmentation at all.
- **prepare_subfunctions**: Boolean, whether all subfunctions should be prepared and backup to disk before starting the batch generation (True), or should the subfunctions preprocessing be performed during runtime? (False).
- **prepare_batches**: Boolean, whether all batches should be prepared and backup to disk before starting the training (True), or should the batches be created during runtime? (False).
- **analysis**: String. Modus selection of analysis type. Options: ["fullimage", "patchwise-crop", "patchwise-grid"]
- **patch_shape**: Integer tuple. Size and shape of a patch.
- **use_multiprocessing**: Boolean. Uses multi-threading to prepare subfunctions if True (parallelized).

Returns:

A Preprocessor class object. Have to be passed to the Neural Network Model class.

Figure 4.13: Extract of the documentation entry for the Preprocessor class from the MIScnn wiki.

class processes the data into batches that are ready to be used for training, prediction, and validation. The user is only required to create an instance of the Preprocessor class with the desired specifications and Data IO instance (optional also a Data Augmentation instance). The docstring and argument descriptions of the Preprocessor class are summarized in Figure 4.13 and an example in Figure 4.14.

Example:

```
from processing.preprocessor import Preprocessor
from processing.subfunctions import Clipping, Normalization, Resampling

sf = [Clipping(min=-100, max=500), Normalization(z_score=True), Resampling((3.22, 1.62, 1.62))]

pp = Preprocessor(data_io, data_aug=None, batch_size=2, subfunctions=sf,
                 prepare_subfunctions=True, prepare_batches=False,
                 analysis="patchwise-crop", patch_shape=(80,160,160))

pp.patchwise_overlap = (40, 80, 80)
```

Figure 4.14: Example initialization of the Preprocessor class from the MIScnn wiki.

The Neural Network class is the last core class of MIScnn and functions as a hub to run the complete pipeline. It provides functionality for handling all model methods which includes running training, prediction, and evaluation processes. This core class inherits and uses all previously defined core classes. The docstring and argument descriptions of the Neural Network class are summarized in Figure 4.15.

```
Neural_Network(preprocessor, architecture=Architecture(),
               loss=tversky_loss, metrics=[dice_soft],
               learning_rate=0.0001, batch_queue_size=2,
               workers=1, multi_gpu=False)
```

Initialization function for creating a Neural Network (model) object.

Arguments:

- **preprocessor**: Preprocessor class instance which provides the Neural Network with batches.
- **architecture**: Instance of a neural network model Architecture class instance. By default, a standard U-Net is used as Architecture.
- **loss**: The Metric function which is used as loss for training.
- **metrics**: List of one or multiple Metric Functions, which will be shown during training.
- **learning_rate**: Learning rate in which weights of the neural network will be updated.
- **batch_queue_size**: The batch queue size is the number of previously prepared batches in the cache during runtime.
- **workers**: Number of workers/threads which preprocess batches during runtime.
- **multi_gpu**: Parameter which decides, if multiple gpus will be used for training (Distributed training).

Returns:

A Neural Network Model class object.

Example:

```
model = miscnn.Neural_Network(preprocessor=pp)
model.train(sample_list[0:100], epochs=50)
```



Figure 4.15: Extract of the documentation entry for the Neural Network class from the MIScnn wiki.

Basic Usage

In the following subchapter, a simple example demonstrates how to use MIScnn for building modern deep learning based MIS pipelines in less than 20 lines of code. As introduced in the subchapter Core Classes of Chapter 4.2.3, a MIScnn pipeline is built with the three mandatory

```
1 # Import the MIScnn module
2 import miscnn
3
4 # Create a Data I/O interface for kidney tumor CT scans in NIfTI format
5 from miscnn.data_loading.interfaces import NIFTI_interface
6 interface = NIFTI_interface(pattern="case_000[0-9]*", channels=1, classes=3)
7
8 # Initialize data path and create the Data I/O instance
9 data_path = "/home/mudomini/projects/KITS_challenge2019/kits19/data.original/"
10 data_io = miscnn.Data_IO(interface, data_path)
11
12 # Create a Preprocessor instance to configure how to preprocess the data into batches
13 pp = miscnn.Preprocessor(data_io, batch_size=4, analysis="patchwise-crop",
14                          patch_shape=(128,128,128))
15
16 # Create a deep Learning neural network model with a standard U-Net architecture
17 from miscnn.neural_network.architecture.unet.standard import Architecture
18 unet_standard = Architecture()
19 model = miscnn.Neural_Network(preprocessor=pp, architecture=unet_standard)
```

Code Snippet 4.2: Setup of a medical image segmentation pipeline with MIScnn.

core classes: Data IO, Preprocessor, and Neural Network (Data Augmentation as an optional core class). As demonstrated in Code Snippet 4.2, each core class is initialized and then passed to the next class. After initializing all core classes, the returning model object represents a fully functional MIS pipeline.

```

1 # Training the model with 80 samples for 50 epochs
2 sample_list = data_io.get_indiceslist()
3 model.train(sample_list[0:80], epochs=50)
4
5 # Predict the segmentation for remaining 20 samples
6 preds = model.predict(sample_list[80:100], return_output=True)

```

Code Snippet 4.3: MIScnn pipeline utilization for training and prediction processes.

With a MIScnn model, which represents a complete MIS pipeline, it is possible to run training and prediction processes. As shown in Code Snippet 4.3, the model is used for training on the first 80 samples from the dataset and, afterward, used for predicting the segmentation mask of the remaining 20 samples. This example reveals the simplicity of building and using a state-of-the-art MIS pipeline with MIScnn.

4.2.4 Community Contributions, Support, and Popularity

As previously stated, the proposed MIScnn framework was released in July 2019 as an open-source project for standardized medical image segmentation. Since then, it has been maintained, supported, and further developed. Also, MIScnn has been not only utilized in in-house studies which are discussed in the next chapters but has also become popular in the community of MIS. In order to empirically quantify the prevalence of the proposed framework, the GitHub repository metadata as well as the PyPI download statistics were evaluated.

In terms of further project development and community contributions, the number of issues, in which users reported errors, asked questions, or suggested feature requests, and the number of

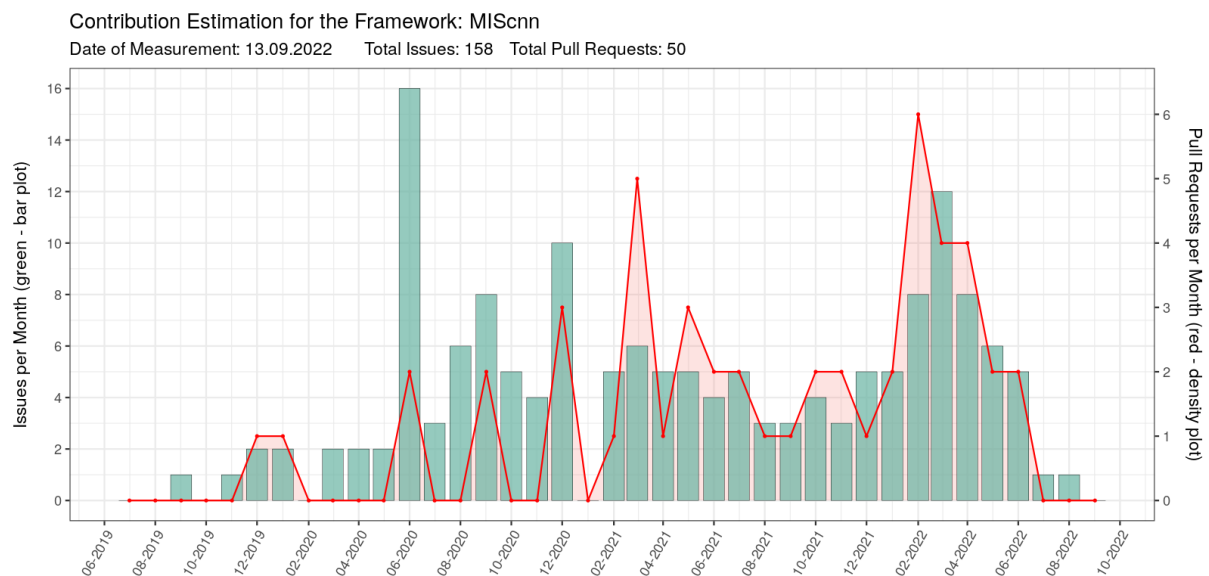


Figure 4.16: Community contribution estimation for the MIScnn framework.

pull requests, in which users directly contributed to the project, were analyzed and visualized in Figure 4.16. The data revealed a constant number of issues and pull requests per month implying the presence of an active community around the project. A peak of contributions occurred between February to April 2022. The highest number of created issues was in June 2020 with 16 issues per month. This indicates not only the required effort of maintaining an open-source project but also shows the community interest in the MIScnn project.

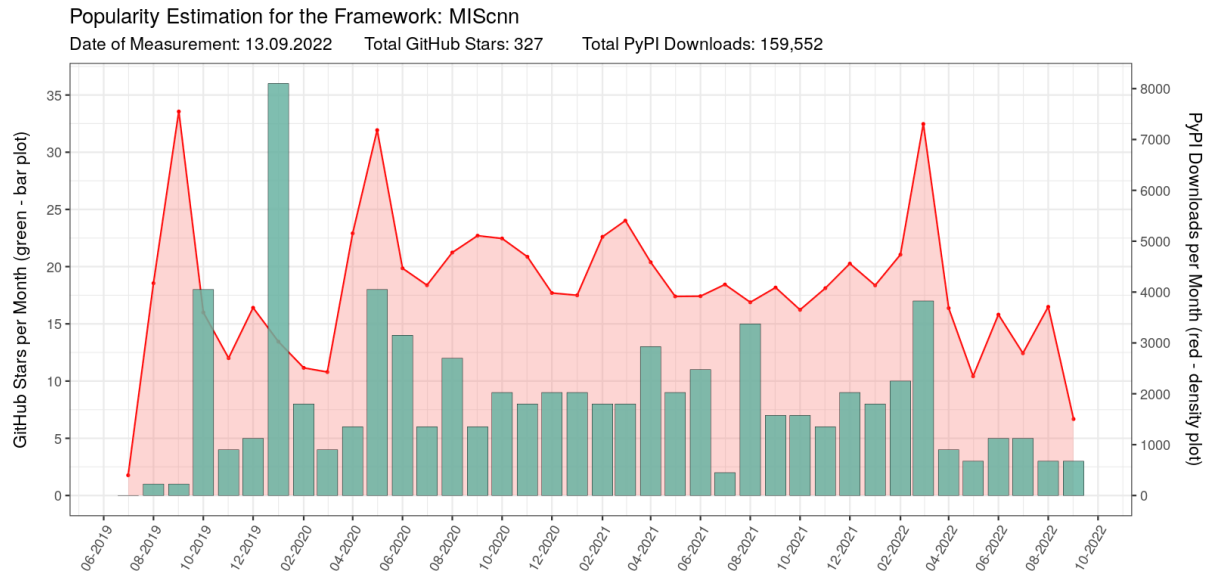


Figure 4.17: Popularity estimation for the MIScnn framework.

For popularity estimation of the framework and to approximate the number of users (community size), the GitHub Stars, which act as bookmarks in GitHub, and the number of package downloads from the PyPI statistics database (also called Linehaul project) [284] was analyzed and visualized in Figure 4.17. In contrast to the analyzed community contribution features, the number of downloads and GitHub Stars are a more direct indicator of community prevalence and quantity. Overall constant popularity with noticeable standard deviation in the monitored time period was observed. However, the last observed month indicates a slight reduction in traffic which correlates with the last published release in February 2022. Furthermore, the identified peak of issue creations in June 2020 also correlates with a peak in package downloads indicating a community growth of active and contributing users.

4.3 Study: Kidney Tumor Segmentation

In this subchapter, data from the Kidney Tumor Segmentation Challenge 2019 was analyzed and evaluated using MIScnn. The main idea for this experiment was to demonstrate the functionality and ‘out-of-the-box’ performance of MIScnn without thorough and time-consuming optimization on the dataset or on the medical abnormality.

4.3.1 Dataset

With more than 400,000 kidney cancer diagnoses worldwide in 2018, kidney cancer is among the top 10 most common cancer types in men and under the top 15 in women [285]. The development of advanced tumor visualization techniques is highly important for efficient surgical planning. Due to the variety in kidney and kidney tumor morphology, automatic image segmentation is challenging but of great interest [129].

The goal of the KiTS19 (kidney tumor segmentation 2019) challenge was the development of reliable and unbiased kidney and kidney tumor semantic segmentation methods [129]. Therefore, the challenge built a dataset for arterial phase abdominal CT scans of 300 kidney cancer patients [129]. The original scans have an image resolution of 512x512 pixels and on average 216 slices (highest slice number is 1059). For all CT scans, a ground truth semantic annotation was created by experts. This semantic annotation labeled each pixel with one of three classes: Background, kidney, or tumor. An example CT scan including its annotation is shown in Figure 4.18 in which the ground truth segmentation is visualized for both kidneys (red) and tumor (blue) [129]. 210 of the CT scans including annotations were published during the training phase of the challenge, whereas 90 CT scans without published ground truth were released afterward in the submission phase. The CT scans were provided in the NIfTI file format in original resolution and also in interpolated resolution with slice thickness normalization.

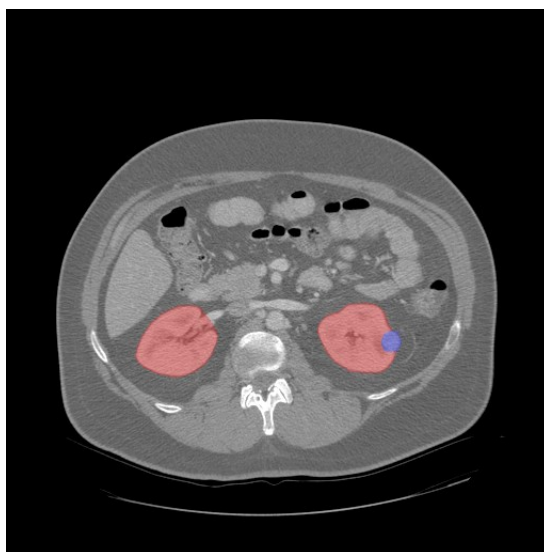


Figure 4.18: Computed tomography image from the KiTS19 dataset.

4.3.2 Application

In this subchapter, the MIScnn and its pipeline configuration are described. For the evaluation of the MIScnn framework usability, a subset of 120 CT scans with slice thickness normalization were retrieved from the KiTS19 training dataset (210 samples in total). An automatic 3-fold cross-validation was run on this KiTS19 subset with MIScnn. In order to reduce the overfitting risk, the cross-validation testing sets did not influence the fitting process and were not used for any automatic hyperparameter tuning.

MIScnn Configurations

The MIScnn pipeline was configured to perform a multi-class, patch-wise analysis with 80x160x160 pixel patches and a batch size of 2. Pixel value normalization was applied by Z-Score, clipping to the range -79 and +304 HU, as well as resampling to the voxel spacing 3.22x1.62x1.62 mm. For image augmentation, all implemented augmenting techniques were used. This includes creating patches through random cropping, scaling, rotations, elastic deformations, mirroring, brightness, contrast, gamma, and Gaussian noise augmentations.

The standard 3D U-Net [108, 117] with batch normalization layers was used as deep learning and convolutional neural network model. The training was performed using the Tversky loss [241] for 1,000 epochs with a starting learning rate of $1E^{-4}$ and batch shuffling after each epoch. Additionally, early stopping and learning rate reduction techniques were utilized based on training loss monitoring. For prediction, overlapping patches were created with an overlap size of 40x80x80 pixels in x, y, and z-axes.

Code Demonstration

In this subchapter, the required code to setup a MIScnn pipeline for model training is demonstrated. The first step in the MIScnn pipeline is to establish a data I/O. MIScnn offers the utilization of custom Data IO interfaces for fast integration of specific data structures into the pipeline. The KiTS19 CT scans were encoded in the NIfTI file format, therefore a Data IO class with the NIfTI interface for handling the NIfTI file format was deployed.

```
1 # Library import
2 from miscnn.data_loading.interfaces.nifti_io \
3     import NIFTI_interface
4 from miscnn.data_loading.data_io import Data_IO
5
6 # Initialize the NIfTI I/O interface and configure
7 # the images as 1 channel (grayscale) with 3 segmentation classes
8 interface = NIFTI_interface(pattern="case_00[0-9]*",
9                             channels=1, classes=3)
10
11 # Specify the kits19 data directory
12 data_path = "/storage/kits19/data/"
13 # Create the Data I/O object
14 data_io = Data_IO(interface, data_path)
```

Code Snippet 4.4: KiTS19 dataset loading in MIScnn.

After the Data IO Interface initialization, the Data Augmentation and Preprocessor class can be configured. A Preprocessor object with default parameters automatically initializes a Data Augmentation class with default values, but in this study, it was initialized by hand to illustrate the exact workflow of the MIScnn pipeline. The Data Augmentation is configured for using all possible augmentation techniques in order to run extensive image augmentation and avoid overfitting. Furthermore, a list of subfunctions was selected that were applied to the dataset. It is possible to add already provided subfunctions from MIScnn or implement custom subfunctions and pass these to the MIScnn pipeline. In this study, the following subfunctions were used: Pixel value normalization, clipping, and resampling.

Finally, the Data IO, the Data Augmentation, and the list of Subfunctions can be passed to create the Preprocessor. Additionally, the Preprocessor was configured to create batches by randomly cropping patches with a shape of 80x160x160 pixels out of the image and an overlap between patches.

```

1 # Library import
2 from miscnn.processing.data_augmentation import Data_Augmentation
3
4 # Create and configure the Data Augmentation class
5 data_aug = Data_Augmentation(cycles=2, scaling=True, rotations=True, elastic_deform=True,
6                               mirror=True, brightness=True, contrast=True, gamma=True,
7                               gaussian_noise=True)
8
9 # Library imports
10 from miscnn.processing.subfunctions.normalization import Normalization
11 from miscnn.processing.subfunctions.clipping import Clipping
12 from miscnn.processing.subfunctions.resampling import Resampling
13
14 # Create a pixel value normalization Subfunction through Z-Score
15 sf_normalize = Normalization(mode="z-score")
16 # Create a clipping Subfunction between -79 and 304
17 sf_clipping = Clipping(min=-79, max=304)
18 # Create a resampling Subfunction to voxel spacing 3.22 x 1.62 x 1.62
19 sf_resample = Resampling((3.22, 1.62, 1.62))
20
21 # Assemble Subfunction classes into a List
22 # Be aware that the Subfunctions will be executed according to the List order!
23 subfunctions = [sf_resample, sf_clipping, sf_normalize]
24
25 # Library import
26 from miscnn.processing.preprocessor import Preprocessor
27
28 # Create and configure the Preprocessor class
29 pp = Preprocessor(data_io, data_aug=data_aug, batch_size=2, subfunctions=subfunctions,
30                  prepare_subfunctions=True, prepare_batches=False,
31                  analysis="patchwise-crop", patch_shape=(80, 160, 160),
32                  use_multiprocessing=True)
33
34 # Adjust the patch overlap for predictions
35 pp.patchwise_overlap = (40, 80, 80)

```

Code Snippet 4.5: Initialization of the Preprocessor class for kidney tumor segmentation.

The Neural Network class is the hub of the MIScnn pipeline and allows running training and prediction operations. To show the simplicity and effectiveness of MIScnn, a simple 3D U-Net architecture was selected as neural network without integrating any further manual optimizations. As loss function for training, the Tversky loss was used.

```

1 # Library import
2 from miscnn.neural_network.model import Neural_Network
3 from miscnn.neural_network.metrics import dice_soft, tversky_loss
4
5 # Create the Neural Network model
6 model = Neural_Network(preprocessor=pp, loss=tversky_loss, metrics=[dice_soft],
7                        batch_queue_size=3, workers=3, learning_rate=0.0001)
    
```

Code Snippet 4.6: Initialization of the Neural_Network class for kidney tumor segmentation.

After the setup of the MIScnn pipeline by initializing the core classes, the MIA pipeline is functional, and its model can be fitted in a training process. Due to MIScnn being based on TensorFlow [72], it is possible to integrate powerful monitoring mechanics, called ‘callbacks’. Here, the learning rate reduction schedule and early stopping callback are utilized. For the training process, the automatic cross-validation by MIScnn was performed with 3 as the number of k-folds.

```

1 # Define Callbacks
2 from tensorflow.keras.callbacks import ReduceLRonPlateau
3 cb_lr = ReduceLRonPlateau(monitor='loss', factor=0.1, patience=20, verbose=1,
4                          mode='min', min_delta=0.0001, cooldown=1, min_lr=0.0001)
5 from tensorflow.keras.callbacks import EarlyStopping
6 cb_es = EarlyStopping(monitor='loss', min_delta=0, patience=150, verbose=1, mode='min')
7
8 # Create the sample ID list consisting of the first 120 samples
9 validation_samples = data_io.get_indiceslist()[0:120]
10
11 # Library import
12 from miscnn.evaluation.cross_validation import cross_validation
13 # Run cross-validation function
14 cross_validation(validation_samples, model, k_fold=3, epochs=1000, draw_figures=True,
15                callbacks=[cb_lr, cb_es])
    
```

Code Snippet 4.7: Performing a 3-fold cross-validation on the KiTS19 dataset with MIScnn.

The cross-validation functionality of MIScnn runs the training and prediction process k -times on with different combinations of subsets from the provided data.

Code Reproducibility

To ensure full reproducibility, the complete code of this study is publicly available.

The original Git repository: <https://github.com/muellerdo/kits19.MIScnn>.

The reworked and more effective Notebook (which was illustrated in Subchapter 4.3.2 Code Demonstration):

<https://github.com/frankkramer-lab/MIScnn/blob/master/examples/KiTS19.ipynb>.

All data generated and analyzed during this study is available in the Zenodo repository: <https://doi.org/10.5281/zenodo.3962097>.

4.3.3 Results and Discussion

The cross-validation was run on two Nvidia Quadro P6000 (24GB VRAM) and took 58 hours. With the MIScnn pipeline, it was possible to successfully set up a complete, working medical image multi-class segmentation pipeline. The evaluation of the 3-fold cross-validation of 120 CT scans for kidney and tumor segmentation was based on the class-wise Dice Similarity Coefficient and the normalized DSC (soft DSC) in which all class-wise DSCs are macro-averaged. The scores were computed during the training process, as well as in the prediction process with the final fitted model.

The fitted model achieved strong performance for kidney segmentation. It was possible to achieve a DSC median of around 0.9544 for kidney segmentation. The tumor segmentation prediction showed a considerably high but weaker performance than the kidney with a median of around 0.7912.

Table 4.3: Performance results of the 3-fold cross-validation for tumor and kidney segmentation.

Metric	Training	Validation
Tversky loss	0.3672	0.4609
Soft Dice Similarity Coefficient (soft DSC)	0.8776	0.8235
Categorical cross-entropy	- 0.8584	- 0.7899
Dice Similarity Coefficient: Background	X	0.9994
Dice Similarity Coefficient: Kidney	X	0.9319
Dice Similarity Coefficient: Tumor	X	0.6750

For each cross-validation fold, the training and prediction scores are visualized in Figure 4.19 and summed up in Table 4.3. In Table 4.3, each metric was computed between the provided ground truth mask and the proposed model prediction. Afterward, the scores were averaged between the three folds. Figure 4.19 is grouped into three subplots. The left plot shows a fitting



Figure 4.19: Fitting curves and performance assessment for the kidney tumor segmentation pipeline.

curve between the Tversky loss and the number of epochs illustrating the loss development during training for the training and testing set according to the cross-validation sampling. The center plot shows another fitting curve but with the soft DSC as metric. The right box plot visualizes the DSC distribution for the kidney and tumor for all 120 samples.

Besides the computed metrics, MIScnn created segmentation visualizations for manual comparison by eye between the ground truth and prediction mask. As illustrated in Figure 4.20, the predicted semantic segmentation of kidneys and tumors is highly accurate. In the figure, the kidney is indicated with a red overlay whereas the tumors with blue.

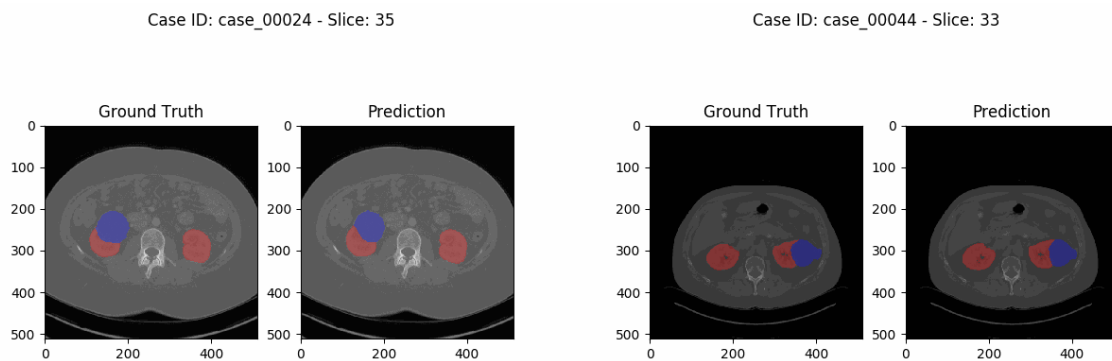


Figure 4.20: Visualization of annotated and predicted segmentation masks for kidney tumor segmentation.

Interpretation and Enhancements

The predictive power of the proposed neural network models was impressive in the context of using only the standard U-Net architecture with mostly default hyperparameters. From the medical perspective, through the variety in kidney tumor morphology, which is one of the reasons for the KiTS19 challenge, the weaker tumor results are quite reasonable [129]. Also, the models were trained with only 38% of the original KiTS19 dataset due to 80 images for training and 40 for testing being randomly selected. The remaining 90 CTs were excluded to reduce run time in the cross-validation. Additionally, fast switching the model to a more precise architecture for high-resolution images, like the Dense U-Net, would presumably result in an even better performance [113]. However, this gain would go hand-in-hand with an increased fitting time and higher GPU memory (VRAM) requirement, which was not possible with the Author's lab sharing schedule for GPU hardware. Still, the possibility of swift switching between model architectures to compare their performance on a dataset is a promising step forward in the field of MIS. Nevertheless, it was possible to build a powerful pipeline for kidney tumor segmentation with MIScnn resulting in a model with high performance, which is directly comparable with modern, optimized, standalone pipelines [108, 112, 117, 200, 262].

4.3.4 Conclusions

In order to show the reliability of MIScnn, a pipeline was setup for kidney tumor segmentation in CT imaging. For pipeline building, it was possible to effectively utilize state-of-the-art

methods like the U-Net architecture as deep learning model as well as on-the-fly image augmentation. This study proved that it was possible to successfully build a powerful pipeline for medical image segmentation with just a few lines of code using the MIScnn framework.

4.4 Study: Nucleus Segmentation and Analysis in Breast Cancer

Analyzing nuclei from CT slides is a difficult and repetitive task in biomedical image analysis. Therefore, more and more automated processes such as deep learning based MIS are coming to the fore. One of the main fields of application is pathology, which focuses on the detection of tumors and other abnormalities in the context of cell nuclei [286, 287]. A general problem with this type of automated analysis is the limited number of sufficiently well-annotated training data by experts. The large-scale NuCLS study by Amgad et al. [288] presents an efficient method that can be used to create and verify a large amount of training data from different CT scans of cell nuclei. This allows demonstrating that these annotated data could be used to build a standardized pipeline framework for the segmentation and classification of cell nuclei from breast tissue. In this study, the Author wants to show that repetitive and complex tasks in a special pathology field could be efficiently automated with a deep neural network pipeline.

4.4.1 Dataset

The raw images are provided by the Cancer Genome Atlas (TCGA) program, which is supervised by the National Cancer Institute [289]. Every slide is associated with a breast cancer patient and a scan of the breast tissue. The NuCLS dataset study [288] has split down the scans into 1 mm^2 picture tiles, which capture an ROI. They also provide for every tile a separated mask, which is annotated with 13 different nucleus classes. These different region classes could merge into three ‘super classes’ like tumor, stromal and sTILs (stromal tumor-infiltrating lymphocytes) because of performance and class imbalance reasons. The first channel of the mask encodes pixel-wise the label of the specific class on the tile. The instance label of each unique nucleus segmentation could be generated with the matrix product of the second and third channels. For object detection, the NuCLS dataset provided a separate file for each mask, in which every class is annotated by coordinates. The masks were created by 25 non-pathologists, which are guided by supervisors [288]. They labeled bounding boxes or complete segmentation masks for over 220,000 nuclei. This process results in two different quality datasets. The ‘single-rater’ dataset contains 1,744 samples, which are partially corrected and approved by the

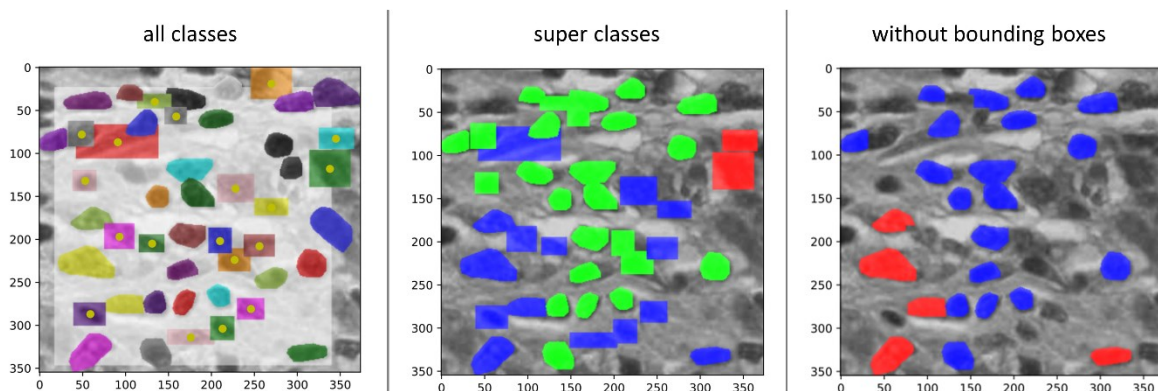


Figure 4.21: Annotation of the same sample represented through different masks.

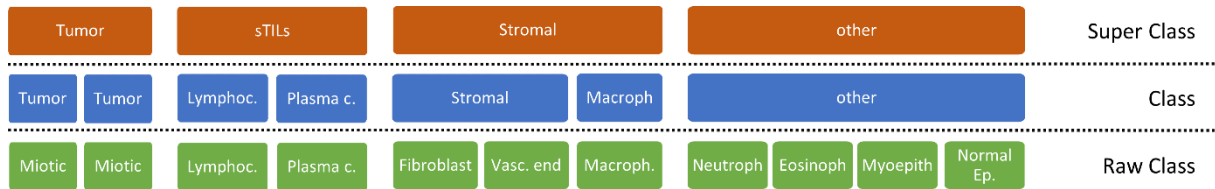


Figure 4.22: Overview of the class categories in the NuCLS dataset.

study coordinators. The ‘multi-rater’ dataset (also called ‘inferred P-truth’ by the authors) contains only 53 samples, which are annotated separately by 7 experienced pathologists. The different masks for a single sample are merged resulting in high-quality annotations.

Data Exploration

The tumor, stromal, and sTILs class distribution is roughly equal but indicates a strong class imbalance towards the super class ‘other’. All samples contain the FOV-class (field of view), which consists of any excluded classes that have not been reflected into a super class, or all areas that have not been annotated [288].

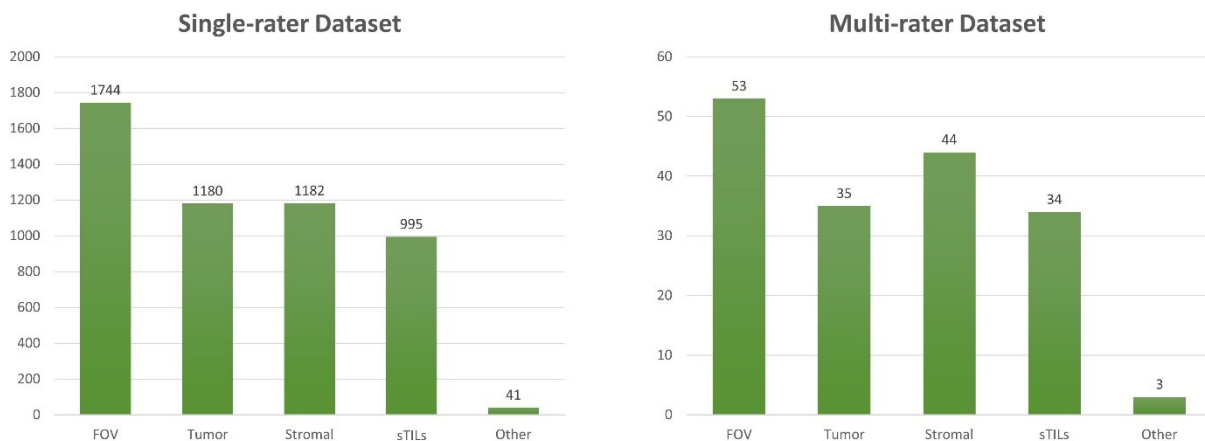


Figure 4.23: Class distribution of the super class annotations in the NuCLS dataset.

Before training, the data was analyzed with an exploration algorithm. These calculated the mean dimensions and the class distribution over all tiles. This information was used for the padding routine and the class imbalance correction later in the network configuration phase.

4.4.2 Application

In order to build a complete and efficient MIS pipeline, the framework MIScnn was utilized [49]. In the following two subchapters, the MIScnn and pipeline configuration are described. To ensure full reproducibility, the complete code of this study is available in a public Git repository: <https://github.com/Pfleiderer-Adrian/NuCLS.MIScnn>.

Preprocessing

For the preprocessor phase, a customized Data IO Interface for the MIScnn pipeline was created

which handles the correct data input from NuCLS images and masks. Some masks and tiles have different dimensions, therefore, the oversized image is cropped and matched to the mask. The functionality to optionally identify and remove bounding boxes with the corresponding localization file has also been implemented. Finally, the mask and their raw classes were transformed into super classes that delivered more samples for each segmentation class.

Three different input datasets were created for the network. The first contains only the single-rater dataset. The second set contains also the single-rater dataset but all bounding boxes are eliminated. Both sets were sampled with a percentage split into a 60-20-20 ratio (training-validation-testing). The last set is a combination of the original single-rater dataset and the multi-rater dataset, whereby the complete multi-rater dataset was used for the testing phase. The percentage split procedure without cross-validation was selected for sampling due to hardware limitations and large enough set sizes to ensure robust fitting.

Additionally, the MIScnn image augmentation functionality was used to increase data variance. Besides color changes through contrast, brightness, gamma, and gaussian noise, spatial transformations like scaling, elastic deform mirroring, and rotation were applied. Tiles and masks adapted to the filter kernel were required for neural network input. During the data exploration, the average size of all tiles was calculated. These were used to determine the padding parameter, which is the nearest value in the power of 2. This was also used as resize shape which was applied for images that were larger than the target shape. In the context of the single-rater dataset, a maximum resolution of 796x830 pixels was detected, resulting in a padding and resize shape of 768x768 pixels. Finally, each input batch was normalized via Z-Score, which computes the distance between the pixel value and the distribution mean with the unit of the standard deviation.

Neural Network Model

The neural network is based on the effective standard U-Net architecture implementation from MIScnn. The standard U-Net architecture was already successfully tested in many publications around the MIScnn framework [110, 290, 291] and is widely used in the image segmentation domain [13].

A Tversky variant of the reference function from Salehi et al. [241] was used, in which the authors were able to achieve strong training improvements with 3D convolutional deep networks. To further strengthen the function, the standard cross-entropy was calculated. Finally, both loss functions were combined by sum. Furthermore, the super class ‘other’ was excluded from training and evaluation in order to ensure result comparability with the NuCLS Study which also excluded this outlier class [288]. A learning rate starting value of 0.001 was used and decreased by a factor of 0.1 if the model reaches a plateau during training. A minimal rate learning rate of $1E^{-5}$ was defined so that the learning process remains effective. The validation loss was monitored with a loss change threshold of $1E^{-4}$ for the learning rate reduction schedule. Additionally, the pipeline included early stopping for training. If the model did not improve on validation loss, the training process has been stopped after 10 epochs.

For a setup with an NVIDIA TITAN RTX (24 GB VRAM) and the previously specified target shape, it was possible to use a batch size range between 4 and 10. After internal experiments, an optimal batch size of 8 was selected. For custom modifications, it is recommended to utilize a batch size that can be divided by 2, because the number of physical processors in GPUs often has a power of 2 distribution, which enables optimal pluralism [292].

4.4.3 Results and Discussion

For performance assessment and evaluation, the Author's metric framework, MISEval (Chapter 7.1), was utilized [293]. This allowed robust and simple evaluation out of the box by the Dice Similarity Coefficient, Accuracy, and Matthews Correlation Coefficient (MCC).

The data exploration revealed that the datasets contain many inaccurate bounding boxes which introduced a noticeable bias in the evaluation. Another issue identified was the rare classes and the super class 'other'. To avoid extreme class imbalances and comparability issues with the original NuCLS study, these classes did not get a separate super class and therefore are integrated into the FOV class. Further, the samples were taken from different device types in different clinical environments [288]. An internal analysis indicated a variance in annotation quality and device artifacts in the samples. In the worst case, the model would learn these peculiarities from the different recording devices resulting in a bias-affected model and evaluation. The NuCLS study even reinforced this issue by sorting the samples according to the devices into separate cross-validation folds. This approach is bypassed in this study by using a percentage split and shuffling the complete dataset.

Three models were created for the three different datasets. After around 40 - 60 epochs, the training progress began to converge without any more significant loss improvement. After 100 - 150 epochs the early stopping callback exited the training process. The individual results of the three model evaluations are summarized and interpreted in the following sections. All scores have been rounded to one decimal place and are normalized between the range 0 to 100.

Evaluation: Single-rater Dataset

The single-rater dataset balances the positive effect of a high number of samples and the negative effect of lower-quality as well as error-prone annotations. For evaluation, 347 samples were used to compute performance capabilities. The stromal class did not perform as high in

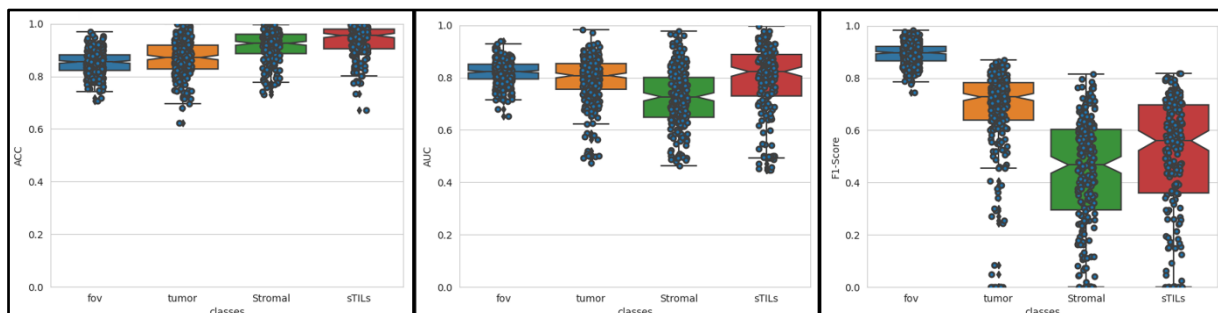


Figure 4.24: Box plots showing the result distribution for the single-rater dataset.

comparison to the other classes. Tumor and sTILs were able to achieve relatively better results if the outliers with a zero score were disregarded.

Table 4.4: Achieved macro-averaged results on the single-rater dataset.

Class	MCC	IoU	ACC	AUC	DSC
FOV	64.7	81.3	85.5	82.2	89.7
Tumor	63.4	57.1	87.1	80.8	72.7
Stromal	44.7	30.5	92.5	72.6	46.8
sTILs	54.3	38.9	95.5	82.3	56.0
AVG	55.3	54.3	88.6	78.6	66.4

Evaluation: Single-rater Dataset & Multi-rater Dataset

This model was trained and validated with 1,744 single-rater samples and evaluated with the multi-rater dataset containing 53 samples with high-quality annotations. The model performed in all categories similarly to slightly better than the exclusive single-rater dataset approach. This demonstrated that poorly annotated samples can be compensated by utilizing a higher number of samples as well as a state-of-the-art MIS pipeline setup like MIScnn.

Table 4.5: Achieved macro-averaged results of the multi-rater dataset.

Class	MCC	IoU	ACC	AUC	DSC
FOV	58.8	82.8	85.3	80.0	90.6
Tumor	60.3	55.8	90.9	79.5	71.6
Stromal	41.2	30.5	89.8	71.2	46.7
sTILs	58.4	42.9	96.3	78.9	60.0
AVG	53.0	53.3	89.7	77.2	65.7

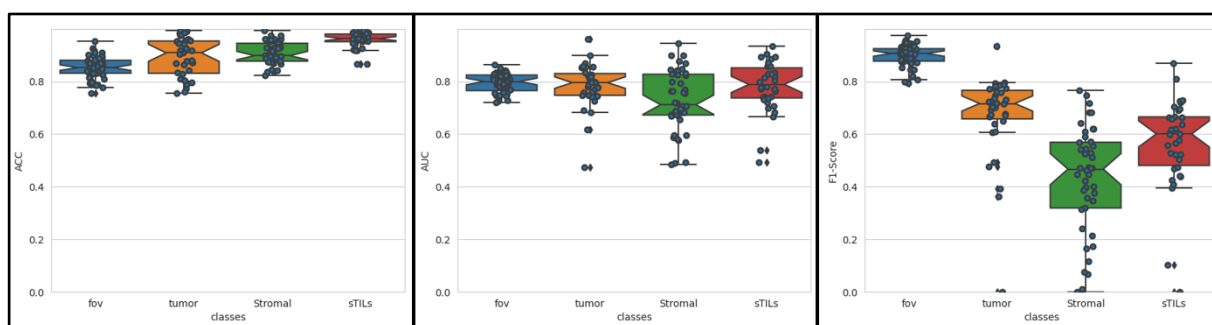


Figure 4.25: Box plots showing the result distribution for the multi-rater dataset.

Evaluation: Single-rater Dataset without Bounding Boxes

All bounding boxes were removed without a replacement for the training process. The idea behind this approach was to strengthen the annotations with high quality by excluding noise introducing bounding boxes. Unfortunately, the bounding boxes made up too much of the

annotations. This resulted in a model that could hardly classify any data. The approach was therefore not effective, and inferior compared to the other two experiment settings.

Comparison with Related Work

For benchmarking, the NuCLS study provided a custom-designed neural network model as well as a model trained with Mask R-CNN [288, 294]. Both models were trained, validated, and evaluated with the single-rater dataset. Thus, the achieved single-rater dataset results from the proposed pipeline using MIScnn offer the highest comparability. All results are median-averaged in order to be consistent with the results from the NuCLS study.

Table 4.6: Comparison of nuclei segmentation performance between NuCLS, Mask R-CNN, and MIScnn.

Class	NuCLS Model			Mask R-CNN			Achieved Results (MIScnn)			
	MCC	ACC	AUC	MCC	ACC	DSC	MCC	ACC	AUC	DSC
Tumor	71.7	-	94.2	63.0	83.2	74.9	63.3	87.1	80.8	72.7
Stromal	49.4	-	85.4	26.3	82.2	17.5	44.7	92.5	72.6	46.8
sTILs	73.5	-	94.7	58.1	75.5	77.4	54.3	95.4	82.2	56.1
other	-	-	-	12.0	97.9	11.9	-	-	-	-
Overall	65.6	77.7	86.0	52.7	69.1	-	55.3	88.6	78.6	66.4

The NuCLS Mask R-CNN results are slightly worse or equal to MIScnn. However, the NuCLS custom model outperforms both predictors. Nevertheless, it is important to note that the NuCLS model used k-fold cross-validation, which was sorted by device and recording sessions. It is plausible to expect that the device-grouped samples share similar unintentional features or artifacts that could have a positive impact on the evaluation. Still, the customized model architecture from the NuCLS study proved high-performant and efficient nuclei segmentation in comparison to the standard U-Net.

Limitations

The proposed approach to eliminate all bounding boxes in the single-rater dataset did not lead to satisfying results. A solution might be to not eliminate the bounding boxes completely, but instead to only adjust their weight per instance. Currently, MIScnn is not sufficiently adapted to integrate instance-based weighting of loss functions. Only class weights can be applied globally across all samples. An improved technique could be to compute weights based on samples or even coordinates in the mask. Such weighting technique would offer a beneficial advancement to the MIScnn features. Nevertheless, a more precise annotation, especially in the single-rater dataset, would be beneficial. The hybrid annotation types of bounding boxes and ROI masks have a negative impact on evaluation which should not be underestimated.

In order to further strengthen the proposed model, specific adjustments to the pipeline are needed. Possible improvements are the usage of class probability vectors or increasing the density of region proposals, which are both present in the NuCLS study [288]. Another

important aspect is image augmentation, which was performed with default configurations. Reasonable adjustments and optimizations to the imaging features of a cell nucleus can be further adapted. Open to discussion would be the alignment towards the NuCLS sampling by sorting according to the recording device. Furthermore, integrating the custom NuCLS neural network architecture in MIScnn would be an interesting enhancement that allows further increased comparability.

4.4.4 Conclusions

The study showed that with a standard U-Net architecture based MIScnn pipeline, it was possible to achieve adequate results in multi-class segmentation of cell nuclei. The MIScnn framework allows the efficient building of complex MIS pipelines which allowed broad experimentation and application on noise-introduced datasets. However, the created models cannot fully reach the level of a customized and adapted approach such as the NuCLS model. Still, after further extensive adjustments to the proposed pipeline, equal performance results to the NuCLS study should be reached, especially with regard to the device-based sampling. Concluding, it was demonstrated that MIScnn provides mature and dynamic templates for tackling a wide range of complex image segmentation problems.

4.5 Study: Segmentation of COVID-19 Lung Infection based on limited Data

Extensive clinical experience and time-consuming manual assessment are obligatory for reliable diagnostics with medical images. However, the coronavirus pandemic disclosed that these resources are limited resulting in inefficient clinical workflows and a decline in patient care quality [23, 295–297]. A solution for this challenge could be clinical decision support systems based on medical image segmentation for automated identification and labeling of regions of interest e.g. organs like lungs or medical abnormalities like cancer and lesions. It would be a helpful tool to implement such an automatic segmentation for novel diseases like COVID-19 infected regions as clinical decision support for physicians. By automatic highlighting abnormal features and ROIs, image segmentation is able to aid radiologists in diagnosis, disease course monitoring, reduction of time-consuming inspection processes and improvement of accuracy [13, 296, 298]. Nevertheless, training accurate and robust models requires sufficient annotated medical imaging data. Because manual annotation is labor-intensive, time-consuming, and requires experienced radiologists, it is common that publicly available data is limited [13, 23, 296]. This lack of data often results in an overfitting of the traditional data-hungry models. Especially for COVID-19, large enough medical imaging datasets are currently unavailable [23, 296].

In this study, the Author pushed towards creating an accurate and state-of-the-art MIS pipeline for COVID-19 lung infection segmentation, which is capable of being trained on small datasets consisting of 3D CT volumes. In order to avoid overfitting, extensive on-the-fly image augmentation as well as diverse preprocessing methods were exploited. To further reduce the risk of overfitting, the standard U-Net architecture instead of other more computational complex variants, like the residual architecture of the U-Net, was implemented. Furthermore, a sensitivity analysis with k-fold cross-validation for reliable performance evaluation was used.

4.5.1 Dataset

The ongoing coronavirus pandemic has spread to 199 countries in the world [299]. The World Health Organization (WHO) declared the outbreak as a “Public Health Emergency of International Concern” on the 30th of January 2020 and as a pandemic on the 11th of March 2020 [295, 300]. Because of the rapid spread of severe respiratory syndrome coronavirus 2 (SARS-CoV-2), billions of lives around the world were changed. A SARS-CoV-2 infection can lead to severe pneumonia with potentially fatal outcome [295, 301, 302]. Additionally, the rapid increase of confirmed cases and the resulting estimated basic reproduction numbers show that SARS-CoV-2 is highly contagious [301, 303, 304]. Through a combined international effort, multiple vaccines were rapidly developed, and various countries already hosted large vaccine campaigns. The WHO named this new disease “coronavirus disease 2019”, short form: COVID-19.

An alternative solution to the established reverse transcription polymerase chain reaction (RT-PCR) as standard approach for COVID-19 screening or monitoring is medical imaging like X-ray or computed tomography. The medical imaging technology has made significant progress in recent years and is now a commonly used method for diagnosis, as well as for quantification assessment of numerous diseases [13, 296, 305]. Particularly, thorax (chest) CT screening has emerged as a routine diagnostic tool for pneumonia. Therefore, chest CT imaging has also been strongly recommended for COVID-19 diagnosis and follow-up [306]. In addition, CT imaging is playing an important role in COVID-19 quantification assessment, as well as disease monitoring. COVID-19 infected areas are distinguishable on CT images by ground-glass opacity (GGO) in the early infection stage and by pulmonary consolidation in the late infection stage [303, 306, 307]. In comparison to RT-PCR, several studies showed that CT is more sensitive and effective for COVID-19 screening, and that chest CT imaging is more sensitive for COVID-19 testing even without the occurrence of clinical symptoms [296, 306–308]. Notably, a large clinical study with 1,014 patients in Wuhan (China) [306] determined that chest CT analysis can achieve 0.97 Sensitivity, 0.25 Specificity, and 0.68 Accuracy for COVID-19 detection.

In this study, two public datasets were used: Ma et al. [102] as limited dataset for model training as well as validation, and An et al. [309] as a larger hold-out dataset for additional testing purpose. An illustration of COVID-19 infected regions on a CT scan can be seen in Figure 4.26. The left image of the figure is the unsegmented CT scan, whereas the right image shows the segmentation of lungs (blue) and infection (red). The infected regions are distinguishable by GGOs and pulmonary consolidation in the lung regions. The image was obtained from the analyzed CT dataset from Ma et al. [45].

The Ma et al. dataset consists of 20 annotated COVID-19 chest CT volumes [23, 102]. All cases were confirmed COVID-19 infections with a lung infection proportion ranging from 0.01% to 59% [23]. This dataset was one of the first publicly available 3D volume sets with annotated COVID-19 infection segmentation [23]. The CT scans were collected from the Coronacases Initiative and Radiopaedia and were licensed under CC BY-NC-SA. Each CT volume was first labeled by junior annotators, then refined by two radiologists with 5 years of experience and

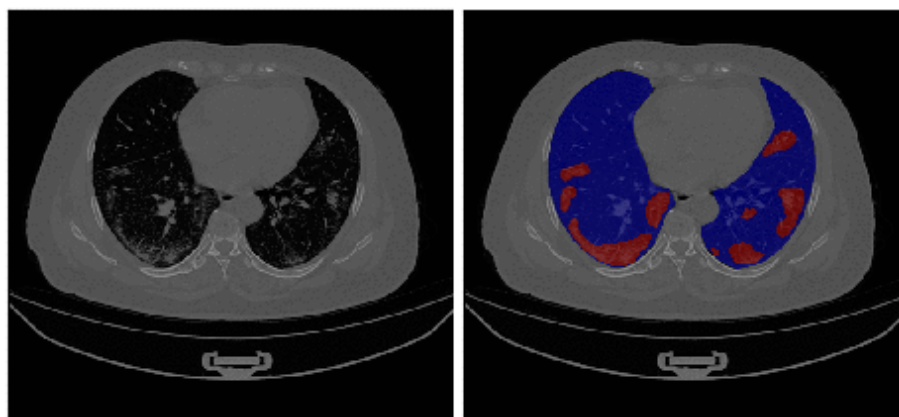


Figure 4.26: Visualization of COVID-19 infected regions in a thorax CT.

afterward the annotations verified by senior radiologists with more than 10 years of experience [23]. Despite the fact that the sample size is rather small, the annotation process led to an excellent high-quality dataset. The volumes had a resolution of 512x512 pixels (Coronacases Initiative) or 630x630 pixels (Radiopaedia) with a number of slices of about 176 by mean (200 by median). The CT images were labeled into four classes: Background, lung left, lung right, and COVID-19 infection.

The An et al. dataset consists of unenhanced chest CT volumes from 632 patients with COVID-19 infections and is one of the largest publicly available COVID-19 CT datasets [309]. The CT scans were collected through the outbreak settings from patients with a combination of symptoms, exposure to an infected patient, or travel history to an outbreak region [206, 309]. All patients had a positive RT-PCR for SARS-CoV-2 from a sample obtained within 1 day of the initial CT [206, 309]. The annotation of the dataset was made possible through the joint work of Children's National Hospital, NVIDIA, and National Institutes of Health for the COVID-19-20 Lung CT Lesion Segmentation Grand Challenge [172]. The challenge authors were able to annotate a subset of 295 patients through American board certified radiologists [172]. Through the characteristic as a challenge, not all volumes had publicly available annotations. Nevertheless, it was possible to obtain a subset of 100 patients as additional testing set. The volumes had a resolution of 512x512 pixels with a number of slices of about 75 by mean (65 by median). The CT images were labeled into two classes: Background and COVID-19 infection.

4.5.2 Related Work

In the middle of the year 2020, clinicians started to publish COVID-19 CT images with annotated ROIs, which allowed the training of segmentation models. Automated segmentation is highly desired as COVID-19 application [296, 310]. The segmentation of lung, lung lobes, and lung infection provide accurate quantification data for progression assessment in follow-up, comprehensive prediction of severity in the enrollment and visualization of lesion distribution using the percentage of infection (POI) [296]. Still, the limited amount of annotated imaging data causes a challenging task for detecting the variety of shapes, textures, and localizations of lesions or nodules. Nonetheless, multiple approaches try to solve these problems with different methods. The most popular network models for COVID-19 segmentation are variants of the U-Net which achieved reasonable performance on sufficiently sized 2D datasets [184, 188, 190, 296, 302, 311–315]. In order to compensate limited dataset sizes, more attention has been drawn to semi-supervised learning pipelines [296, 316, 317]. These methods optimize a supervised training on labeled data along with an unsupervised training on unlabeled data. Another approach is the development of special neural network architectures for handling limited dataset sizes. Frequently, attention mechanisms are built into the classic U-Net architecture like the Inf-Net from Fan et al. [316] or the MiniSeg from Qiu et al. [318]. Wang et al. [319] utilized transfer learning strategies based on models trained on non-COVID-19 related conditions. Particularly worth mentioning is the development of a

benchmark model with a 3D U-Net from Ma et al. [23, 102], because the authors also provide high reproducibility through a publicly available dataset.

4.5.3 Application

In order to setup a complete and state-of-the-art MIS pipeline, MIScnn was utilized [49]. The structure of the implemented MIS pipeline is illustrated in Figure 4.27. The workflow is starting with the COVID-19 dataset and is ending with the computed evaluation results for each fold in the cross-validation.

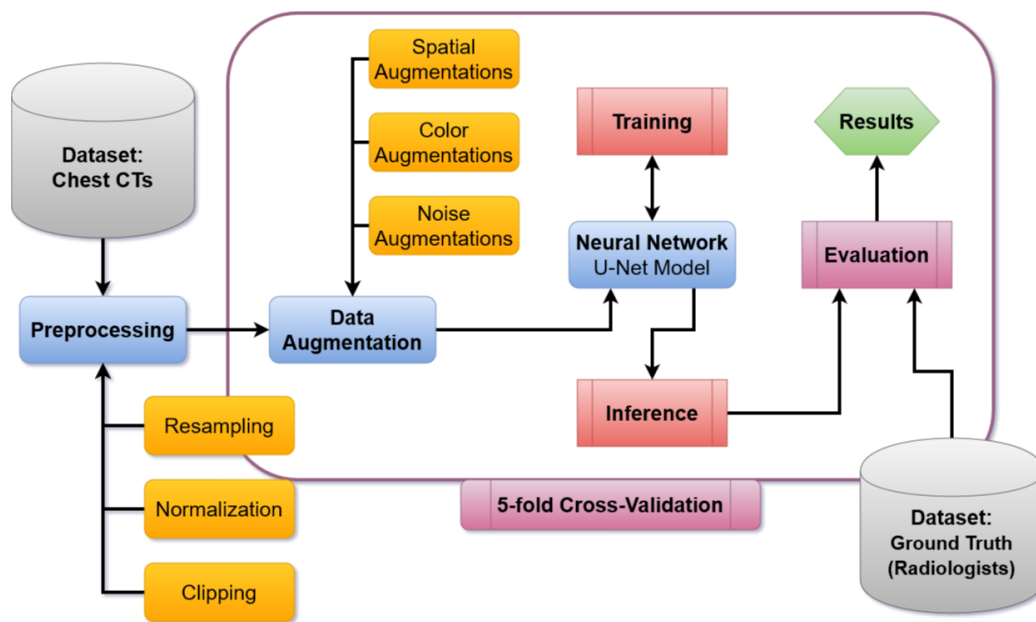


Figure 4.27: Flowchart diagram of the COVID-19 lung infection segmentation pipeline.

Preprocessing

In order to simplify the pattern finding and fitting process for the model, several preprocessing methods were applied to the datasets.

The Hounsfield Units scale was exploited by clipping the pixel intensity values of the images to -1,250 as minimum and +250 as maximum, because infected regions (+50 to +100 HU) and lung regions (-1,000 to -700 HU) were the points of interest [121]. It was possible to apply the clipping approach to the Coronacases Initiative and An et al. CTs because the Radiopaedia volumes were already normalized to a grayscale range between 0 and 255.

Varying signal intensity ranges of images can drastically influence the fitting process and the resulting performance of segmentation models [320]. For achieving dynamic signal intensity range consistency, it is recommended to scale and standardize imaging data. Therefore, the remaining CT volumes were normalized likewise to the grayscale range. Afterward, all samples were standardized via Z-Score.

Medical imaging volumes have commonly inhomogeneous voxel spacings. The interpretation of diverse voxel spacings is a challenging task for deep neural networks. Therefore, it is possible to drastically reduce complexity by resampling volumes in an imaging dataset to homogeneous voxel spacing, which is also called target spacing. Resampling voxel spacings also directly resizes the volume shape and determines the contextual information, which the neural network model is able to capture. As a result, the target spacing has a huge impact on the final model performance. All CT volumes were resampled to a target spacing of 1.58x1.58x2.70 mm, resulting in a median volume shape of 267x254x104 pixels.

Data Augmentation

The aim of data augmentation is to create more data of reasonable variations of the desired pattern and, thus, artificially increase the number of training images. This technique results in improved model performance and robustness [163, 165, 166]. In order to compensate the small dataset size, extensive data augmentation was performed by using the batchgenerators interface within MIScnn. Three types of augmentations were utilized: Spatial augmentation by mirroring, elastic deformations, rotations, and scaling. Color augmentations by brightness, contrast, and gamma alterations. Noise augmentations by adding Gaussian noise. Furthermore, each augmentation method had a random probability of 15% to be applied to the current image with random intensity or parameters (e.g. random angle for rotation) [118, 142].

Instead of traditional upsampling approaches, on-the-fly data augmentation was performed on each image before it was forwarded into the neural network model. Through this technique, the probability that the model encounters the exact same image twice during the training process decreases significantly, which proved to reduce the risk of overfitting drastically [118].

Patch-wise Analysis

In image analysis, there are three popular methods: The analysis of full images, the slice-wise analysis for 3D data or patch-wise by slicing the volume into smaller cuboid patches [13]. The patch-wise approach was selected in order to exploit random cropping for the fitting process. Through random forwarding of only a single cropped patch from the image to the fitting process, another type of data augmentation is induced, and the risk of overfitting is additionally decreased. Furthermore, full image analysis requires unnecessary resolution reduction of the 3D volumes in order to handle the enormous GPU memory requirements. By slicing the volumes into patches with a shape of 160x160x80 pixels, it was possible to utilize high-resolution data.

For inference, the volumes were sliced into patches according to a grid. Between the patches, an overlap of half the patch size (80x80x40 pixels) was introduced to increase prediction performance. After the inference of each patch, they were reassembled into the original volume shape, whereas overlapping regions were averaged.

Neural Network Model

In this study, the standard 3D U-Net was implemented as architecture without any custom modification in order to avoid unnecessary parameter increase by more complex architectures like the residual variant of the 3D U-Net [108, 112, 117]. The input of the architecture was a $160 \times 160 \times 80$ pixels patch with a single channel consisting of normalized HUs. The output layer of the architecture normalized the class probabilities through a softmax function and returned the $160 \times 160 \times 80$ pixels mask with four channels representing the probability for each class (background, lung left, lung right, and COVID-19 infection). Upsampling was achieved via transposed convolution and downsampling via maximum pooling. The architecture used 32 feature maps at its highest resolution and 512 at its lowest. All convolutions were applied with a kernel size of $3 \times 3 \times 3$ in a stride of $1 \times 1 \times 1$, except for up- and downsampling convolutions which were applied with a kernel size of $2 \times 2 \times 2$ in a stride of $2 \times 2 \times 2$. After each convolutional block, batch normalization was applied. The architecture can be seen in Figure 4.28. The figure shows that the network takes a 3D patch (cuboid) and outputs the segmentation of lungs and infected regions by COVID-19. Skip connections were implemented with concatenation layers. The figure contains the following abbreviations: Conv - Convolutional layer; ReLU - Rectified linear unit layer; BN - Batch normalization.

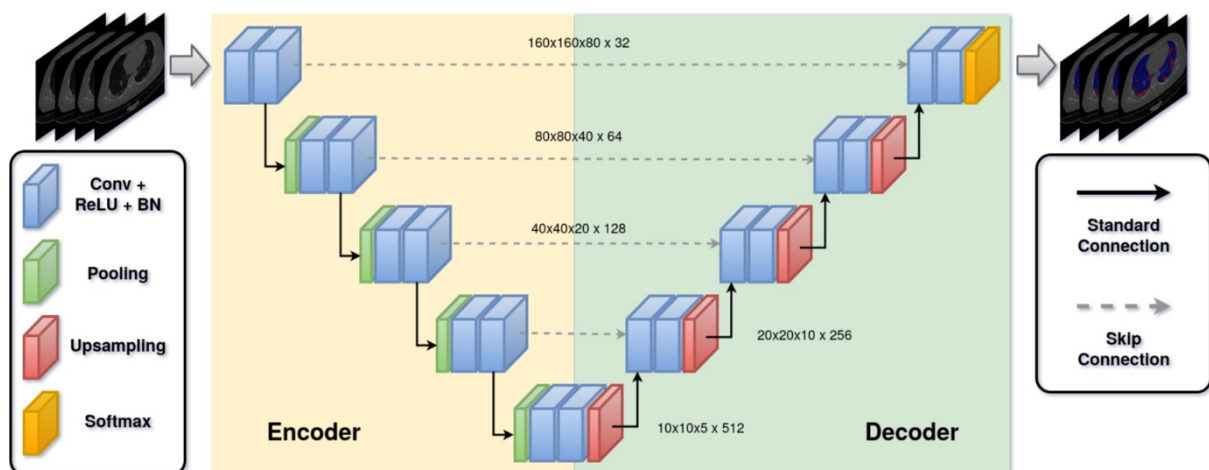


Figure 4.28: Structure diagram of the standard 3D U-Net architecture.

In medical image segmentation, it is common that semantic annotation includes a strong bias in class distribution towards the background class. The Ma et al. dataset revealed a class distribution of more than 89% for background, 9% for lungs, and 1% for infection. In order to compensate this class bias, the sum of the Tversky index [241] and the categorical cross-entropy was utilized as loss function for model fitting. For model fitting, an Adam optimization [201] was used with the initial weight decay of $1E^{-3}$. A dynamic learning rate was integrated which reduced the learning rate by a factor of 0.1 in case the training loss did not decrease for 15 epochs. The minimal learning rate was set to $1E^{-5}$. In order to further reduce the risk of overfitting, the early stopping technique was exploited for training, in which the training process stopped without a fitting loss decrease after 100 epochs. The neural network model was trained for a maximum of 1,000 epochs. Instead of the common epoch definition as a single

iteration over the dataset, an epoch was defined as the iteration over 150 training batches. This allowed for an improved fitting process for randomly generated batches in which the dataset acts as a variation database. According to the Author's available hardware resources (GPU VRAM), a batch size of 2 was selected.

Sensitivity Analysis with Cross-Validation

For reliable robustness evaluation, a sensitivity analysis was performed to estimate the generalizability and sensitivity of the pipeline. Thus, multiple k-fold cross-validations were performed on the Ma et al. dataset to obtain various models based on limited training data as well as different validation subsets.

As a k-fold multitude, a range from 2 up to 5 was used for the sensitivity analysis resulting in 4 separate cross-validation analyses with in total 14 models. Each model was created through a training process on $k-1$ folds and validated through the leftover fold in each cross-validation sampling. Training and validation were performed on the small Ma et al. dataset, whereas the An et al. dataset was used as additional testing set to further ensure a robust evaluation. For example, this technique resulted in the following sampling for a 5-fold cross-validation: 16 samples as training dataset (Ma et al.), 4 samples as validation dataset (Ma et al.), and 100 samples as testing dataset (An et al.).

Furthermore, the impact of the preprocessing and data augmentation techniques on model performances was analyzed for the 5-fold cross-validation. No hyperparameters were configured afterward on basis of validation results and no validation monitoring based training techniques were used, which allowed utilizing the validation results for hold-out evaluation, as well.

Code Reproducibility

In order to ensure full reproducibility and to create a base for further research, the complete code of this study, including extensive documentation, is available in a public Git repository: <https://github.com/frankkramer-lab/covid19.MIScnn>.

All data generated and analyzed during this study is available in the Zenodo repository, <https://doi.org/10.5281/zenodo.3902293>.

4.5.4 Results and Discussion

The sequential training of the complete cross-validation on 2 NVIDIA QUADRO RTX 6000 with 24GB VRAM, an Intel Xeon Gold 5220R using 4 CPUs and 20GB RAM took around 182 hours. All models did not require the entire 1,000 epochs for training and instead were early stopped after an average of 312 epochs.

Evaluation

During the fitting process, the segmentation performance was computed for each epoch on randomly cropped and data augmented patches from the validation dataset. This allowed for an evaluation of the overfitting on the training data.

After the training, four widely popular evaluation metrics were used for performance measurement on the validation and testing set: Dice Similarity Coefficient, Intersection-over-Union, Sensitivity, and Specificity. The performance measurement was based on the segmentation overlap between prediction and ground truth, which was manually annotated through the consensus of multiple radiologists, as described in the dataset subchapter. For the Ma et al. dataset, the two lung classes ('lung left' and 'lung right') were averaged by mean into a single class ('lungs') during the evaluation.

Table 4.7: Achieved results of the sensitivity analysis for COVID-19 segmentation.

k-fold CV	Dataset: Ma et al.								Dataset: An et al.			
	Lungs				COVID-19 Lesion				COVID-19 Lesion			
	DSC	IoU	Sens	Spec	DSC	IoU	Sens	Spec	DSC	IoU	Sens	Spec
k=2	0.960±0.06	0.923±0.10	0.970	0.998	0.775±0.20	0.635±0.19	0.747	0.999	0.555±0.07	0.386±0.07	0.485	0.998
k=3	0.966±0.07	0.934±0.10	0.968	0.999	0.778±0.19	0.636±0.18	0.730	0.999	0.598±0.10	0.426±0.11	0.580	0.999
k=4	0.951±0.22	0.907±0.29	0.948	0.999	0.711±0.27	0.552±0.25	0.731	0.999	0.661±0.07	0.494±0.09	0.561	0.999
k=5	0.971±0.07	0.944±0.11	0.971	0.999	0.804±0.20	0.672±0.19	0.778	0.999	0.623±0.04	0.453±0.04	0.513	0.998

After the training, the inference revealed a strong segmentation performance for lungs and COVID-19 infected regions. Overall, the k-fold cross-validation models achieved a DSC and IoU of around 0.971 and 0.944 for lungs, as well as 0.804 and 0.672 for COVID-19 infection segmentation on the Ma et al. dataset, respectively. On the additional testing set from An et al., the models achieved a DSC of around 0.661 and an IoU of around 0.494 for COVID-19 infection segmentation. Furthermore, the models obtained a Sensitivity and Specificity of 0.778 and 0.999 on the validation set, as well as 0.580 and 0.999 on the testing set for COVID-19 infection, respectively. More details on inference performance are listed in Table 4.7 and visualized in Figure 4.29. Table 4.7 groups the scores according to the median DSC, the IoU the Sensitivity and Specificity of Lung and COVID-19 infection segmentation for each k-fold cross-validation of the sensitivity analysis for the Ma et al. and An et al. dataset. Standard deviation is included for DSC and IoU. Figure 4.29 summarizes the DSC distributions from validation and testing on the Ma et al. and An et al. datasets. The figure is divided into three parts. A: Boxplot showing the results of the 5-fold cross-validation on the Ma et al. dataset. B:

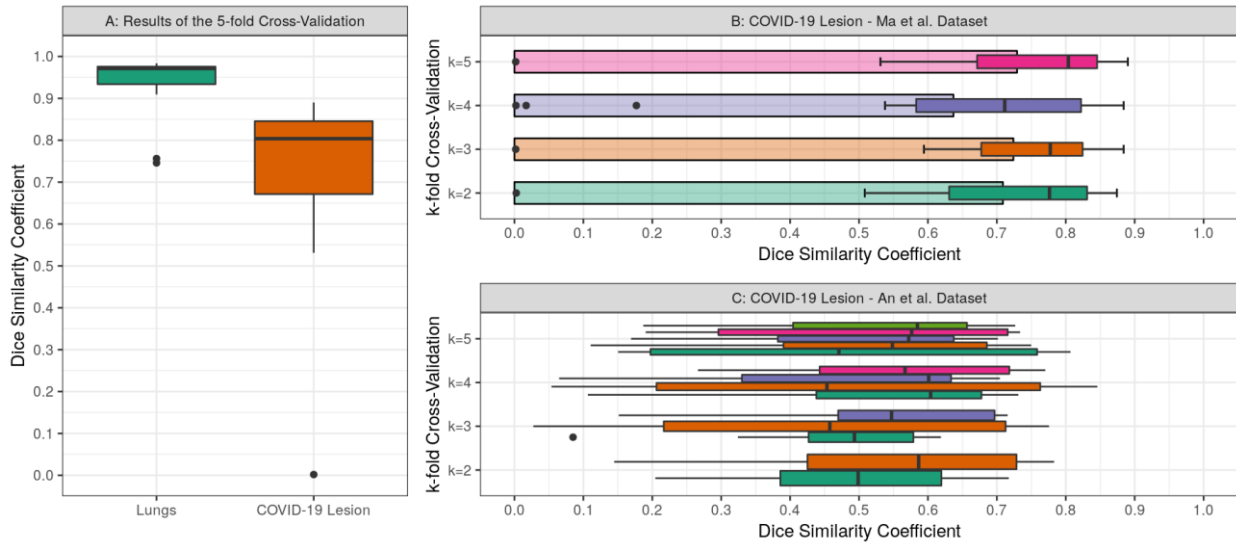


Figure 4.29: Boxplots showing validation and testing performance for COVID-19 segmentation.

Boxplots and bar plots showing the average DSC for each k-fold cross-validation run on the Ma et al. dataset. C: Boxplots for each model of the k-fold cross-validation on the An et al. testing dataset.

Table 4.8: Performance impact of data augmentation and preprocessing on COVID-19 segmentation.

Fold	DataAug: Excluded		DataAug: Included		DataAug: Excluded		DataAug: Included	
	Lungs	COVID-19	Lungs	COVID-19	Lungs	COVID-19	Lungs	COVID-19
1	0.711	0.031	0.397	0.166	0.867	0.530	0.907	0.556
2	0.046	0.186	0.275	0.050	0.979	0.819	0.977	0.801
3	0.190	0.241	0.168	0.057	0.951	0.814	0.952	0.829
4	0.080	0.005	0.175	0.114	0.979	0.819	0.979	0.853
5	0.520	0.194	0.360	0.201	0.964	0.798	0.967	0.765
AVG	0.309	0.131	0.275	0.118	0.948	0.756	0.956	0.761

For the sensitivity analysis, average evaluation metrics were calculated for each k-fold cross-validation (Table 4.7) as well as for each data augmentation and preprocessing configuration (Table 4.8). The 5-fold cross-validation revealed the best performance on all evaluation metrics on the validation set, whereas the 4-fold cross-validation was superior on the testing set. The DSC difference between the best k-fold cross-validation and the worst is 0.093 on validation and 0.106 on testing for COVID-19 lesion segmentation. The inclusion of data augmentation and preprocessing increased the pipeline performance on average by 0.647 for lung and by 0.630 for COVID-19 lesion segmentation based on the DSC, which is summarized in Table 4.8. Table 4.8 also shows the achieved results with in-/excluded preprocessing (PreProc) and data augmentation (DataAug) to evaluate their performance influence on the model. The table presents the median DSC on Lung and COVID-19 infection segmentation for each CV fold of the 5-fold cross-validation and the global average (AVG) based on the Ma et al. dataset.

Through validation monitoring, no overfitting was observed. The training and validation loss function revealed no significant distinction from each other, which can be seen in Figure 4.30. The figure shows the loss course during the training process for training (red) and validation (cyan) data for the 5-fold cross-validation from four pipeline runs including ('on') or excluding ('off') data augmentation (DataAug) and preprocessing (PreProc) techniques. The lines were computed via Gaussian Process Regression and represent the average loss across all folds for each 5-fold cross-validation pipeline run. The final pipeline fitting curve is illustrated in the bottom-right corner (D). During the fitting, the performance settled down at a loss of around 0.383 for the 5-fold cross-validation (Figure 4.30-D) which is a generalized DSC (average of all class-wise DSCs) of around 0.919. Because of this robust training process without any signs of overfitting, it was concluded that fitting on randomly generated patches via extensive data augmentation and random cropping from a variant database, is highly efficient for limited imaging data.

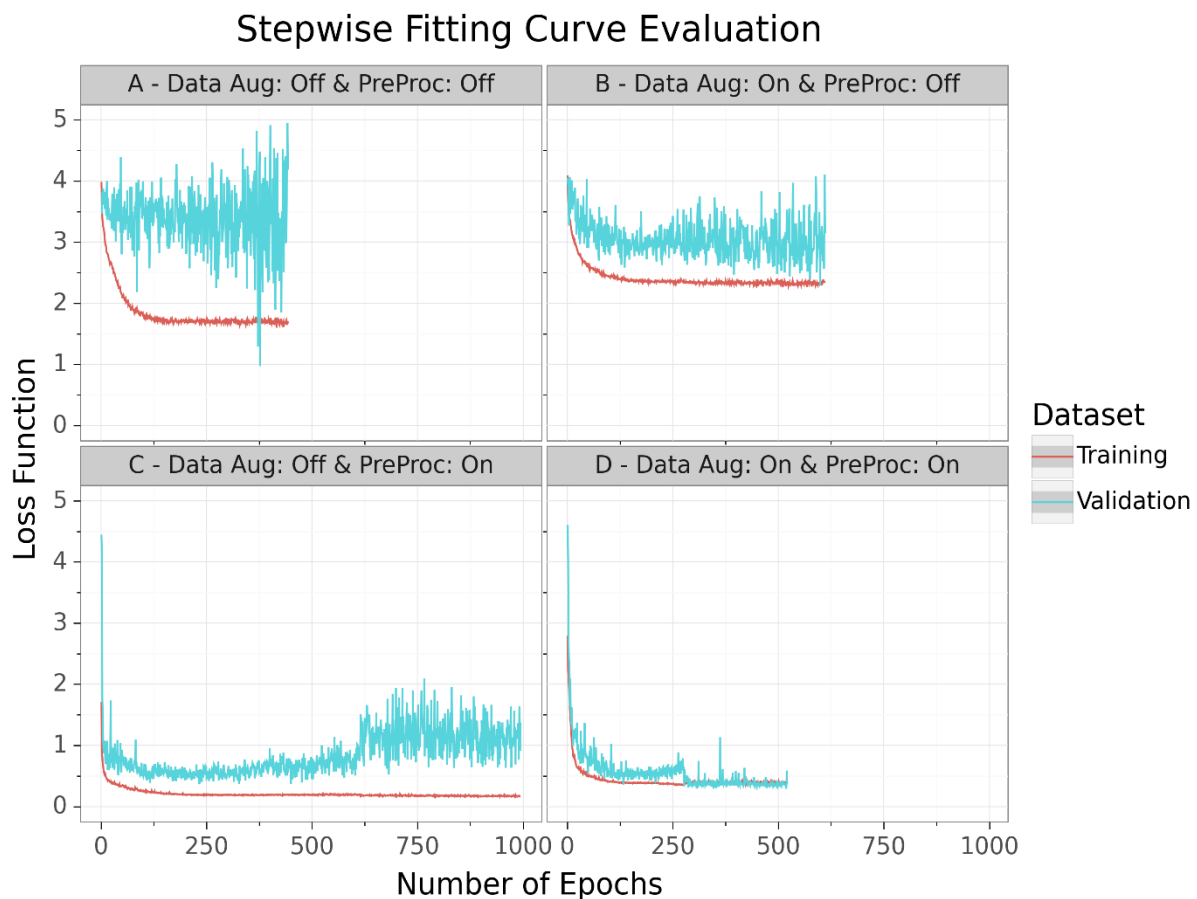


Figure 4.30: Fitting curves of the COVID-19 segmentation pipeline.

Exemplary for the model performance of the 5-fold cross-validation, four samples with annotated ground truth and predicted segmentation are visualized in Figure 4.31.

Discussion

From a medical perspective, detection of COVID-19 infection is a challenging task and one of the reasons for the weaker segmentation accuracy in contrast to the lung segmentation. The reason for this is the variety of GGO and pulmonary consolidation morphology. In contrast to the Specificity, the DSC as well as the Sensitivity are showing a lower but more reliable performance evaluation comparable with the visualized segmentation correctness. The reason for this is that false negative predictions have a strong impact on these two metrics. Especially, in medical image segmentation, in which ROIs are quite small compared to the remaining image, a few incorrect predicted pixels have a large impact on the resulting score. Such strict metrics are required in order to compensate the class imbalance between mostly background and small ROIs in medical imaging. Nevertheless, the proposed MIS pipeline allowed the fitting of a model which is able to segment COVID-19 infection with state-of-the-art accuracy that is comparable to models trained on large datasets.

In order to provide further insights on the influence of the implemented methodology on the achieved performance, the proposed pipeline was analyzed through a sensitivity analysis based on cross-validation and variable data augmentation as well as applied preprocessing configuration. All other configurations as well as the neural network architecture remained the same as described in the methods section. Thus, this experiment resulted in 30 models (14

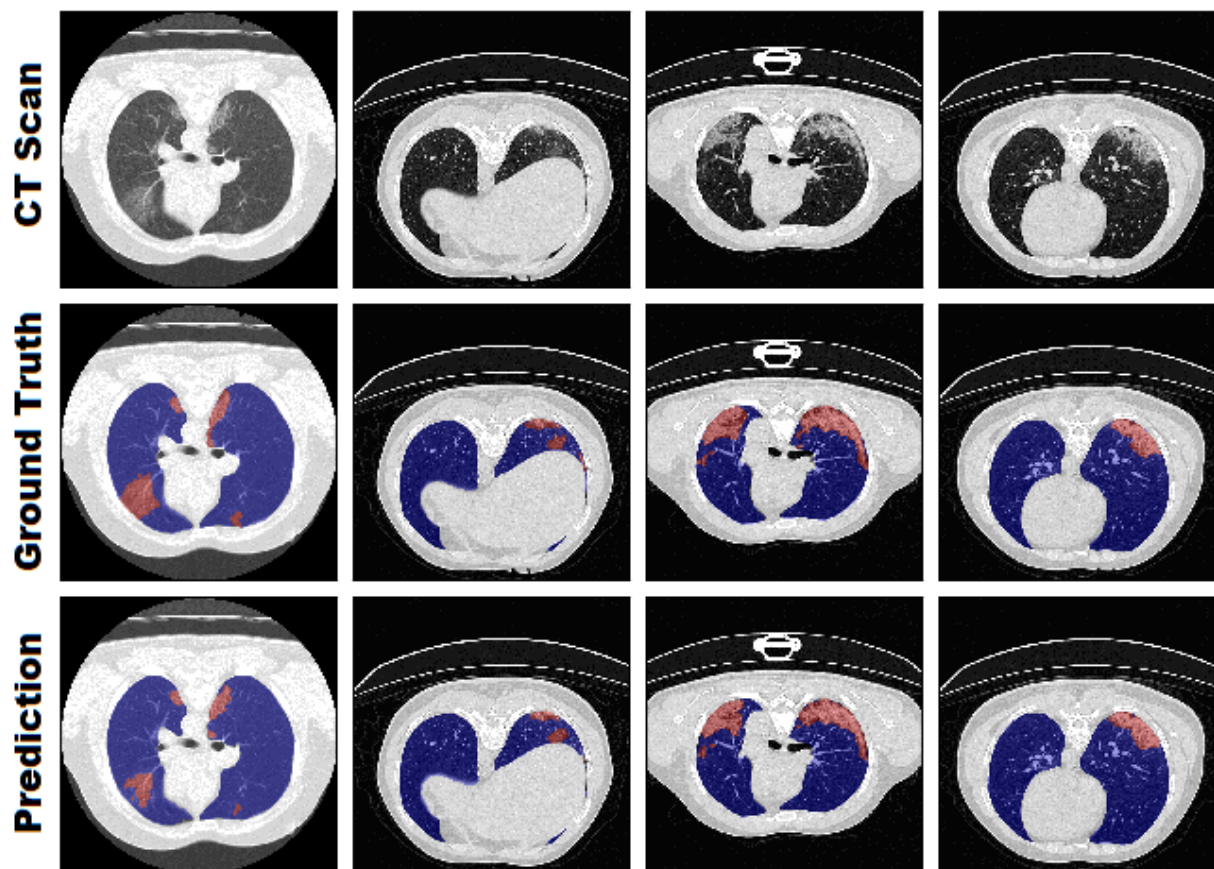


Figure 4.31: Visual comparison between radiologist annotations and the proposed segmentation pipeline.

models from cross-validation ranging from k-fold 2 up to 5 and 15 models from three 5-fold cross-validation runs with variable data augmentation as well as preprocessing configuration).

The fitting process of the different runs revealed that extensive data augmentation plays an important role in avoiding overfitting and improving model robustness, as can be seen in the fitting curves of Figure 4.30. Therefrom, the model overfitted on the training data. The on-the-fly data augmentation helped the model to learn a more generalized pattern for recognizing the lungs and infected regions instead of just memorizing the training data. In contrast, the preprocessing methods increased the overall performance of the model by simplifying the computer vision task. The applied methods like resampling or clipping led to a search space reduction which increased the chances of the model identifying patterns in the imaging data. This advantage was also shown in the resulting performances, which can be seen in Table 4.8. As expected, the pipeline run with no data augmentation as well as no preprocessing appeared to be the worst model. In contrast, the preprocessing techniques demonstrated the highest performance increase on the testing data of the 5-fold cross-validation. Therefore, the final pipeline build combined data augmentation, for improving robustness, and preprocessing techniques, for increasing performance, in order for optimizing inference quality.

The performance evaluation of the sensitivity analysis revealed that there is only a marginal but notable difference between the k-fold cross-validations. As example, the 3-fold cross-validation with a training dataset size of only 13 samples achieved accurate segmentation results on the validation as well as testing set. Interestingly, the 4-fold cross-validation (15 training samples) obtained the best DSC and IoU, and the 3-fold cross-validation had the best Sensitivity on the larger testing set. This demonstrated that generalizability is one of the most important hallmarks of a model, especially if trained on a limited dataset. If all important visual features for the medical condition are present in the training set, a low number of samples can be sufficient by using extensive image augmentation and preprocessing techniques for creating a powerful model. However, if too many samples share similar morphological features without any variation, the risk of overfitting or generating a less generalized model is still present.

Comparison with Prior Work

For further evaluation, the proposed pipeline was compared to other available COVID-19 segmentation approaches based on CT scans. Information and further details of related work were structured and summarized in Table 4.9. The table categories the related work in terms of model architecture, training dataset information for comparability like source, dimension (Dim), sample size as well as the presence of non-COVID-19 slices (Control) and their performance on a validation/testing set.

The authors (Ma et al.), who also provided the dataset used in this experiment, implemented a 3D U-Net approach as a baseline for benchmarking [23]. They were able to achieve a DSC of 0.70355 and 0.6078 for lungs and COVID-19 infection, respectively. The proposed model in this thesis was able to outperform that baseline. It is important to mention that the authors of this baseline trained with a 5-fold cross-validation sampling of 20% training and 80%

validation, whereas the proposed model used the inverted distribution for the k -fold cross-validations ($k-1$ folds for training and the k fold for validation). Based on the Ma et al. dataset, Wang et al. [319] gathered more samples, expanded the dataset, and also applied a 3D U-Net which resulted in a DSC of 0.704. Another approach from Yan et al. [321] developed a novel neural network architecture (COVID-SegNet) specifically designed for COVID-19 infection segmentation with limited data. The authors tested their architecture on a limited dataset consisting of ten COVID-19 cases from Brainlab Co. Ltd (Germany) and were able to achieve a DSC of 0.987 and 0.726 for lungs and infection, respectively. Hence, COVID-SegNet as well as the proposed approach in this thesis achieved similar results. This raises the question if it is possible to further increase the performance of the proposed model by switching from the standard U-Net to an architecture specifically designed for COVID-19 infection segmentation like COVID-SegNet. Further approaches, with the aim to utilize specifically designed architectures, were Inf-Net (Fan et al.) [316] and MiniSeg (Qiu et al.) [318]. Both were trained on 2D CT scans and achieved for COVID-19 infection segmentation DSCs of 0.764 and 0.773, respectively. Although diverse datasets were used for training, which leads to incomparability of the results, it is highly impressive that they achieved similar performance as approaches based on 3D imaging data. The 3D transformation of these architectures and the integration into the proposed pipeline would be an interesting experiment to evaluate improvement possibilities. Other high-performance 2D approaches like Saood et al. [314] and Pei et al. [188] were difficult to compare due to these models being purely trained and evaluated on 2D slices with COVID-19 presence [322].

Table 4.9: Related work overview and comparison for COVID-19 segmentation.

Related Work		Training Dataset				Validation/Testing Performance	
Author	Model Architecture	Source	Dim	Sample Size	Control	COVID-19 (DSC)	Sample Size
Amyar et al. [302]	U-Net (Standard)	Amyar et al. [302]	2D	1,219	Yes	0.78	150
Fan et al. [316]	Inf-Net (Attention U-Net)	Fan et al. [316]	2D	1,650	Yes	0.764	50
Qiu et al. [318]	MiniSeg (Attention U-Net)	Qiu et al. [318]	2D	3,558	Yes	0.773	3,558
Saood et al. [314]	U-Net (Standard)	SIRM [322]	2D	80	No	0.733	20
Saood et al. [314]	SegNet	SIRM [322]	2D	80	No	0.749	20
Pei et al. [188]	MPS-Net (Supervision U-Net)	SIRM [322]	2D	300	No	0.833	68
Zheng et al. [315]	MSD-Net	Zheng et al. [315]	2D	3,824	Yes	0.785	956
Wang et al. [190]	COPLE-Net (enhanced U-Net)	Wang et al. [190]	2D	59,045	Yes	0.803	17,205
Ma et al. [102]	U-Net (Standard)	Ma et al. [102]	3D	20	Yes	0.608	20
Ma et al. [23, 102]	nnU-Net	Ma et al. [102]	3D	20	Yes	0.673	20
Wang et al. [319]	U-Net (Standard)	Wang et al. [319]	3D	211	Yes	0.704	211
Yan et al. [321]	COVID-SegNet	Yan et al. [321]	3D	731	Yes	0.726	130
He et al. [317]	M ² UNet (Segmentation only)	He et al. [317]	3D	666	Yes	0.759	666
Proposed Pipeline	U-Net (Standard)	Ma et al. [102]	3D	20	Yes	0.804 / 0.661	20 / 100

Limitations

However, it is important to note that the majority of current segmentation approaches in research are not suited for clinical usage. The bias of current models is that the majority are only trained with COVID-19 related images. Therefore, it is not certain how well the models can differentiate between COVID-19 lesions and other pneumonia, or entirely unrelated

medical conditions like cancer. Furthermore, identical to COVID-19 classification, the models reveal huge differences depending on which dataset they were trained on. Segmentation models purely based on COVID-19 scans are often not able to segment accurately in the presence of other medical conditions [23]. Additionally, there is a high potential for false positive segmentation of pneumonia lesions that are not caused by COVID-19. This demonstrates that these models could be biased and are not suitable for COVID-19 screening. Nevertheless, current infection segmentation models are already highly accurate for confirmed COVID-19 imaging. This offers the opportunity for quantitative assessment and disease monitoring as applications in clinical studies.

Despite that the proposed model and those of others, which are based on limited data, are capable of accurate segmentation, it is essential to discuss their robustness. Currently, there are only a handful of annotated imaging datasets publicly available for COVID-19 segmentation. More imaging data with especially more variance (different COVID-19 states, other pneumonia, healthy control samples, etc.) need to be collected, annotated, and published for researchers. Similar to Ma et al. [23, 102], community-accepted benchmark datasets have to be established in order to fully ensure robustness as well as comparability of models.

4.5.5 Conclusions

Even though neural networks are capable of accurate decision support, their robustness is highly dependent on dataset size for training. Various medical conditions like rare or novel diseases lack available data for model training which decreases generalizability and increases the risk of overfitting. In this study, an approach for automated as well as robust segmentation of COVID-19 infected regions in CT volumes based on a limited dataset was developed and evaluated. The proposed method focuses on on-the-fly generation of unique and random image patches for training by performing several preprocessing methods and exploiting extensive data augmentation. Thus, it is possible to handle limited dataset sizes which act as a variant database. Instead of novel and complex neural network architectures, the standard 3D U-Net was utilized. This study proved that the proposed MIS pipeline is able to successfully train accurate and robust models without overfitting on limited data. Furthermore, the pipeline was able to outperform current state-of-the-art semantic segmentation approaches for COVID-19 infected regions. This work has great potential to be applied as a clinical decision support system for COVID-19 quantitative assessment and disease monitoring in a clinical environment. As further research, the Author is planning to integrate ensemble learning techniques in the proposed pipeline to combine the predictive strengths of the k-fold cross-validation models. Additionally, clinical studies are needed for robust validation of clinical performance and generalizability of models based on limited data. Also, it is planned to expand the testing data and evaluation by adding cases with non-COVID-19 conditions like bacterial pneumonia or lung cancer.

5

Medical Image Classification

Image classification has an essential role in medical image processing. The aim of medical image classification (MIC) is the automated labeling of a complete image to predefined classes, e.g. to a diagnosis or a condition, by focusing on the interpretation of an image. Thus, an MIC pipeline is able to classify medical images into a pre-defined set of classes or terminology that can correspond to different tissues, organs, pathologies or abnormalities, and other biologically relevant classifications [10, 13]. The resulting classifications can be utilized by clinicians, documentation workflows, or additional AI approaches [10]. The automatic classification of images is able to aid clinicians in tasks like diagnosis, treatment, and time-consuming processes by providing further insights to increase decision reliability as well as through structured reporting [10, 11, 13]. Popular applications today are differentiation between benign and malignant tumors, disease classification, or automated categorization of medical structures like tissues or organs [10]. Nevertheless, clinical decision support systems for disease detection as well as structured reporting are currently highly popular research topics for clinical trials and get slowly integrated into the clinical workflow of modern hospitals [11, 62].

In this chapter, the Author proposes the framework AUCMEDI which is a software package for the standardized setup of state-of-the-art neural network models for MIC combining the concept of an extensive but simple toolkit for deep learning experts and straightforward reusability for interested clinicians. Furthermore, four studies by the Author utilizing the AUCMEDI framework are presented. The studies not only demonstrate the wide applicability and competitive performance capabilities of AUCMEDI but also contribute with their outcomes to the field of MIC, rare disease detection in ophthalmology, as well as COVID-19 research.

5.1 History and Current State

Since about 1993 starting with the work by Busch and Groß [323], approaches for automatic image classification to detect tumors and lesions are available [10]. Throughout this time, a wide range of algorithms was proposed for classification in medical imaging. In comparison to image segmentation methods, medical image classification focuses on interpretation and diagnosis support. Consequently, due to the medical complexity, the traditional workflow of MIC pipelines is based on prior analysis results from MIS as well as quantitative image analysis for feature extraction and the application of a supervised learning algorithm. In contrast to modern MIC pipelines, nowadays, which directly utilize imaging information, traditional MIC was the last processing step in an MIA pipeline [10]. This subchapter provides a short overview of methods for MIC throughout the last decades and current challenges in the field.

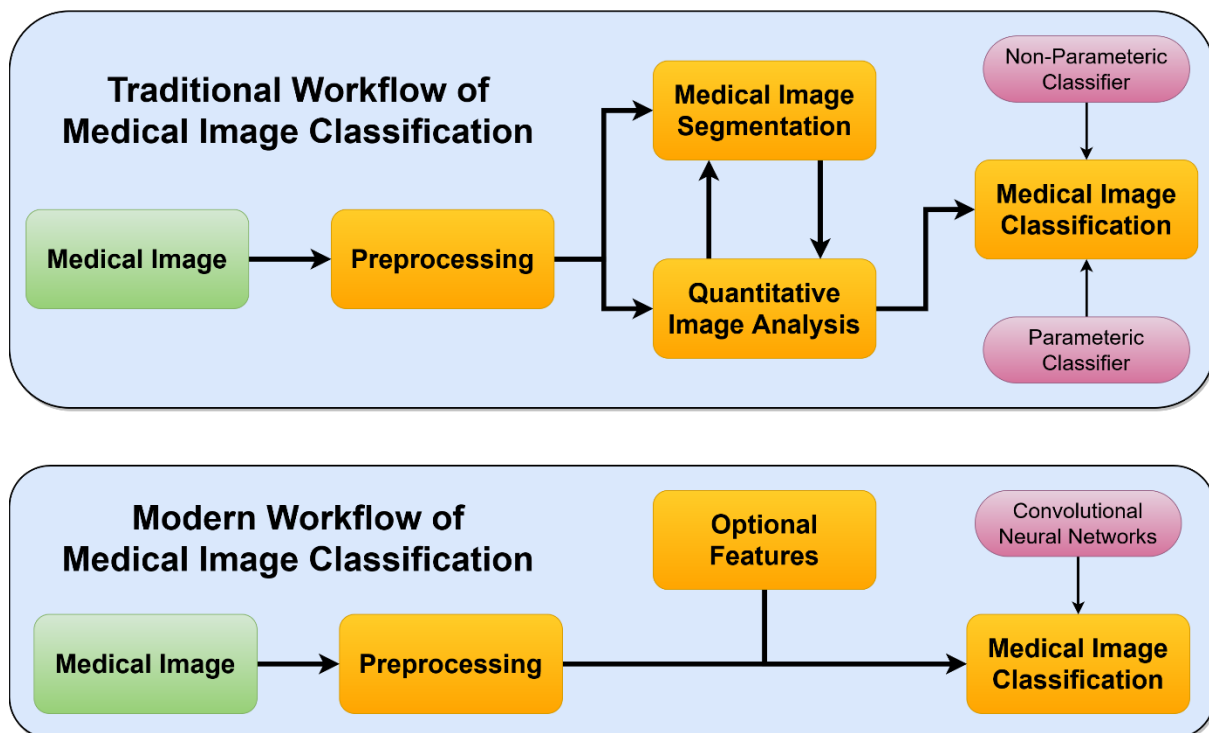


Figure 5.1: Workflow overview in comparison of traditional and modern MIC.

5.1.1 Methods

Feature Extraction

In the early phase of MIC research, the number of methods, which were capable of directly analyzing medical images, was limited [10]. However, there were already studies available on successful segmentation and quantification of medical features [10, 124, 324]. This allowed utilizing the algorithms in a pipeline in which the detected features are used as input for the subsequent classifier. With such a workflow, it was possible to support diagnoses and treatment decisions by automatic image analysis.

The extraction of features can be grouped into three methodology categories. In the first category, the features for the classifier originate purely from automatic quantitative image analysis. Thus, the obtained features can range from general image information like pixel intensity distributions to texture and fractal analyses [10, 325]. In the second category, the feature extraction process utilizes MIS methods prior to the quantitative analysis. The procedure is still automatic without manual interference needed. Further information based on the segmentation of ROIs allows the integration of more detailed quantitative analysis methods like the assessment of elementar (size, distance, or volume) and morphological features (contour or elongation) [10, 325]. The last category of features originated from semi-automatic processes of segmentation and quantitative analysis. Hereby, a clinician manually annotates ROIs which allows a more robust feature extraction with higher reliability on precision and correctness compared to automatic methods. Consequently, the classifier is able to achieve better results due to the noise reduction in its input features [325].

More information about the fields of quantitative image analysis and MIS can be found in Chapter 2.4.1 and further details about MIS methods in Chapter 4.1.1.

Parametric Classifier

For classification of the extracted features, the model algorithms can be distinguished between parametric and non-parametric classifiers. The parametric classifiers, also called statistical classifiers, utilize a finite set of parameters for representing the learned information during the training process [10]. Russell and Norvig described parametric classifiers as follows [326]:

“A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a parametric model. No matter how much data you throw at a parametric model, it won't change its mind about how many parameters it needs.”

Typical algorithms categorized as parametric classifiers are Logistic Regression models and (Naïve) Bayes classifiers [10, 16, 326]. The advantages of parametric classifiers are that the representation of the hypothesis, which is used for the classification process, is often defined through simple statistical and probability-based models. Such representation as well as the fixed number of parameters in such models directly correlates with rapid training speed and fewer required training samples [10, 326]. However, parametric classifiers are improper models for more complex tasks or training on biased features [10]. For more details about parametric classifiers, the Author refers to the work by Russell and Norvig [326] or Mohri et al. [16].

Non-Parametric Classifier

In comparison with parametric classifiers, non-parametric classifiers utilize an infinite set of parameters for representing the learning information during the training process [10]. Depending on the data and feature distribution processed during the training, the necessary number of parameters are estimated for reliable pattern representation. Russell and Norvig described non-parametric classifiers as follows [326]:

“Nonparametric methods are good when you have a lot of data and no prior knowledge, and when you don’t want to worry too much about choosing just the right features.”

Typical algorithms categorized as non-parametric classifiers are k-nearest neighbors and decision trees [10, 16, 326]. The advantages of a non-parametric classifier are its capabilities for handling complex tasks and finding patterns in biased features. Nevertheless, non-parametric classifiers in general require more training data, have a higher risk of overfitting, and need more time for the fitting process [10, 16, 326]. For more details about non-parametric classifier, the Author refers to the works of Russell and Norvig [326] or Mohri et al. [16].

Neural Networks and the State-of-the-Art

Earlier methods for MIC heavily depended on prior quantitative image analysis methods for feature extraction. With the introduction of reliable neural network models using convolutional approaches, it was possible to directly analyze the image without prior analyzing algorithms. More information about the theory of neural networks can be found in Chapter 2.2 and further details about computer vision based image classification in Chapter 2.3.1. This technological progress allowed not only the reduction of noise generated by intermediate processing steps like MIS, but also allowed utilizing the complete information provided by the image without manually selecting specific features for representing the image information [10]. The advantage as well as the disadvantage of this advancement is that the model itself finds the optimal features for classification. Such dynamic feature selection enables the detection of unknown features but also the possible inclusion of biased features like written annotations in X-ray slides resulting in a model detecting the notes on X-ray edges instead of analyzing medically relevant features [327, 328].

The progress of the computer vision community in the field of general neural network based image classification lead to a significant increase in the prediction capabilities of MIC pipelines. In recent years, studies using deep learning based MIC pipelines proved remarkable performance and adaptability in all subfields of medical imaging [11, 13, 34–36, 101]. Due to the increasing popularity in the computer vision field, a large number of architectures for image classification was developed [74]. These deep learning architectures are heavily used in modern MIC pipelines. In terms of architectures, MIC pipelines are highly varying depending on the imaging modality and medical condition to classify [13, 207, 329]. Thus, there is no gold-standard architecture for deep learning based MIC compared to the architecture standardization in MIS by the U-Net. Even so, the majority of applications in MIC are still focused on clinical research, there are an increasing number of efforts to integrate MIC pipelines in clinical workflows for providing automatically filled structured reporting templates in radiology [330–332].

Deep learning based MIC is the core method in all discussed studies of this thesis. The terms ‘convolutional neural networks’, ‘deep learning based MIC’ as well as ‘neural network models’ will be used as equivalent terms and be referred to as the main method for MIC in this work.

5.1.2 Challenges

The automatic analysis of medical images reveals a high complexity due to ambiguous image quality, inconsistent pixel intensities, technical noise as well as artifacts, and high variability [124, 129, 182]. Furthermore, the difficulty in distinguishing between medical abnormalities for diagnosis and ensuring reliable prediction performance are challenging tasks. These challenges in analyzing medical imaging are shared with the field of MIS and are discussed in detail in Chapter 4.1.2.

In comparison to the field of MIS in which each pixel and therefore the ROI itself is marked for the model, an MIC model need to identify possible features for distinguishing by itself. The advantage of this characteristic of MIC is the reduced annotation effort for clinicians [23, 333]. However, the considerable challenge of this characteristic is an increased task complexity which can result in higher training times and, in the worst case, weaker predictors. Furthermore, due to the missing link between relevant features and outcome as well as to ensure statistical robustness in performance computations, more image samples are needed for training as well as for evaluation [264, 334]. Another challenge for MIC can be the annotation quality. As already discussed in Chapter 3.1.3, automatic annotation processes to obtain labels and unspecific as well as inaccurate classes can have a drastically negative impact on the prediction capabilities of MIC models [156]. Modern MIC pipelines have to be aware of these pitfalls and need to integrate methods to ensure reliability and robust predictions by reducing noise and bias influences.

Similar to MIS, generalizability, reproducibility, and reusability are significant problems in the field of MIC. Statements from medical staff in news articles like “*Hundreds of AI tools have been built to catch covid. None of them helped.*” [50] as well as application studies [32, 38, 39] revealed that the integration of pipelines for image classification into a hospital environment presents significant difficulties. Already implemented solutions from the literature are usually independent software, so-called ‘island solutions’, which were developed for a specific disease and optimized for an individual dataset [35, 39, 49]. Due to the lack of generalizability, clinicians are faced with the challenge of lacking reusability on their own datasets and no possibilities for practical usage in clinical research. Further research is needed to find solutions for the challenge of non-reproducible and non-standardized MIC pipelines in order to enable practical usability in clinical research.

5.2 AUCMEDI: a Framework for Automated Classification of Medical Images

The field of AI-based medical image classification has experienced rapid growth in recent years [9]. The possible integration of such deep neuronal networks into the clinical routine is therefore currently a highly popular research topic. The aim of clinicians, especially in the imaging disciplines, is therefore to use the models as clinical decision support in order to improve diagnostic certainty or to automate time-consuming processes [9, 35, 39].

However, as mentioned in the previous chapter, already implemented solutions from the literature are usually independent software lacking reproducibility, generalizability, and reusability. Studies identified these ‘island solution’ pipelines as a major contributor to the lack of reproducibility and therefore the lack of practical use for clinical research [32, 35, 38, 39]. This is why standardized pipelines for MIC research are needed. Another issue of other novel standardized toolkits like MONAI [175] is that these often require extensive deep learning knowledge for building state-of-the-art pipelines and still require heavily utilizing third-party packages or implementing pure base frameworks. Notwithstanding, these toolkits are significantly important for research applications but still do not solve the issue of reusability for clinical applications. In general, clinical researchers do not have the required deep learning expertise for utilizing such toolkits which is why frameworks with intuitive high-level functionality and automated machine learning abilities are needed [32]. In order to pave the way for the integration of MIC pipelines into clinical routine, it is needed to solve these issues.



Figure 5.2: Logo of the proposed framework AUCMEDI.

In this chapter, the Author proposes a framework for intuitive and standardized building of state-of-the-art pipelines for medical image classification. The aim of the proposed framework AUCMEDI (pronunciation AUC-MEDI) is to provide **Automated Classification of Medical Images** utilizing modern convolutional neural networks and deep learning models. The framework combines the concept of an extensive but simple toolkit for deep learning experts, straightforward reusability for interested clinicians, and the capability of integration in clinical workflows as well as sensitive IT infrastructures. To make this possible, the framework offers next to a high-level API for the standardized construction of MIC pipelines, reproducible as well as direct application using dockerization and automated machine learning. Pipelines created with the proposed framework are based on the most recent computer vision as well as medical specialized processing methods, and heavily utilize open interfaces to allow high configurability, adaptability to different workflows, as well as integration of novel methods. With AUCMEDI, researchers and clinicians are able to setup a completely modern and easy-to-integrate pipeline for medical image classification with just a few lines of code.

5.2.1 Implementation

The open-source Python library AUCMEDI is a framework for the intuitive and standardized building of state-of-the-art pipelines for medical image classification with deep neural networks. The framework is community-friendly and straightforward to use but still open for customization as needed. Furthermore, it strives to combine usage simplicity, high performance, and reproducibility. AUCMEDI provides several core features, which are also illustrated in Figure 5.3:

- Support for 2D as well as 3D imaging encoded in prevalent medical file formats
- Various preprocessing methods for preparation, task simplification, and bias reduction
- Library of classic and modern neural network architectures for binary, multi-class and multi-label classification with efficient methods against class imbalance
- Simple integration of ensemble learning techniques in a pipeline
- Explaining opaque decision-making processes of AI models with Explainable AI
- Automated machine learning mentality for easy deployment, integration, and maintenance

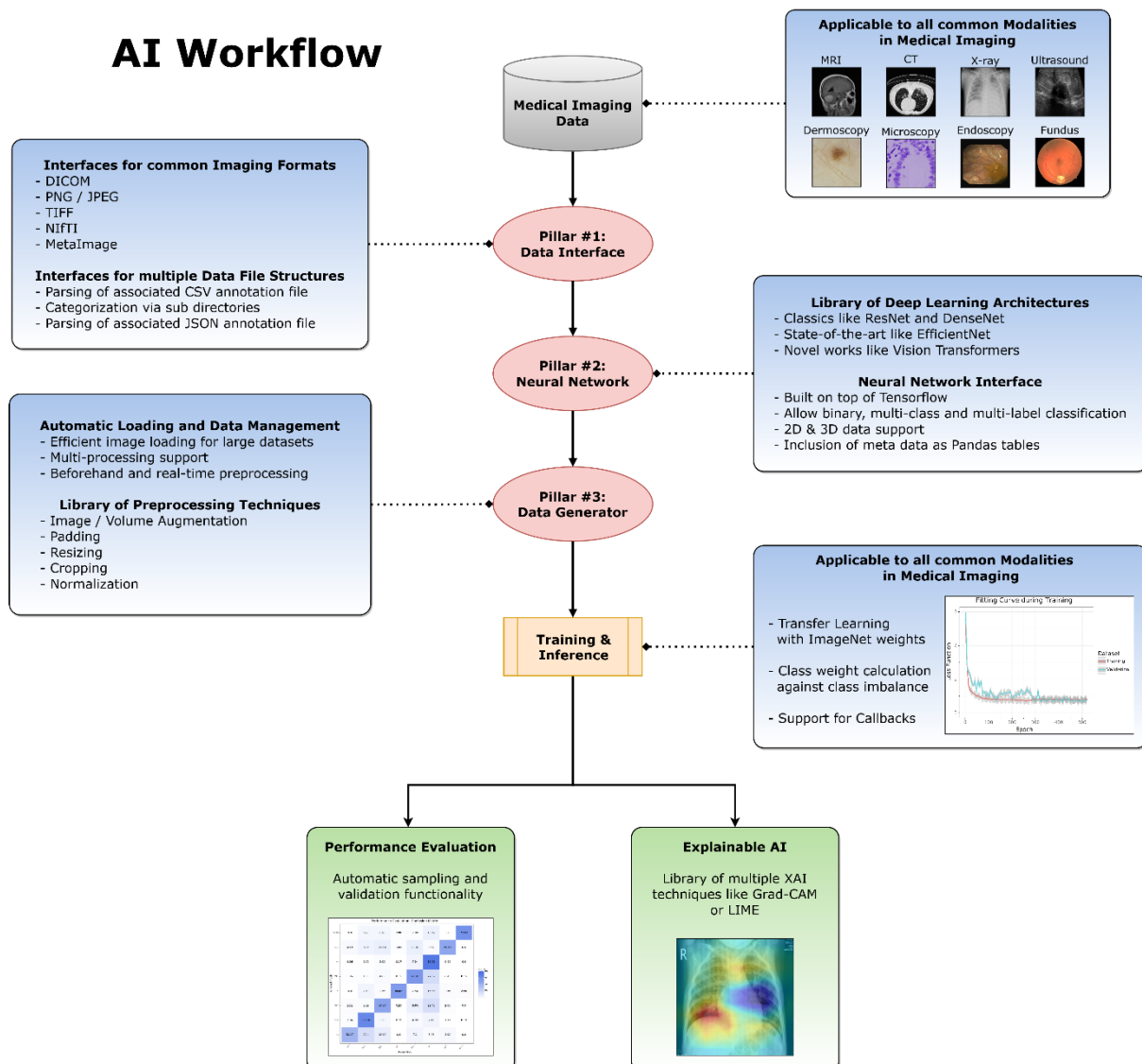


Figure 5.3: Flowchart diagram of the AUCMEDI pipeline.

Data Loading Interface

An MIC pipeline performs several types of data input as well as output operations, which is why efficient data loading is crucial and highly optimized in AUCMEDI to avoid bottlenecks leading to long training times. The large variety of modalities in medical imaging generates different types of representations of an image ranging from different dimensions, pixel intensity scales, metadata, or file structures and formats. The data, in particular the image, loading process of AUCMEDI is highly dynamic in order to be applicable to any common as well as uncommon medical imaging data. In order for the proposed framework to support any common medical imaging modality, the following interfaces for loading images are supported: SimpleITK [134, 135] for loading DICOM, NIfTI, MHA, or other common medical formats, Pillow [266] for loading regular 2D imaging formats like PNG, JPEG, or TIFF, NumPy [138] for loading already preprocessed or uncommon imaging data, and from memory for loading any custom generated data. Next to multi-dimension imaging support, the proposed framework also incorporates imaging-related metadata like voxel spacing (slice thickness) in MRI and CT imaging. For passing annotations into the MIC pipeline, AUCMEDI provides IO interfaces that allow extracting classification labels from datasets in different file structures or formats. More specifically, the labels can be inferred from the directory structure of the dataset, a CSV file, or a JSON file that is provided. Nevertheless, the open structure of AUCMEDI also allows passing annotations directly in memory for optimal integration of any dataset structure.

Image Augmentation

The augmentation of data is an essential and widely used technique in the field of machine learning to increase feature variance as well as dataset size, and decrease overfitting. In particular for MIA, on-the-fly (online) image augmentation is a gold-standard method to create reliable and high-performing classification pipelines [109, 163, 165, 166, 181]. For providing a large selection of augmentation techniques, AUCMEDI integrated three interfaces for popular image (2D) as well as volume (3D) augmentation packages: Albumentations [140], Volumentations [335, 336], and batchgenerators [142]. The Albumentations package is widely popular in the field of computer vision and provides an extensive number of augmentation techniques for regular 2D images. The Volumentations package is a community-driven adaptation of Albumentations for 3D imaging in which the Author actively contributed as well as continued the development of the open-source project. The batchgenerators package was developed by the Division of Medical Image Computing at the German Cancer Research Center (DKFZ) and provides efficiently implemented augmentation techniques particular for MIA. Furthermore, it allows the augmentation of 2D and 3D images. Due to the integration of multiple augmentation packages, AUCMEDI supports the following augmentation techniques: Flipping, rotation, scaling, cropping, grid distortion, compression, downscaling, elastic transformation, introducing gaussian blur as well as noise, and altering brightness, contrast, saturation, hue, as well as gamma.

Preprocessing and Dynamic Subfunctions

The preprocessing of medical images is one of the key steps in any MIC pipeline. The field of preprocessing methods aims to achieve task simplification, bias reduction, lowering hardware requirements, and general image preparation for further workflow processes. Moreover, the preparation of medical images require specially configured techniques to avoid information loss or unintended exclusion of relevant medical context. More information about image preprocessing methods can be found in Chapter 3.3. Summarized, preprocessing is needed to enable compatibility for the required model input, improve performance, and ensure adequate model fitting. AUCMEDI offers an automatic and powerful preprocessing interface which is defined in two components. The first component is the automatic preparation as well as packing of images, optionally provided metadata, and annotations for training in batch units to be passable into a neural network model. The second component of preprocessing in AUCMEDI is that the user can select desired methods that are applied to the images. The proposed frameworks provide an extensive library of the following preprocessing methods which are called Subfunctions in AUCMEDI: Chromer, clipping, color constancy, cropping, padding, resizing, and normalization. With the open interface for preprocessing in AUCMEDI, custom Subfunctions can be easily integrated into the pipeline.

Neural Network Model

The neural network model is the core of an MIC pipeline and represents the solver of the classification task. A model requires prior training based on supervised learning in order to be used for the prediction of unknown data. This concept is directly represented in AUCMEDI in which a model object can start a training or prediction process but requires fitted weights for reliable inference. In AUCMEDI, the neural network of a model is represented by the subpackage Keras [272] from the base framework TensorFlow [72]. This allows building neural network architectures with the widespread base framework but integrating these into the intuitive and robust AUCMEDI workflow. Through the hyperparameter interface in AUCMEDI, architectures are also switchable resulting in a dynamic model representation with flexible customization options. Next to a dynamic architecture, AUCMEDI offers full control of hyperparameter configuration but utilizes appropriate default values in order to be adapted to any imaging modality. These hyperparameters include training-related parameters like the loss function, model input parameters like the input shape, and model output parameters like the last activation function defining the classification task. AUCMEDI supports binary, multi-class, as well as multi-label classification.

For the training process, AUCMEDI utilizes the Focal loss from Lin et al. [213] by default which can be changed to any TensorFlow-compatible loss function. Also, the proposed framework offers functionality computing and integrating class weights. The training in AUCMEDI provides options for modern epoch definitions with a fixed number of iterations and for transfer learning utilization with automated shallow-tuning and deep-tuning phases. Furthermore, advanced training processes with techniques based on validation monitoring like dynamic learning rates can be built straightforwardly by integrating Keras Callbacks [272].

Library of Deep Learning Architectures

The field of modern computer vision consists of a large number of different image classification architectures. These architectures have different strengths and weaknesses depending on the application [74, 75, 329]. Differences between architectures are acceptable image dimension, the number of parameters, general complexity, hardware requirements, default input shape (like 224x224 pixels), available ImageNet weights for transfer learning, and expected pixel intensity normalization technique. Even so, there are architectures that are more often used in studies than others, the optimal architecture selection is highly task and data specific [13, 67, 74, 101]. This is why AUCMEDI provides an extensive library of various deep learning architectures which is summarized in Table 5.1.

Table 5.1: Overview of implemented deep learning architectures in AUCMEDI.

Architectures for 2D Imaging			
Architecture	ImageNet Weights	Architecture	ImageNet Weights
Vanilla (custom architecture)	No	EfficientNetB0 [79]	Yes
ResNet50 [80]	Yes	EfficientNetB1 [79]	Yes
ResNet101 [80]	Yes	EfficientNetB2 [79]	Yes
ResNet152 [80]	Yes	EfficientNetB3 [79]	Yes
ResNet50V2	Yes	EfficientNetB4 [79]	Yes
ResNet101V2 [337]	Yes	EfficientNetB5 [79]	Yes
ResNet152V2 [337]	Yes	EfficientNetB6 [79]	Yes
ResNeXt50 [88]	Yes	EfficientNetB7 [79]	Yes
ResNeXt101 [88]	Yes	InceptionResNetV2 [89]	Yes
DenseNet121 [90]	Yes	InceptionV3 [87]	Yes
DenseNet169 [90]	Yes	MobileNet [91]	Yes
DenseNet201 [90]	Yes	MobileNetV2 [92]	Yes
NASNetMobile [96]	Yes	Vision Transformer (ViT) B16 [94]	Yes
NASNetLarge [96]	Yes	Vision Transformer (ViT) B32 [94]	Yes
VGG16 [84]	Yes	Vision Transformer (ViT) L16 [94]	Yes
VGG19 [84]	Yes	Vision Transformer (ViT) L32 [94]	Yes
Xception [97]	Yes		
Architectures for 3D Imaging			
Architecture	ImageNet Weights	Architecture	ImageNet Weights
Vanilla (custom architecture)	No	ResNet18 [80]	Yes
DenseNet121 [90]	Yes	ResNet34 [80]	Yes
DenseNet169 [90]	Yes	ResNet50 [80]	Yes
DenseNet201 [90]	Yes	ResNet101 [80]	Yes
MobileNet [91]	Yes	ResNet152 [80]	Yes
MobileNetV2 [92]	Yes	ResNeXt50 [88]	Yes
VGG16 [84]	Yes	ResNeXt101 [88]	Yes
VGG19 [84]	Yes		

Furthermore to already implemented architectures, AUCMEDI allows intuitive integration of custom implemented architectures in order to encourage the development, testing, and comparison of novel architectures.

Sampling, Validation, and Evaluation

Next to the core functions of data loading, preprocessing, as well as the neural network model, an MIC pipeline also needs and is determined by sampling, validation, and evaluation.

In order to obtain validation and testing subsets, AUCMEDI offers two sampling functions. The percentage split and the k-fold cross-validation. Both functions support stratified and iterative sampling to group images, multi-class as well as multi-label annotations, and metadata.

Performance and fitting evaluation are essential to assess model quality and reliability. This is why, AUCMEDI offers automatic evaluation functionality for data exploration, fitting curve analysis, performance assessment, and performance comparison of multiple models. The idea of the evaluation function library is to provide quick and simple functionality for inspecting and analyzing data as well as results generated by AUCMEDI. The generated results can be either returned as tables or already visualized figures like roc curves. For performance assessment, the following metrics can be selected: F1-score, Accuracy, Sensitivity, Specificity, AUC, Precision, FPR, false negative rate, false discovery rate, and the number of TP, TN, FP, as well as FN.

Ensemble Learning

The technique of ensemble learning describes the process of combining multiple predictions originating from different models in order to improve performance and robustness [338–341]. The methods for combining the predictions, which are called pooling functions, can be grouped into aggregate functions (equal model weighting) and metalearner (unequal model weighting). The concept of ensemble learning is introduced in detail in Chapter 6. To benefit from these state-of-the-art strategies, AUCMEDI implemented the following ensemble learning techniques: Augmenting for test-time augmentation of unknown images to infer, Bagging for cross-validation based equal model training with different sampling, Stacking for ensembling unequal models trained on the same data with a fitted metalearner stacked on top of it, as well as a combination of Stacking and Bagging through cross-validation with a fitted metalearner stacked on top of it. The aggregate functions were custom implemented, whereas more complex algorithms like metalearner were integrated from scikit-learn [342]. Overall, AUCMEDI offers 5 aggregate functions including averaging as well as majority vote, and 10 metalearners including logistic regression and random forest.

Explainable Artificial Intelligence

Due to neural networks being capable of processing a large amount of information, the predictions from these models are often not comprehensible to humans [126, 343]. The utilization of neural network models for MIC in the context of clinical decision support implies that the provided decisions may not be understandable for clinicians. This is why, explainable artificial intelligence methods (XAI), which aim to explain opaque decision-making processes of AI models, are integrated into AUCMEDI. The framework provides the ‘decoder’ interface for explaining MIC models. This interface generates visualizations of which regions were crucial for the neural network model to predict a classification on a provided unknown image. These regions are visualized as a heatmap overlay on the input image and can be automatically created for each prediction in AUCMEDI. The proposed framework integrated the following XAI methods to compute region importance: Grad-CAM [344], Grad-CAM++ [345], Guided

Grad-CAM [344, 346], Saliency Maps [347], Guided Backpropagation [346], Integrated Gradients [348], Occlusion Sensitivity [343], and LIME [349].

Automated Machine Learning

Especially for clinical scientists without a computer science background, research or application of state-of-the-art MIC approaches is a complex task [26]. In addition, the majority of developed MIC pipelines are specialized for a single dataset without the possibility of fast reproducibility or translational application on other datasets [35, 39, 49]. Clinical scientists are currently confronted with the challenge that high-performing models are existing but can not be applied without various pipeline modifications and extensive knowledge in the field of deep learning [32, 39]. Thus, one of the key features of AUCMEDI is automated machine learning for the building and application of state-of-the-art MIC pipelines. The concept of AutoML is to automate the process of building, deployment, and usage of artificial intelligence [7, 33]. By simplifying this process, the advantages of AI models can be easier utilized outside research environments. In the medical field, AutoML enables easy application, integration into clinical workflows, and maintenance of complex models for MIC [33].

The AutoML module of AUCMEDI offers three workflows: Training, prediction, and evaluation. The three workflows allow building one or multiple models stored as files which can be used for the prediction workflow to classify unknown images. These workflows can be

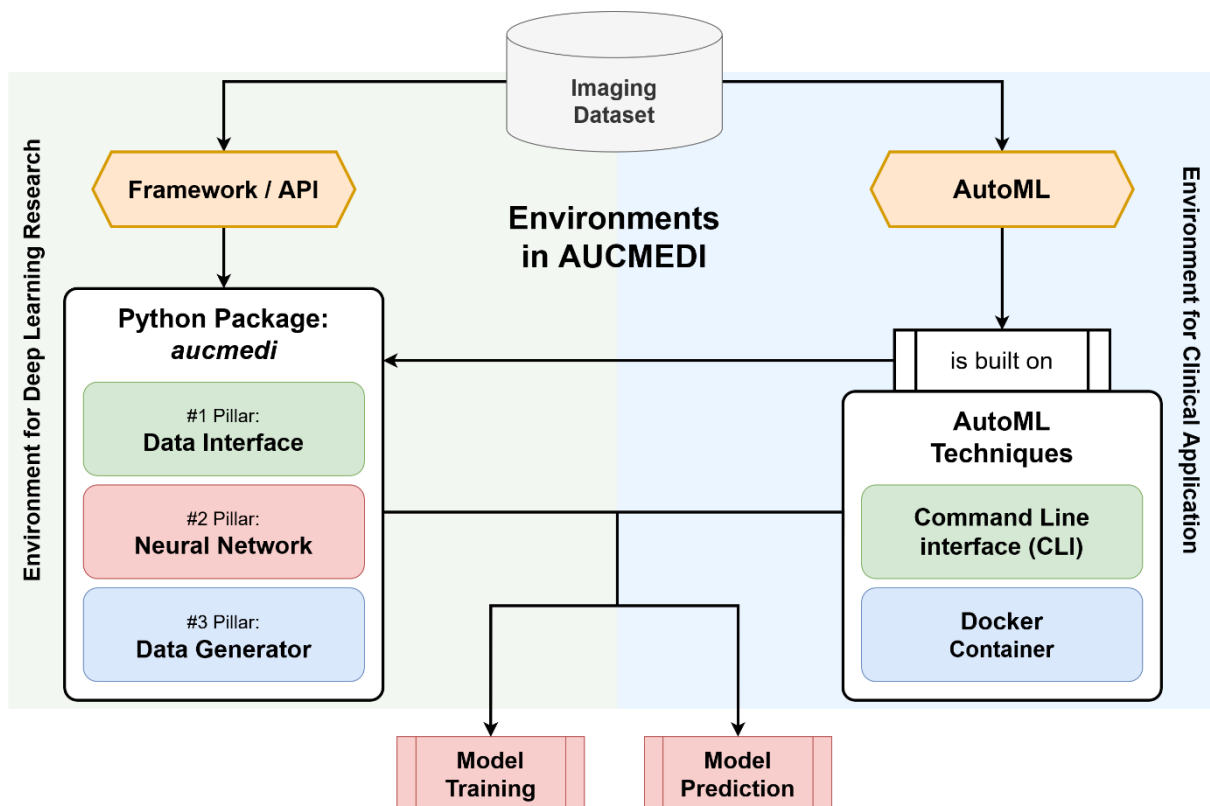


Figure 5.4: Overview of application environments in AUCMEDI.

configured as needed but provide appropriate default configurations as well as automated inferred options which allows straightforward application instead of manually implementing a complete MIC pipeline. With the evaluation workflow, AUCMEDI also provides automatic but extensive performance assessment to quantify the reliability of a model. The AutoML module can be accessed by a command line interface (CLI) as well as by Docker containers [350]. Multiple meta-analyses and guidelines recommended the virtualization software Docker to improve reproducible research in machine learning [42, 351–353]. Through the incorporation of AUCMEDI into the Docker environment, it is possible to ensure functionality in external IT environments. Whereas the CLI is intended for direct as well as uncomplicated usage of AUCMEDI in a local environment and is recommended for clinical research purposes, the Docker containers allow standardized usage in secure environments which is ideal for integration in workflows or usage in sensitive IT infrastructures.

Application Environments

The idea of AUCMEDI is to combine an extensive but intuitive toolkit for deep learning researchers with the capability of straightforward reusability for interested clinicians and secure integration in clinical workflows. Therefore, the proposed software package is designed as a dual-environment framework. AUCMEDI offers a library as a high-level API for the standardized construction of modern MIC pipelines and an AutoML module for reproducible installation as well as direct application via CLI or Docker. An overview of the AUCMEDI environments is illustrated in Figure 5.4. For intuitive and simple usage, the API of the framework is based on three pillars for obtaining general information from the dataset, building the deep learning model, and loading as well as preprocessing any images or volumes. Whereas the Python API allows researchers to build complete MIC pipelines with just a few lines of code through the pillar structure of AUCMEDI, the AutoML module provides fast application via CLI and robust application via Docker. In the two modules, AUCMEDI combines straightforward and fast setup via AutoML with the option of detailed customization via the API as toolbox.

5.2.2 Open-Source Development and Deployment

Similar to the MIScnn project (see Chapter 4.2.2), the aim of the Author was to establish a standardized and community-accepted framework as an open-source project in which the software package has a high development quality, consistent updates as well as further support are provided, contributions are possible without obstacles, and the focus is on usability as well as package stability in multiple environments [273]. As one of the key features of AUCMEDI is to ensure the reusability and reproducibility of MIC pipelines, the project strived to incorporate the MI-CLAIM checklist from Norgeot et al. [354] and the open-source principles stated by Karl Fogel [274].

The following subchapter points out the efforts of the Author to establish a high-quality open-source project.

Philosophy of AUCMEDI

Based on the open-source principles as well as gained experience from the MIScnn project, AUCMEDI has been developed to comply with the following philosophies:

➤ User friendliness

AUCMEDI is an intuitive API designed for human beings, not machines. With a stronger growing interest in medical imaging processing, building MIC pipelines should not be like ‘inventing the wheel’ for every new user. To meet the continuously growing interest in medical imaging processing, AUCMEDI provides consistent and simple APIs for minimizing the number of user actions required for common use cases.

➤ Modularity

The general steps in an MIC pipeline are identical for the majority of projects. Nevertheless, switching to another neural network architecture or dataset format breaks most of the publicly available MIA software, today. AUCMEDI attempts to change this situation. In particular, data I/O, pre-/postprocessing methods, metrics, and model architectures are interfaces that are easily interchangeable.

➤ Easy extensibility

Contributions are simple to integrate into the AUCMEDI pipeline. AUCMEDI provides interfaces and abstract base classes for almost all types of classes or functions in its API. That facilitates defining the structure as well as setup of third-party code and results in easy integration of architectures, subfunctions, or adapting AUCMEDI to new data structures.

➤ Applicability

The dual-environment of the AUCMEDI allows not only the utilization of the Python API as an intuitive toolkit for deep learning researchers but also straightforward usability in clinical research. Besides the API toolkit, the framework also provides AutoML for fast application, robust integration in workflows, and secure deployment in sensitive IT infrastructures.

Open-Source Development

Due to development being a continuing process in open-source projects, it is essential to establish as well as sustain a high development quality [274]. As already introduced in Chapter 4.2.2, the version control system Git is the gold-standard and backbone of any open-source project. The proposed framework AUCMEDI is centralized in a GitHub repository. GitHub hosts not only a Git based version control system but also offers various integrated features like an issue and pull request system, further infrastructure for hosting development operations (DevOps) as well as complete websites, project management techniques, and structured package releases [219]. The complete development process has been done in a Git repository

frankkramer-lab / **aucmedi** Public

Notifications Fork 0 Star 10

Code Issues 47 Pull requests Discussions Actions Security Insights

master 3 branches 7 tags Go to file Code - About

muellerdo build: Updated to version 0.7.2 ✓ 73304f7 2 hours ago 714 commits

.github	ci(PRs): added code quality CI to PRs targeting ...	last month
aucmedi	style(DataGenerator): formatting of docstring	4 hours ago
docs	docs(Example): added multiple API examples an...	2 hours ago
examples	docs(Example): added multiple API examples an...	2 hours ago
tests	test(XAI): added test for xai_decoder with direct...	2 months ago
.gitignore	Updated .gitignore	5 months ago
CITATION.cff	docs(Reference): Added ORCID for Simone May...	4 months ago
Dockerfile	feat(AutoML): dockerized AUCMEDI AutoML - cl...	3 months ago
LICENSE.md	More general software structuring	2 years ago
README.md	docs(README): removed getting started chapte...	2 months ago
mkdocs.yml	docs(Contributing): enhanced contributing guid...	2 months ago
requirements.txt	build(matplotlib): maxed to v3.5.3 due to plotnin...	4 hours ago
setup.py	build: Updated to version 0.7.2	2 hours ago

README.md

AUCMEDI a framework for **Automated Classification of Medical Images**

python 3.8 | 3.10 build passing coverage 96% docs online pypi v0.7.1 downloads 157/month license GPL-3.0

Readme GPL-3.0 license Cite this repository 10 stars 3 watching 0 forks

Releases 7

aucmedi_v0.7.1 Latest on 28 Jul + 6 releases

Packages 1

aucmedi

Figure 5.5: Starting page of the AUCMEDI repository on GitHub.

which was released to the public at the end of May 2022. However, the first closed-source release on PyPi was in March 2021. The AUCMEDI code is sorted into three types of branches: The master branch containing the current stable release, the development branch containing the current working state of MIScnn including work-in-progress submodules, the ‘gh-pages’ containing the documentation as well as the website, and optional remaining branches for individual feature development until being merged in the development branch with a pull request. Features and further updates are managed with the issue system of the repository which also allows other users or contributors to report bugs, feature requests as well as open questions.

The AUCMEDI framework also integrated Git commit conventions for development which assert human and machine readability in Git commit messages. The utilized Conventional Commits specification is one of the most widespread Git commit conventions and provides a light set of rules for an explicit Git commit history [355]. Besides the improved readability of Git commits from contributors that are often minimalistic providing no insights into the contributed changes, the convention enables tools for automated analysis of development history, as well.

Documentation

The provision of a wiki for APIs and example applications is the most common form for documenting a software framework [274, 278]. In order to encourage community interest and ensure the reusability of a framework, extensive documentation is fundamental for any open-source project. Further advantages of well-written documentation are listed in Chapter 4.2.2. In contrast to the MIScnn framework which utilizes the integrated GitHub wiki with manually written entries, AUCMEDI utilizes the MkDocs site generator [356]. MkDocs allows fast and simple documentation building by assembling a full but easily adjustable website in which entries are encoded in the markdown file format. The main advantage of MkDocs is that it can be extended with plugins for automated generation of documentation entries. Still, the deployed MkDocs based website for AUCMEDI is hosted on GitHub Pages [219] to center all project services at a single infrastructure provider. The website acts as a hub for the complete AUCMEDI project and provides documentation about the general workflow, installation procedure, getting started guides, examples, tutorials, and a complete API reference of the API and the AutoML module. In addition, the website contains information about the project itself, funding, current development status, contributors, and references on how to cite the work.

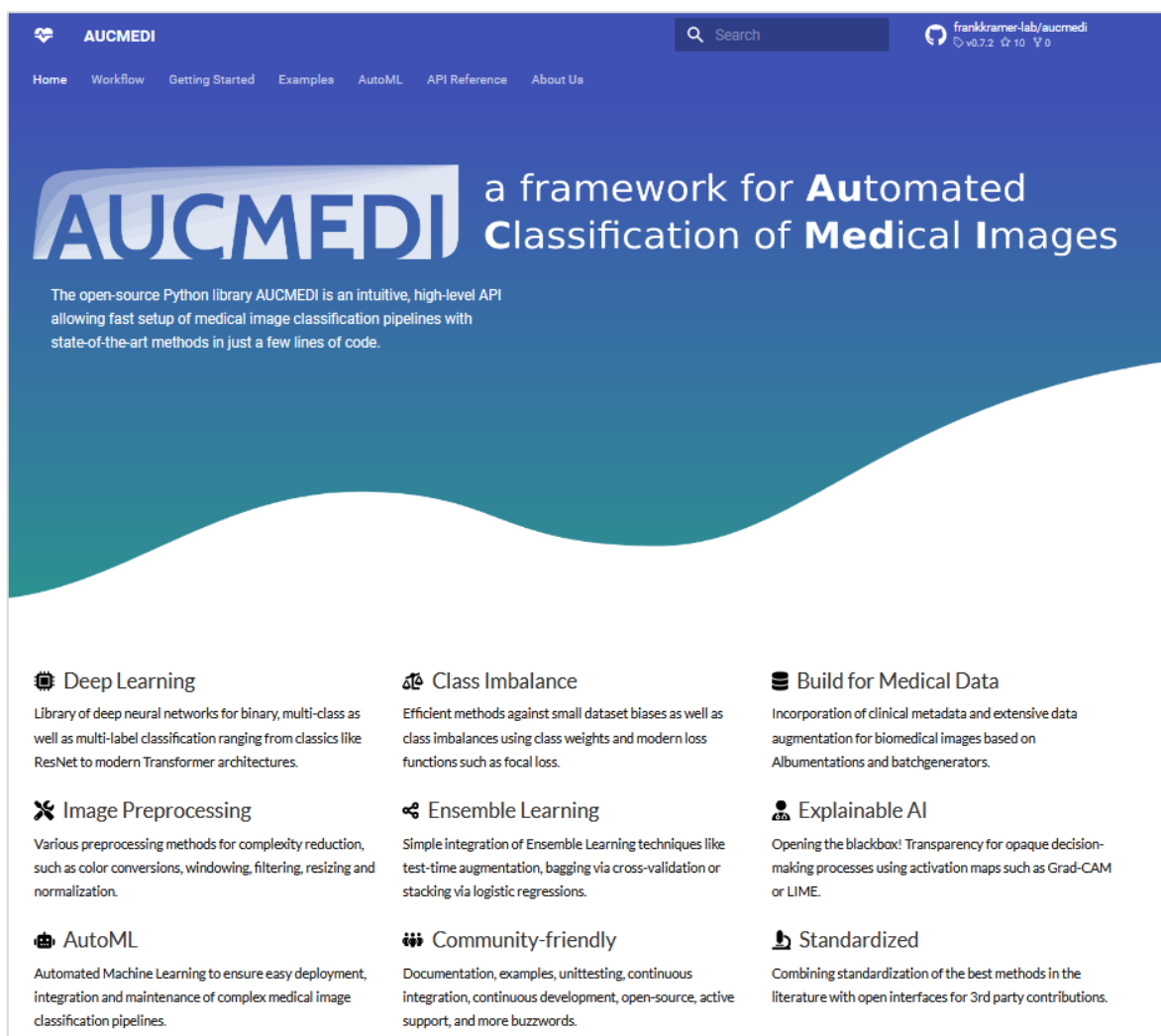
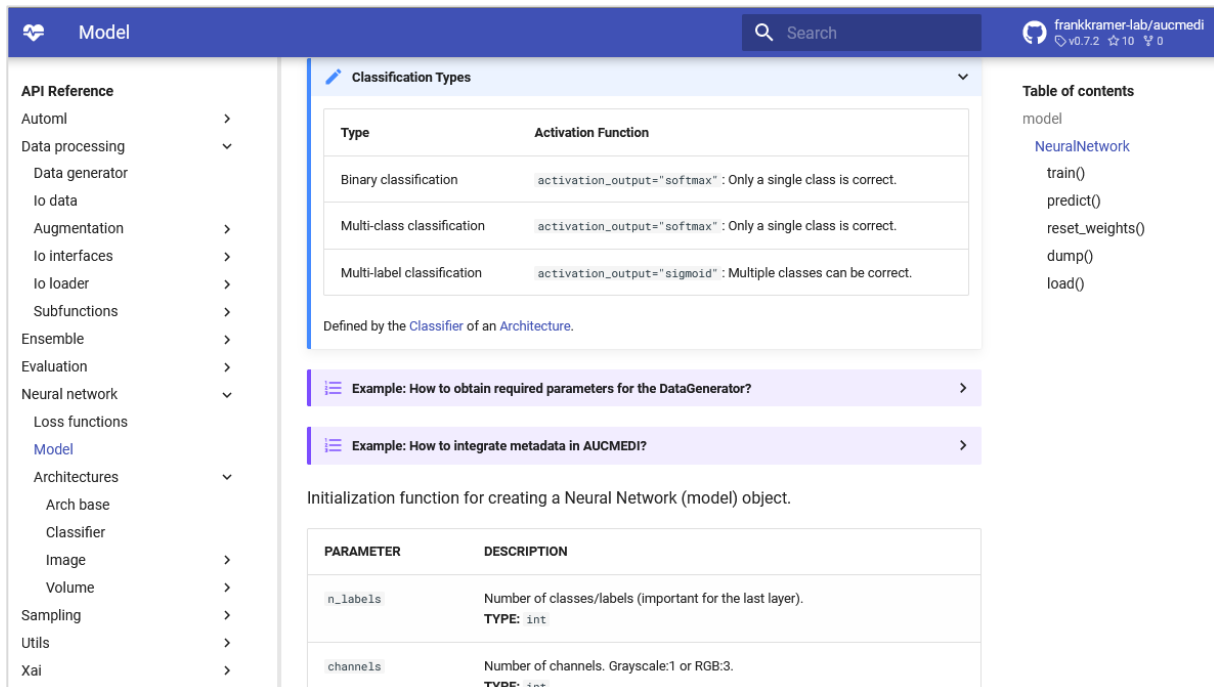


Figure 5.6: Website of the AUCMEDI framework hosted on GitHub Pages.

Through the integration of the plugin `mkdocstring` from Mazzucotelli et al. [357] for `MkDocs`, it is possible to automatically generate documentation entries by processing docstrings in the source code. A docstring is a comment block describing a code segment and acting as a manual. The difference to standard code comments is that docstrings can be queried at runtime during the development process within an integrated development environment. AUCMEDI incorporated docstrings for the entire code base based on the Google Python Style Guide convention for docstrings [358]. Through the incorporation of docstrings and the integration of `mkdocstring`, the AUCMEDI wiki offers documentation for all functions as well as classes in the package. The entries contain function parameters, short usage examples, as well as further details or warnings. This allows a comprehensive, continuously updated, and standardized documentation of the project resulting in especially high documentation quality. An example entry of the AUCMEDI wiki is illustrated in Figure 5.7.



The screenshot shows the AUCMEDI wiki page for the 'Model' section. The page is titled 'Model' and has a search bar and a user profile for 'frankkramer-lab/aucmedi'. The main content is under the heading 'Classification Types' and includes a table with the following data:

Type	Activation Function
Binary classification	<code>activation_output="softmax"</code> : Only a single class is correct.
Multi-class classification	<code>activation_output="softmax"</code> : Only a single class is correct.
Multi-label classification	<code>activation_output="sigmoid"</code> : Multiple classes can be correct.

Below the table, it states 'Defined by the Classifier of an Architecture.' There are two example links: 'Example: How to obtain required parameters for the DataGenerator?' and 'Example: How to integrate metadata in AUCMEDI?'. The page also includes a 'Table of contents' on the right with links to 'model', 'NeuralNetwork', 'train()', 'predict()', 'reset_weights()', 'dump()', and 'load()'. At the bottom, there is a table for initialization parameters:

PARAMETER	DESCRIPTION
<code>n_labels</code>	Number of classes/labels (important for the last layer). TYPE: <code>int</code>
<code>channels</code>	Number of channels. Grayscale:1 or RGB:3. TYPE: <code>int</code>

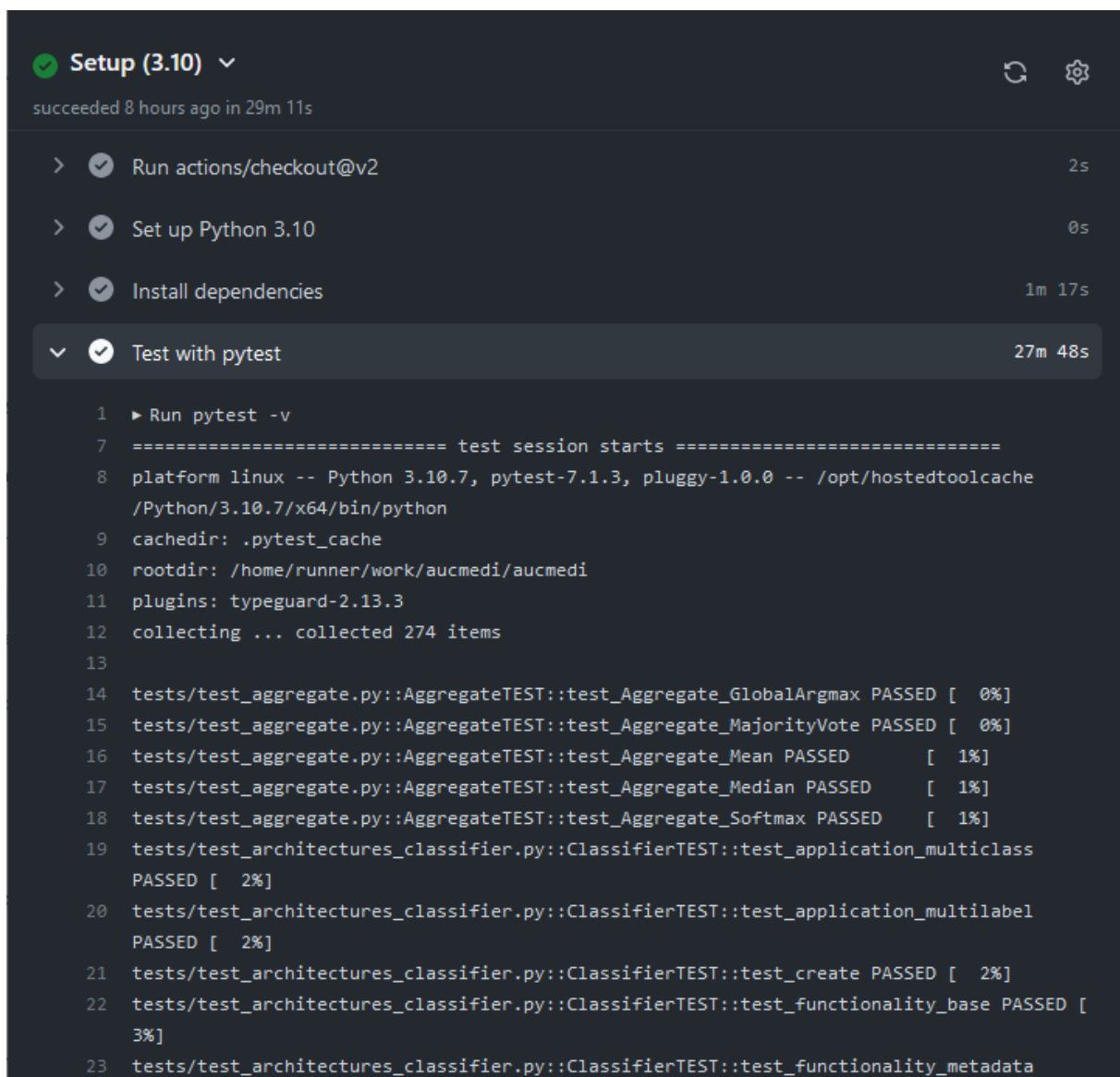
Figure 5.7: Extract of the AUCMEDI wiki on the project website.

Next to the wiki, examples as well as tutorials are provided on the AUCMEDI website. As previously stated, implementations or usage examples of open-source software are often the first steps for new users or researchers [278]. In the AUCMEDI project, examples are grouped in the following categories: Quick start tutorials for the API and the AutoML module (2x), an intensively documented example of a complete pipeline for the API (1x), a standardized application of the API on 8 medical imaging modalities (10x) to demonstrate broad adaptability that is described in detail in Chapter 5.3, multiple tutorials for individual components of the AUCMEDI framework (11x), AutoML examples with Docker and CLI (2x), and references to studies utilizing AUCMEDI (5x). In total, the website provides 32 examples in order to establish a comprehensive pool of demonstrations on how to apply AUCMEDI.

Continuous Integration

The concept of continuous integration is fundamental in modern open-source projects and describes the principles of automatic rebuilding, testing, and evaluation of the software to ensure functionality [279]. An in detail definition of CI and unit testing can be found in Chapter 4.2.2 - Continuous Integration and in the work by Karl Fogel [274].

The AUCMEDI framework utilizes extensive unit testing to ensure consistent functionality and high software quality. This allows more efficient reviewing and robust integration of community contributions without risking software stability introduced by flawed code. The unit testing consists of general pipeline testing with multiple types as well as formats of imaging data, thorough parameter testing of the AUCMEDI pillars, individual function testing, and edge case testing for the API as well as for the AutoML module. In total, 274 unit tests were implemented. An overview of unit tests grouped by submodules in AUCMEDI is summarized



```
Setup (3.10) ✓
succeeded 8 hours ago in 29m 11s

> ✓ Run actions/checkout@v2 2s
> ✓ Set up Python 3.10 0s
> ✓ Install dependencies 1m 17s
v ✓ Test with pytest 27m 48s

1 ▶ Run pytest -v
7 ===== test session starts =====
8 platform linux -- Python 3.10.7, pytest-7.1.3, pluggy-1.0.0 -- /opt/hostedtoolcache
  /Python/3.10.7/x64/bin/python
9 cachedir: .pytest_cache
10 rootdir: /home/runner/work/aucmedi/aucmedi
11 plugins: typeguard-2.13.3
12 collecting ... collected 274 items
13
14 tests/test_aggregate.py::AggregateTEST::test_Aggregate_GlobalArgmax PASSED [ 0%]
15 tests/test_aggregate.py::AggregateTEST::test_Aggregate_MajorityVote PASSED [ 0%]
16 tests/test_aggregate.py::AggregateTEST::test_Aggregate_Mean PASSED [ 1%]
17 tests/test_aggregate.py::AggregateTEST::test_Aggregate_Median PASSED [ 1%]
18 tests/test_aggregate.py::AggregateTEST::test_Aggregate_Softmax PASSED [ 1%]
19 tests/test_architectures_classifier.py::ClassifierTEST::test_application_multiclass
  PASSED [ 2%]
20 tests/test_architectures_classifier.py::ClassifierTEST::test_application_multilabel
  PASSED [ 2%]
21 tests/test_architectures_classifier.py::ClassifierTEST::test_create PASSED [ 2%]
22 tests/test_architectures_classifier.py::ClassifierTEST::test_functionality_base PASSED [
  3%]
23 tests/test_architectures_classifier.py::ClassifierTEST::test_functionality_metadata
```

Figure 5.8: Extract of the CI pipeline logs for AUCMEDI on GitHub.

in Table 5.2. The building of AUCMEDI is tested not only based on predefined module versions of dependency packages like TensorFlow to ensure package reproducibility but also based on the newest versions of dependency packages to ensure consistent package stability.

Table 5.2: Distribution of unit tests for the AUCMEDI framework.

Submodule	Unit Tests
IO Interfaces	8
IO Loader	18
DataGenerator	12
Image Preprocessing	14
Image Augmentation	9
Neural Network Model	4
Loss Functions	4
Classifier Head	6
Architectures for 2D	32
Architectures for 3D	15
Sampling	9
Evaluation	24
Ensemble Learning	27
Aggregate Functions	5
Metalearner	20
Explainable Artificial Intelligence	33
AutoML Building Blocks	21
AutoML Command Line Interface	9
Utilities	4

Next to the code functionality, the AUCMEDI CI pipeline also analyzes the code quality. For unit testing coverage assessment to evaluate what percentage of the framework is tested and what percentage is potentially deficient, the Codecov platform was utilized [281]. In the proposed framework, 6,555 code lines from a total of 6,823 lines (without documentation and formatting lines) are covered by unit testing (date: 29.09.2022) which results in a coverage of 96.07%. Such high unit testing coverage implies strong software robustness and consistent functionality. Another type of code quality evaluation is compliance with agreed coding conventions. AUCMEDI automatically inspects Git commit messages in added or contributed code to comply with the incorporated Git commit convention. Further convention analysis like docstring and general code style is strongly encouraged and utilized in AUCMEDI but is currently not mandatory. However, the integration of enforced code style and formatting conventions like black [359], flake8 [360], or pylint [361] are planned as future work.

The complete CI pipeline of AUCMEDI is hosted in the GitHub infrastructure [219]. This allows direct integration in the version control system as well as in the pull request system of GitHub in order to strengthen reflected reviewing procedures. Also, the deployment of CI pipelines in a single infrastructure host drastically reduces the integration complexity of CI pipeline result collection from multiple infrastructures.

Continuous Delivery and Package Availability

Similar to CI, the process of continuous delivery (CD) describes a pipeline in which, after successful testing by CI strategies, the software is automatically built, packaged, and delivered to one or multiple production environments [362, 363]. The automation of the delivery process offers persistent releases for users across different delivery platforms as well as reduces time-consuming manual tasks [363]. AUCMEDI utilizes continuous delivery strategies to automatically generate standardized releases which are published in three delivery platforms: A GitHub release [219] integrated directly into the Git repository for obtaining the associated source code, a release in the Python Package Index (PyPI) [364] for deployment of the API as well as AutoML CLI module in a Python environment, and a release in the GitHub Container Registry [219] for deployment of the AutoML Docker module in a sensitive infrastructure. Therefore, the AUCMEDI framework can be directly installed using `pip install aucmedi` for the API and `docker pull ghcr.io/frankkramer-lab/aucmedi:latest` for the AutoML module.

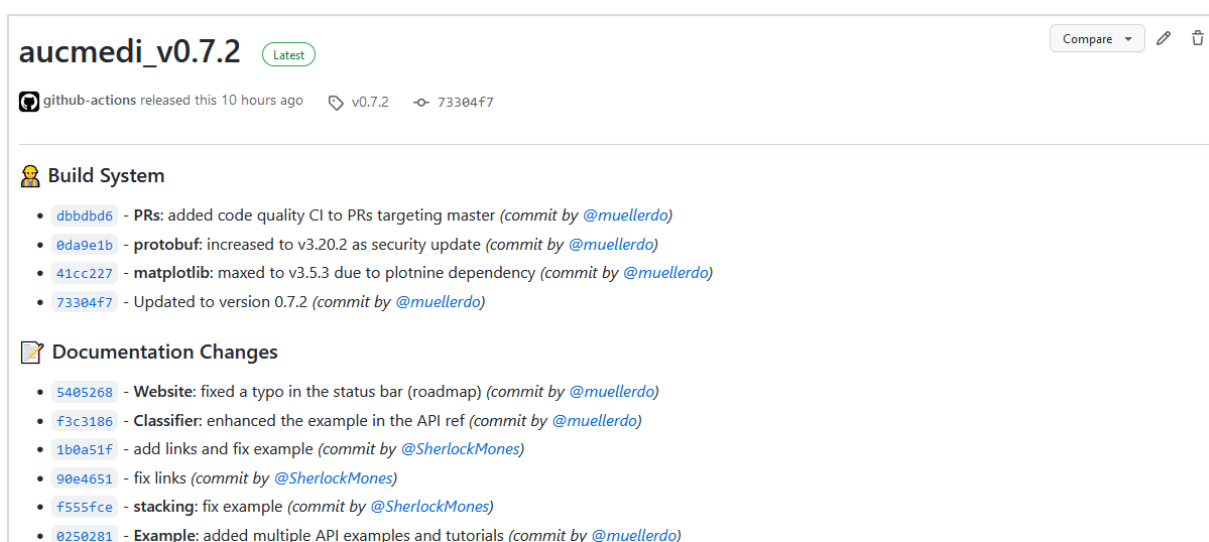


Figure 5.9: Extract of a release changelog of AUCMEDI on GitHub.

Besides the CD of the framework, the corresponding documentation of AUCMEDI is also automatically updated and deployed. For each release, the website including the complete API reference is updated, rebuilt, and uploaded to GitHub Pages. Furthermore, through the utilization of the machine-readable Git commit conventions, the Git commit history of the release is automatically categorized (in groups like documentation-related changes or newly introduced features) in order to generate a full changelog for the release which allows detailed provenance.

The website of AUCMEDI including general information, documentation, and links to all other resources is available at: <https://frankkramer-lab.github.io/aucmedi/>.

The source code and the associated CI pipelines are available in the Git repository: <https://github.com/frankkramer-lab/aucmedi>. AUCMEDI is licensed under the open-source GNU General Public License Version 3.

5.2.3 API Usage

The framework AUCMEDI is a toolkit and AutoML software for medical image classification, and, thus, can be utilized as an API in a Python environment for custom pipeline building as well as to automatically build a pipeline for straightforward application. This subchapter describes the installation process, the documentation of the pillars of AUCMEDI, and a simple example of how to use AUCMEDI as API and as AutoML software.

Installation

There are three options to install AUCMEDI. For utilization of the API as well as the AutoML module, the recommended way of installation is using PyPI for which the installation manager `pip` provides a fully automatic and simplistic installation procedure of package dependencies. Another option for installing both modules is the manual installation from the source code. This can be achieved by downloading AUCMEDI from the Git repository (GitHub) and installing it by calling the installation script `setup.py`. In contrast to `pip`, the manual installation can not handle more complex dependency issues as well as does not support virtual environments [283]. For utilization of only the AutoML module, the Docker container of AUCMEDI can be installed from the GitHub Container Registry which allows integration into sensitive infrastructure. All installation options are demonstrated in Code Snippet 5.1.

```
1 # Installation via PyPI
2 pip install aucmedi
3
4 # Installation via Docker
5 docker pull ghcr.io/frankkramer-lab/aucmedi:latest
6
7 # Installation from Source
8 git clone https://github.com/frankkramer-lab/aucmedi
9 cd aucmedi
10 python setup.py install
```

Code Snippet 5.1: Installation process of the proposed AUCMEDI framework.

Despite the automatic dependency installation, it is recommended to setup a TensorFlow distribution on the system prior to AUCMEDI installation in order to ensure full library functionality. The reason for this is that TensorFlow relies on operational system drivers like CUDA [220]. For Docker usage, GPU support and accessibility have to be enabled within Docker containers to utilize the full potential of AUCMEDI.

Pillars of AUCMEDI

As previously described in Chapter 5.2.1, the AUCMEDI pipeline is represented by three pillars: The data interface, the neural network model, and the data generator. These pillars handle all required steps for deep learning based medical image classification and can be extensively customized through open interfaces as well as high configurability.

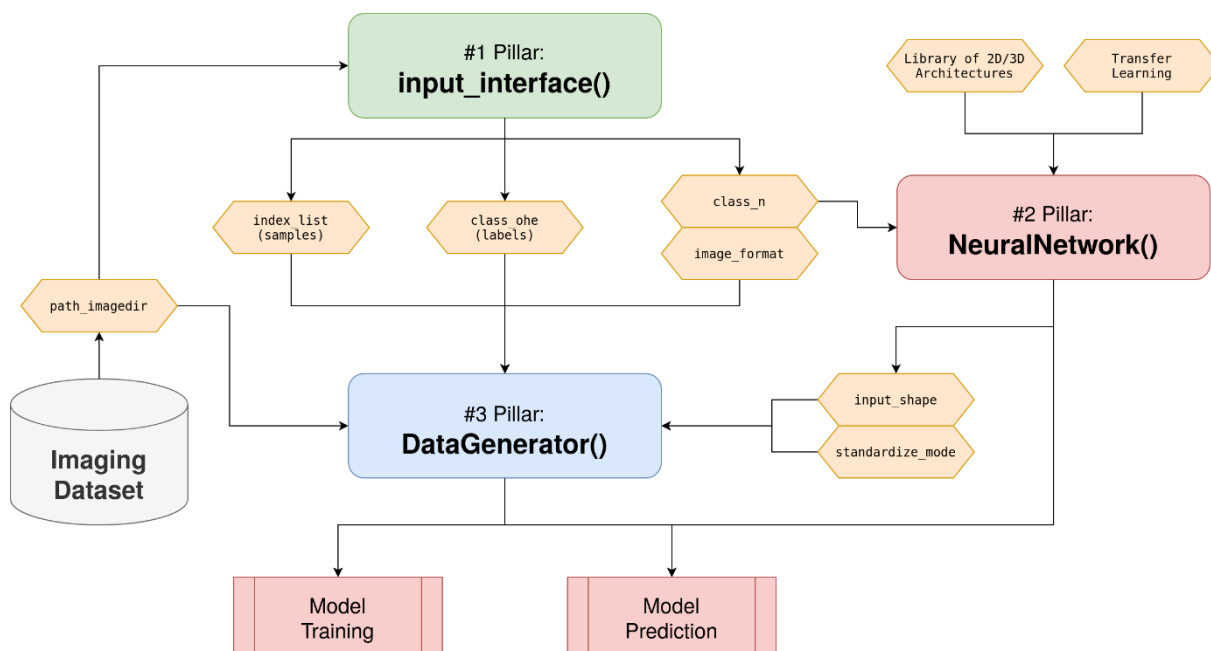


Figure 5.10: Workflow of the three pillars in AUCMEDI.

➤ #1 Pillar: Data Interface

In order to train a convolutional neural network, data and labels have to be loaded into the MIC pipeline. AUCMEDI provides the data interface for this purpose. With the data interface, various 2D and 3D image formats can be loaded and annotations extracted by either automatic inference based on the file structure, from a CSV, or from a JSON file.

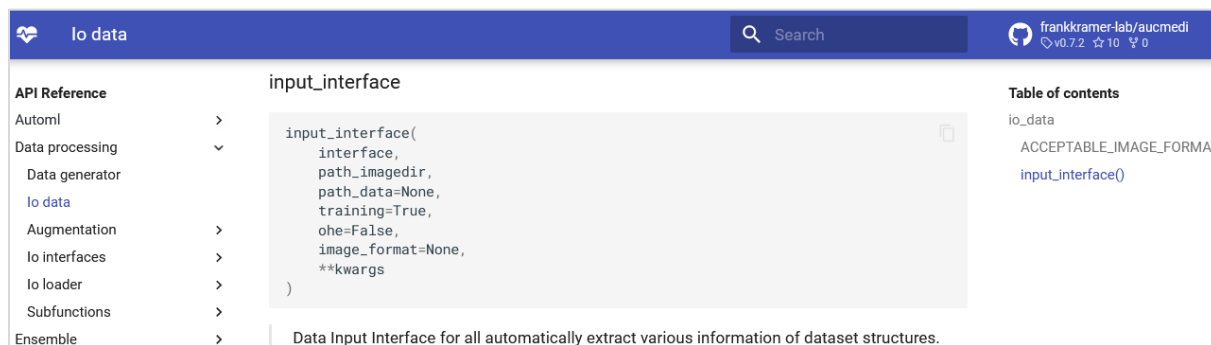


Figure 5.11: Extract of the documentation entry for the data interface pillar from the AUCMEDI wiki.

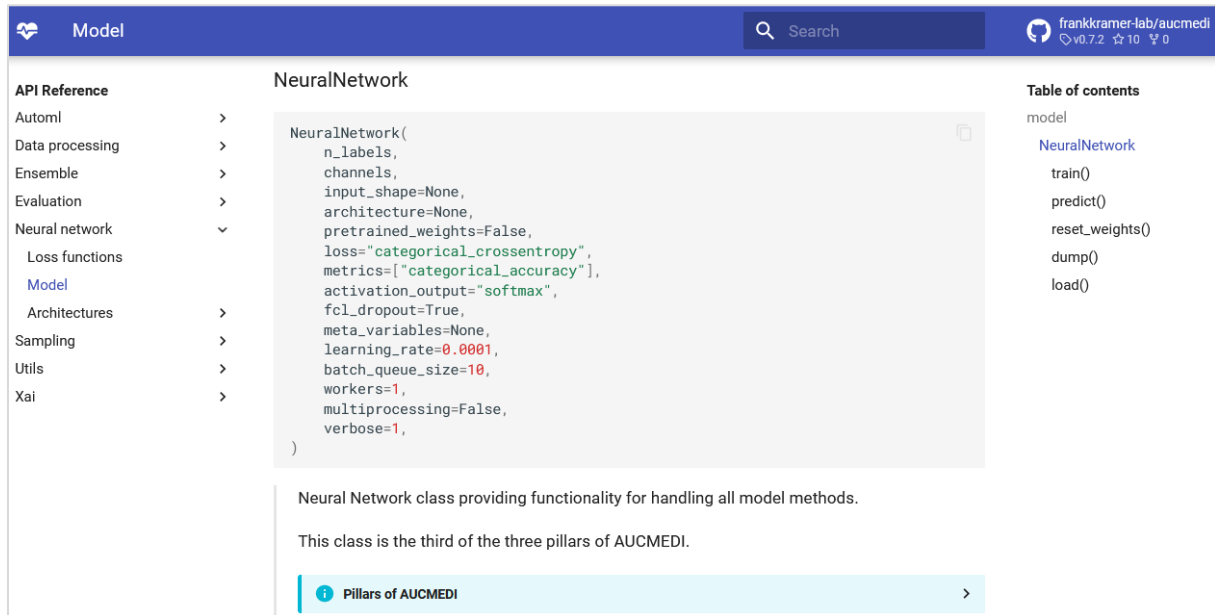
The data interface not only extracts the dataset annotations but also important metadata like image file formats or classification information which can be parsed to the data generator and the neural network model.

Table 5.3: Arguments of the data interface pillar in AUCMEDI.

Argument	Type	Default	Description
path_imagedir	String	REQUIRED	Path to the directory containing the images.
interface	String	REQUIRED	String defining format interface for loading/storing data.
path_data	String	None	Path to the index/class annotation file if required. (csv/json)
training	Boolean	True	Boolean option whether annotation data is available.
ohe	Boolean	False	Boolean option whether annotation data is sparse categorical or one-hot encoded.
image_format	String	None	Force to use a specific image format. By default, image format is determined automatically.
**kwargs	Dictionary	{}	Additional parameters for the format interfaces.

➤ #2 Pillar: Neural Network

The neural network class is the second pillar of AUCMEDI and the interface to the deep learning model of the pipeline. The neural network is a high-level API for TensorFlow optimized for building a medical image classifier through a single class. The class inherits some basic data as well as annotation information from the data interface and builds the model setup based on the selected architecture, loss function, and input shape.



The screenshot shows the documentation for the `NeuralNetwork` class. The class signature is as follows:

```
NeuralNetwork(
    n_labels,
    channels,
    input_shape=None,
    architecture=None,
    pretrained_weights=False,
    loss="categorical_crossentropy",
    metrics=["categorical_accuracy"],
    activation_output="softmax",
    fcl_dropout=True,
    meta_variables=None,
    learning_rate=0.0001,
    batch_queue_size=10,
    workers=1,
    multiprocessing=False,
    verbose=1,
)
```

The class description states: "Neural Network class providing functionality for handling all model methods. This class is the third of the three pillars of AUCMEDI."

The table of contents lists the following methods: `train()`, `predict()`, `reset_weights()`, `dump()`, and `load()`.

Figure 5.12: Extract of the documentation entry for the neural network pillar from the AUCMEDI wiki.

Next to the design and configuration of the deep learning model, the neural network class acts as the hub of the complete pipeline. It provides functionality for handling all model methods which includes running training and prediction processes.

Table 5.4: Arguments of the neural network pillar in AUCMEDI.

Argument	Type	Default	Description
<code>n_labels</code>	Integer	REQUIRED	Number of classes/labels (important for the last layer).
<code>channels</code>	Integer	REQUIRED	Number of channels. Grayscale:1 or RGB:3.
<code>input_shape</code>	Shape	None	Input shape of the batch imaging data (including channel axis).
<code>architecture</code>	Architecture	Vanilla	Neural network model Architecture class instance. By default, a 2D Vanilla Model is used as architecture.
<code>pretrained_weights</code>	Boolean	False	Option whether to utilize pretrained weights e.g. from ImageNet.
<code>loss</code>	Metric	cross-entropy	The metric function which is used as loss for training.
<code>metrics</code>	Metrics	None	List of one or multiple Metric Functions, which will be shown during training.
<code>activation_output</code>	String	Softmax	Activation function which should be used in the classification head.
<code>fcl_dropout</code>	Boolean	True	Option whether to utilize an additional Dense & Dropout layer in the classification head.
<code>meta_variables</code>	Integer	None	Number of metadata variables, which should be included in the classification head.
<code>learning_rate</code>	Float	0.0001	Learning rate in which weights of the neural network will be updated.
<code>batch_queue_size</code>	Integer	10	The batch queue size is the number of previously prepared batches in the cache during runtime.
<code>workers</code>	Integer	1	Number of workers/threads which preprocess batches during runtime.
<code>multiprocessing</code>	Boolean	False	Option whether to utilize multi-processing for workers instead of threading.
<code>verbose</code>	Integer	1	Option (0/1) how much information should be written to stdout.

➤ #3 Pillar: Data Generator

In order to utilize the neural network for training or prediction processes, it is required to pass data to the model. The data generator is a powerful interface for the complete processing of data ranging from image loading to returning model-ready batches. Thus, the data generator can be configured according to the desired processing workflow and passed to the neural network training or prediction function.

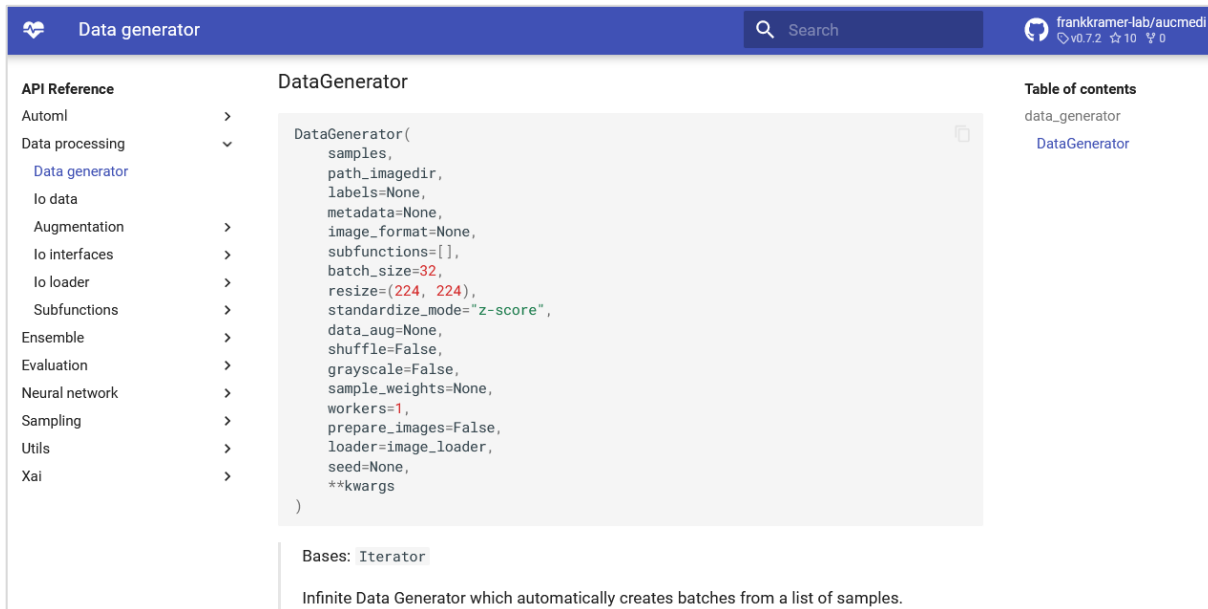


Figure 5.13: Extract of the documentation entry for the data generator pillar from the AUCMEDI wiki.

The data generator provides the following functionalities: Image loading from multiple file formats, application of preprocessing functions (subfunctions), automated resizing and standardization according to the selected architecture recommendations (fixed for transfer learning approaches), application of on-the-fly image augmentation, prior preprocessing of images to avoid CPU bottlenecks, and stacking preprocessed images together into batches.

Table 5.5: Arguments of the data generator pillar in AUCMEDI.

Argument	Type	Default	Description
samples	List of Strings	REQUIRED	List of sample/index encoded as Strings. Provided by the Input Interface.
path_imagedir	String	REQUIRED	Path to the directory containing the images.
labels	NumPy Array	None	Classification list with One-Hot Encoding. Provided by the Input Interface.
metadata	NumPy Array	None	NumPy Array with additional metadata. Shape: (n_samples, meta_variables).
image_format	String	None	Image format to add at the end of the sample index for image loading.
subfunctions	Subfunctions	None	List of Subfunctions class instances which will be SEQUENTIALLY executed on the data set.
batch_size	Integer	32	Number of samples inside a single batch.
resize	Shape	224x224	Resizing shape consisting of an X and Y size. (optional Z size for Volumes)
standardize_mode	String	Z-Score	Standardization modus in which image intensity values are scaled.
data_aug	Augmentation	None	Data Augmentation class instance which performs diverse augmentation techniques.
shuffle	Boolean	False	Boolean, whether dataset should be shuffled.
grayscale	Boolean	False	Boolean, whether images are grayscale or RGB.
sample_weights	List of Floats	None	List of weights for samples.
workers	Integer	1	Number of workers.
prepare_images	Boolean	False	Boolean, whether all images should be prepared and backup to disk before training.
loader	IO Loader	Image Loader	Function for loading samples/images from disk.
seed	Integer	None	Seed to ensure reproducibility for random function.
**kwargs	Dictionary	None	Additional parameters for the sample loader.

Basic API Usage

In the following subchapter, a simple example demonstrates how to use the API of AUCMEDI for building modern deep learning based MIC pipelines in less than 25 lines of code. As introduced in the previous subchapter, an MIC pipeline can be built with the three pillars of AUCMEDI: The data interface, the neural network, and the data generator. The pillar interface of AUCMEDI combines an intuitive and straightforward setup with the possibility of detailed custom specifications.

```

1 # AUCMEDI Library
2 from aucmedi import *
3
4 # Pillar #1: Initialize input data reader
5 ds = input_interface(interface="csv",
6                     path_imagedir="/home/muellerdo/COVdataset/ct_slides/",
7                     path_data="/home/muellerdo/COVdataset/classes.csv",
8                     ohe=False, # OHE short for one-hot encoding
9                     col_sample="ID", col_class="PCRpositive")
10 (index_list, class_ohe, nclasses, class_names, image_format) = ds
11
12 # Pillar #2: Initialize a DenseNet121 model with ImageNet weights
13 model = NeuralNetwork(n_labels=nclasses, channels=3,
14                      architecture="2D.DenseNet121",
15                      pretrained_weights=True)

```

Code Snippet 5.2: Foundation setup of a medical image classification pipeline with AUCMEDI.

As demonstrated in Code Snippet 5.2, the data interface provides crucial information about the dataset which is passed to the neural network. After initializing the two pillars, the returning model object represents the foundation of an MIC pipeline. With the AUCMEDI model, it is possible to run training and prediction processes. As shown in Code Snippet 5.3, the model is

```

1 # Pillar #3: Initialize training Data Generator for first 1000 samples
2 train_gen = DataGenerator(samples=index_list[:1000],
3                          path_imagedir="/home/muellerdo/COVdataset/ct_slides/",
4                          labels=class_ohe[:1000],
5                          image_format=image_format,
6                          resize=model.meta_input,
7                          standardize_mode=model.meta_standardize)
8 # Run model training with Transfer Learning
9 model.train(train_gen, epochs=20, transfer_learning=True)
10
11 # Pillar #3: Initialize testing Data Generator for 500 samples
12 test_gen = DataGenerator(samples=index_list[1000:1500],
13                          path_imagedir="/home/muellerdo/COVdataset/ct_slides/",
14                          labels=None,
15                          image_format=image_format,
16                          resize=model.meta_input,
17                          standardize_mode=model.meta_standardize)
18 # Run model inference for unknown samples
19 preds = model.predict(test_gen)
20
21 # preds <-> NumPy array with shape (500,2)
22 # -> 500 predictions with softmax probabilities for our 2 classes

```

Code Snippet 5.3: AUCMEDI pipeline utilization for training and prediction.

used for training on the first 1,000 samples from the dataset and, afterward, used for predicting the classification of the remaining 500 samples. This example reveals the simplicity of building and using a state-of-the-art MIC pipeline with the AUCMEDI API.

Basic AutoML Usage

In contrast to manual building pipelines for MIC, AUCMEDI also offers the AutoML module. In this subchapter, an example demonstrates how to use the CLI as well as Docker AutoML of AUCMEDI for building and utilizing modern deep learning based MIC pipelines. As introduced in Chapter 5.2.1 - Automated Machine Learning, the AUCMEDI AutoML module offers three modes which are illustrated in Figure 5.14: The training, prediction, and evaluation mode. The AutoML module of AUCMEDI enables straightforward application without the need for extensive deep learning knowledge, integration into clinical workflows, and maintenance of complex models for MIC.

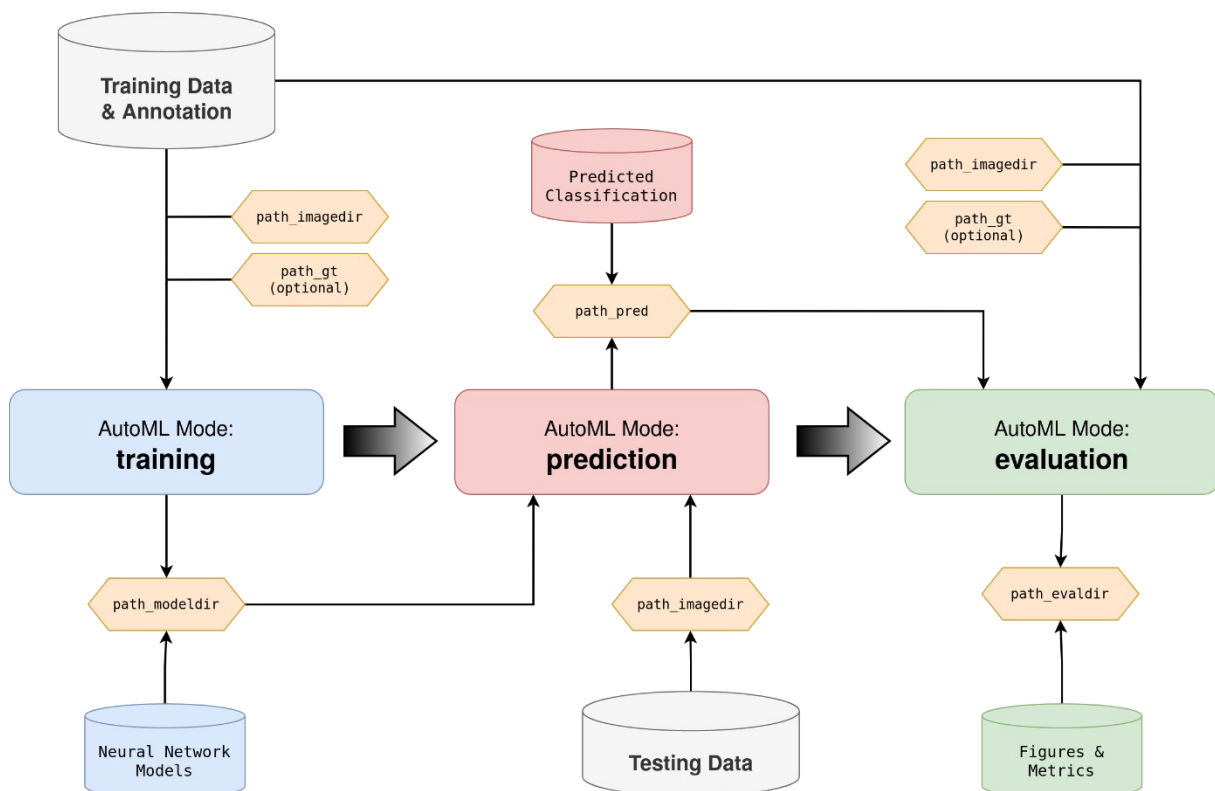


Figure 5.14: Workflow of the AutoML module in AUCMEDI.

As demonstrated in Code Snippet 5.4, the CLI for AUCMEDI AutoML allows training a modern DenseNet121 based pipeline for MIC in a single call in which the training data is by default expected to be in the current working directory. The training mode creates a model

```

1 # Run training with default arguments, but a specific architecture
2 aucmedi training --architecture "DenseNet121"
3
4 # Run prediction with default arguments
5 aucmedi prediction

```

Code Snippet 5.4: AUCMEDI AutoML utilization through the command line interface.

directory which is utilized in the prediction mode in which the testing data is also expected to be in the current working directory by default. The Docker approach for AUCMEDI AutoML utilizes the same commands equivalent to the CLI which is demonstrated in Code Snippet 5.5. However, the default data locations for the Docker interface are inside the container which is why the data directories have to be mounted into the virtualization environments via so-called Docker volumes [350].

```

1 # Run training with default arguments, but a specific architecture
2 docker run \
3   -v /home/dominik/aucmedi.data:/data \
4   --rm \
5   ghcr.io/frankkramer-lab/aucmedi:latest \
6   training \
7   --architecture "DenseNet121"
8
9 # Run prediction with default arguments
10 docker run \
11   -v /home/dominik/aucmedi.data:/data \
12   --rm \
13   ghcr.io/frankkramer-lab/aucmedi:latest \
14   prediction

```

Code Snippet 5.5: AUCMEDI AutoML utilization through the Docker interface.

This example demonstrates the simplicity of the AUCMEDI AutoML application for state-of-the-art medical image classification.

5.2.4 Community Contributions, Support, and Popularity

As previously stated, the proposed AUCMEDI framework for standardized as well as automated medical image classification was released closed-source on PyPI in March 2021 and released open-source on GitHub in May 2022. Since then, it has been maintained, supported, and further developed. Currently, AUCMEDI has been mainly utilized in in-house studies

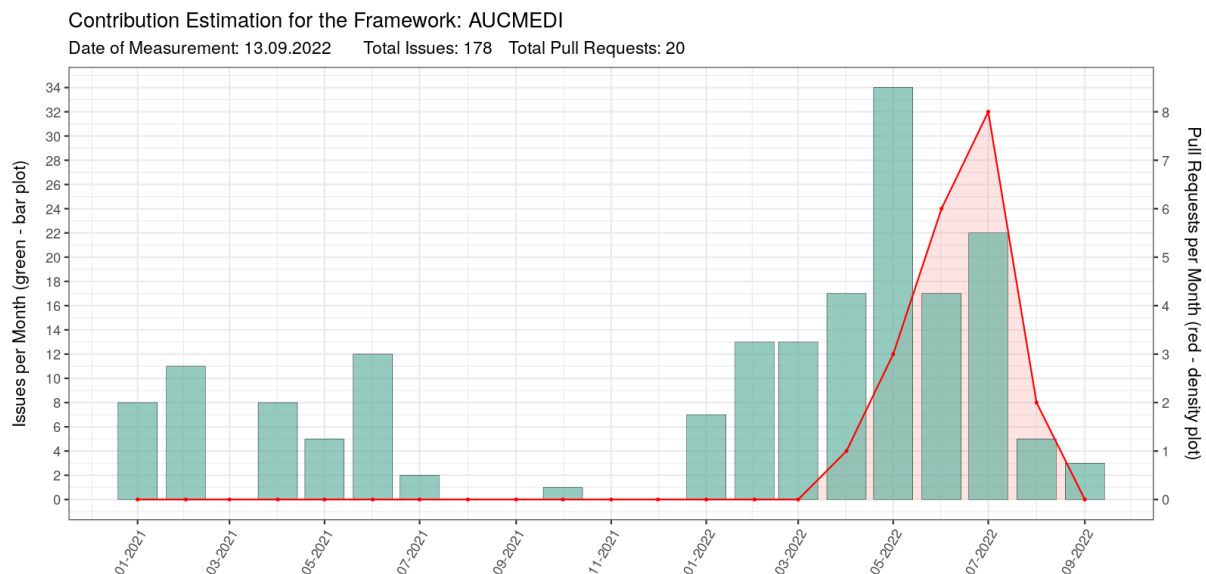


Figure 5.15: Community contribution estimation for the AUCMEDI framework.

which are discussed in the next chapters but also started to slowly gain interest by medical informatics scientists. In order to empirically quantify the prevalence of the proposed framework, the GitHub repository metadata as well as the PyPI download statistics were evaluated. The analysis was equally performed as for the MIScnn framework in Chapter 4.2.4.

For evaluation of the project development and community contributions, the number of issues as well as pull requests were analyzed and visualized in Figure 5.15. The data shows a high number of issues that have been regularly created due to the active agile software development process of AUCMEDI [365]. Close to the open-source release date of AUCMEDI in May 2022, a peak in the number of created issues occurred which indicated an engaging development phase. Similar to the peak of issues, the number of pull requests also started growing and peaking closely after the open-source release. However, the contributions in terms of pull requests as well as issues are currently limited and only associated with in-house contributions or colleagues. The agile development process notably increases transparency but issues as well as pull requests are not suitable as indicators for user contributions due to the indistinguishability between in-house and external contributions.

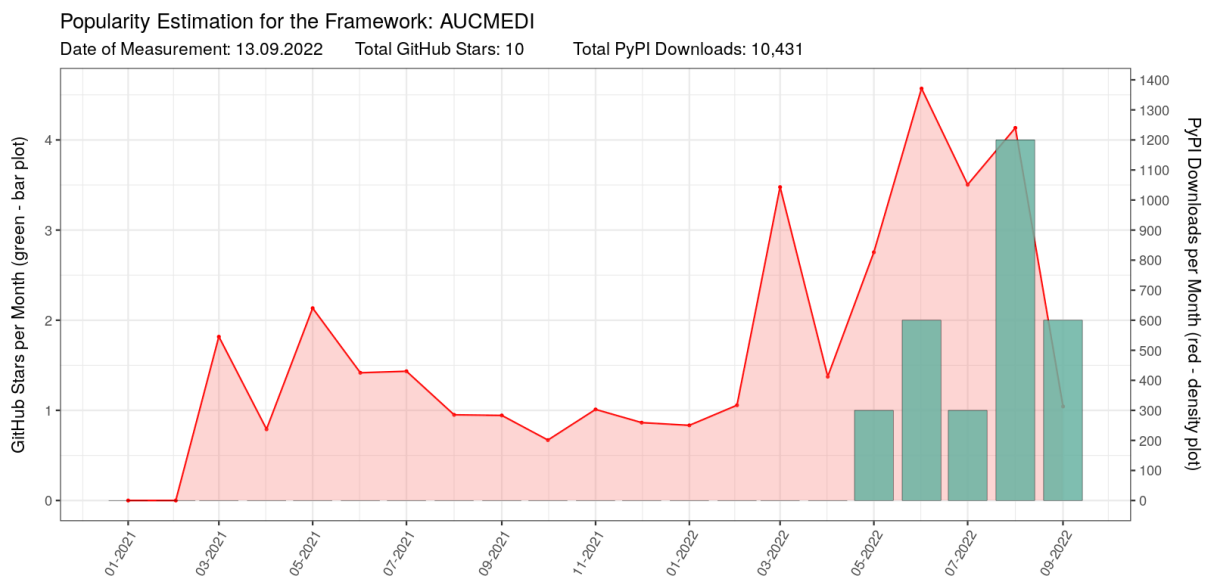


Figure 5.16: Popularity estimation for the AUCMEDI framework.

For popularity estimation of the framework and to approximate the number of users (community size), the GitHub Stars and the number of package downloads from the PyPI statistics [284] were analyzed and visualized in Figure 5.16. In contrast to the applied contribution estimation, the popularity estimation based on PyPI downloads and GitHub Stars is a more direct measurement of community prevalence and quantity. The AUCMEDI project shows an active number of downloads prior to the open-source release due to the utilization of AUCMEDI in multiple studies which have been already publicly available. After the open-source release in May 2022, a strong increase in downloads as well as GitHub Stars has been observed. This implies a slow but steady community growth of the AUCMEDI project.

5.3 Study: Standardized Image Classification across Medical Disciplines

Including deep neural networks in clinical practice remains challenging for several reasons: Collecting and preparing data for a neural network requires both medical as well as technical expertise. Improper preprocessing may cause a bias in the dataset which could ultimately impact the generalization performance [12]. Furthermore, physicians, which are responsible for patient care, are often reluctant to accept a machine-made decision, if the decision is not transparent or comprehensible [366]. Even reliable results generated by a well-performing ‘black box’ can lead to mistrust in continuous reliability [12]. Finally, solving a classification problem with a neural network model often results in complex, isolated solutions. Due to their lack of modularity, such implementations are limited in reusability or even reproducibility.

The proposed framework AUCMEDI aims to simplify such implementations. It allows simple usability resulting in requiring only little domain knowledge and merely basic programming skills. Furthermore, the framework features modularity with regard to network architectures, datasets, and even medical disciplines. The purpose of this study is to demonstrate AUCMEDI's capabilities in terms of standardized application and adaptability. To evaluate the framework, a single pipeline using AUCMEDI was designed and applied to a wide range of image classification tasks from diverse medical disciplines.

5.3.1 Datasets

In order to demonstrate wide applicability, the AUCMEDI framework was applied to datasets from the following medical disciplines and procedures: Dermatology via dermatoscopy, gastroenterology via endoscopy, histopathology via microscopy, neurology via MRI, radiology via X-ray as well as CT, gynecology via ultrasound, and ophthalmology via retinal imaging. For this experiment, only 2D imaging was considered for classification. A brief overview of the analyzed datasets can be found in Table 5.6.

Table 5.6: Overview of the datasets utilized in the broad application experiments.

Medical Discipline	Dermatology	Gastroenterology	Histopathology	Neurology	Radiology	Radiology	Gynecology	Ophthalmology
Medical Procedure	Dermatoscopy	Endoscopy	Microscopy	MRI	X-ray	CT	Ultrasound	Retinal Imaging
Task	Skin Lesion Classification (Melanoma)	Landmark Classification (upper & lower)	Invasive Ductal Carcinoma Detection	Brain Tumor Classification	Pneumonia Detection	COVID-19 Classification	Breast Cancer Screening	Retinal Multi-Disease Detection
Reference	[57–59]	[367]	[368, 369]	[370]	[371]	[62]	[61]	[158]
Samples	25,331	2,695 (upper) 1,409 (lower)	277,524	7,022	5,856	19,685	1,578	3,588
Classes	8	3 (upper) 3 (lower)	2	4	2	3	3	28
Multi-label	No	No	No	No	No	No	No	Yes

Dataset Descriptions

The dermatology dataset providing dermatoscopy images was published by the International Skin Imaging Collaboration (ISIC) [57–59] and consists of the following classes: Melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma, vascular lesion, squamous cell carcinoma, and unknown. 8 of these are present in the training data used for this evaluation. The unknown class does not occur and was explicitly excluded during preparations. The gastroenterology dataset providing endoscopy images was published by Borgli et al. [367]. Two subsets from the dataset were used for separate anatomical landmark classification tasks, one for the upper gastrointestinal (GI) tract and one for the lower GI tract. The upper GI tract dataset consists of the following classes: Pylorus, retroflex-stomach, and z-line. Accordingly, the lower GI tract dataset consists of the following classes: Cecum, ileum, and retroflex-rectum. The histopathology dataset providing microscopy images was published by Madabhushi et al. [368, 369] and consists of the following classes based on invasive ductal carcinoma presence: Negative and positive. Images were originally mapped to certain patients. This information was discarded and not respected for splitting the dataset. The neurology dataset providing 2D MRI slides was published by Msoud Nickparvar [370] and consists of the following classes: Glioma, meningioma, pituitary, and no tumor presence. The radiology dataset providing 2D CT slides was published by Ning et al. [371] and consists of the following classes: Non-informative finding in which lung parenchyma was not captured for any judgment, positive finding in which features associated with COVID-19 pneumonia could be unambiguously discerned, and negative finding in which features in both lungs were irrelevant to COVID-19 pneumonia. The dataset consists of original and preprocessed CT scans. Only original images were considered. The radiology dataset providing X-ray images was published by Kermany et al. [62] and consists of the following classes depending on pneumonia presence: Negative and Positive. The gynecology dataset providing ultrasound images was published by Al-Dhabyani et al. [61] and consists of the following classes: Normal, benign, and malignant. The ophthalmology dataset providing retinal images was published by Pachade et al. [158] and consists of 28 multi-label classes for various medical conditions. The dataset was also utilized and described in detail in Chapter 5.4.

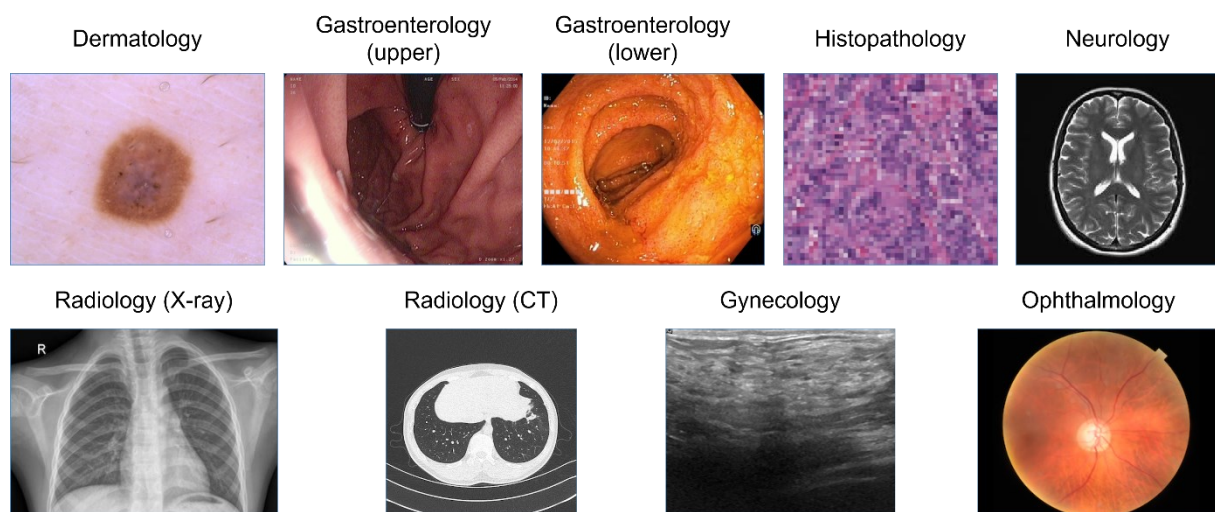


Figure 5.17: Exemplary samples of the analyzed datasets in this study.

5.3.2 Application

AUCMEDI applies several preprocessing techniques by default: Images are augmented by flipping and rotating, as well as applying changes to brightness, contrast, saturation, hue, and scale. Afterward, images are resized to 224x224 pixels and standardized according to Z-score.

Datasets are split into three sets: 70% of the data are used for training, 10% are used as validation during training, and 20% are used as hold-out set for testing. The split is stratified to keep the proportional class distribution similar in all sets.

The neural network model is the core of the AUCMEDI pipeline. It consists of one model based on the DenseNet121 architecture [90]. To keep the required training time at a minimum as well as improve overall generalizability, a transfer learning approach was applied: AUCMEDI internally uses the defined architecture's pretrained weights and fine-tunes these weights using the provided training data. The pretrained weights were based on ImageNet [76]. Training is done for a maximum of 500 epochs. To further reduce the required training time, an early stopping strategy was employed. Additionally, the learning rate was dynamically adjusted to improve the learning progress.

Based on the hold-out set for which predictions were generated with the fitted model, the classification performance was evaluated. For performance assessment, the following metrics were computed: Accuracy, F1-score, the ROC curve, and area under the ROC curve (AUC).

The ability to understand a classification decision by a model, commonly known as explainability, is fundamental in the medical context. One possibility to explain a machine-made decision is by highlighting relevant regions within the classified image which can be achieved by a heatmap visualization. For multi-class datasets, one sample per class was evaluated utilizing the Grad-CAM algorithm [344], whereas, in the case of a multi-label task, Grad-CAM images for only one sample from the hold-out set for all possible classes were generated.

To ensure full reproducibility, the complete code of this study is available in the AUCMEDI Git repository and accessible in the example registry in the AUCMEDI documentation: <https://frankkramer-lab.github.io/aucmedi/examples/framework/>.

5.3.3 Results and Discussion

Training for each of the 9 experiments was performed on a single NVIDIA TITAN RTX GPU. Due to the early stopping strategy and deployed simplistic architecture in terms of the number of parameters, training was finished after on average 51.6 epochs.

The classification performance for all experiments revealed strong predictive capabilities. Achieved AUC, Accuracy, and F1-scores are summarized in Figure 5.18 in which the bar plots represent the average performance by mean and the error bars represent the standard deviation. Since the ophthalmology dataset is a multi-label task, F1-scores hold limited informative value

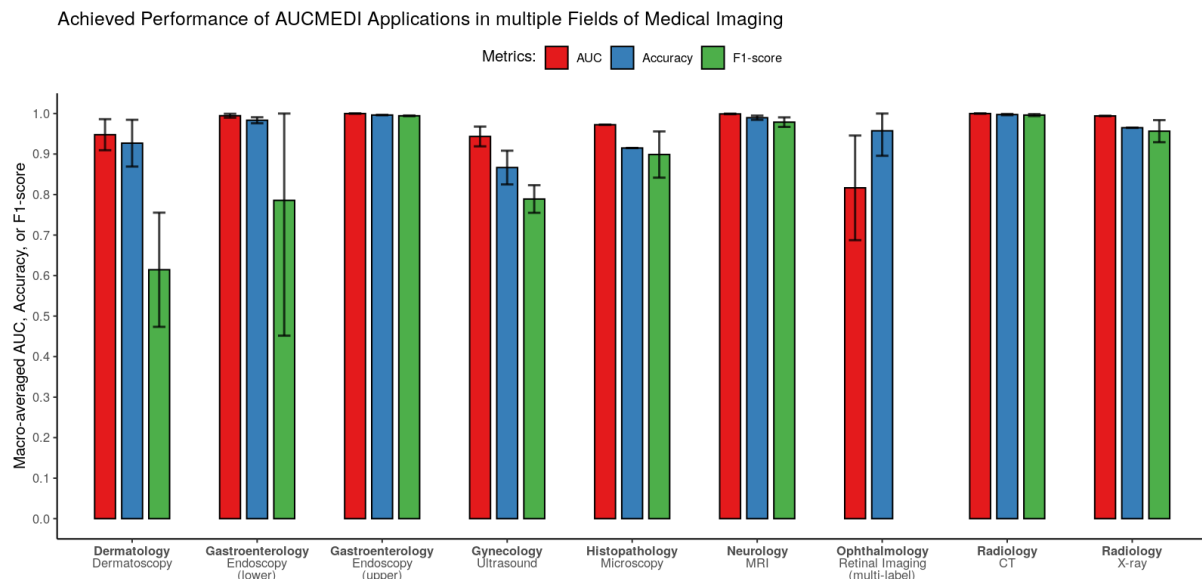


Figure 5.18: Summary of the achieved performance in the conducted experiments by mean and standard deviation.

and were omitted. In the following sections, noteworthy observations for each dataset are elaborated and resulting insights are discussed.

Individual Dataset Results

The dermatology dataset containing images of skin lesions for melanoma classification revealed a strong class imbalance. Nevertheless, the applied model was able to achieve accurate classification performance with an F1-score of approximately 0.6 and an AUC of more than 0.9. The two subsets from the gastroenterology dataset were evaluated as isolated landmark classification tasks. The computed results showed a discrepancy in their performances. By comparing the class distributions between the two datasets, the massive class imbalance for the lower GI tract can be noticed. The ileum class consists of 9 images, as opposed to the cecum class with more than 1,000 samples. Further detailed analysis of the classification results confirmed that inferior predictions are associated with the underrepresented ileum class. The histopathology dataset is a binary classification task to identify invasive ductal carcinoma, the most common subtype of breast cancer [368]. Despite the provided small image sizes of only 50x50 pixels, the model was able to reliably distinguish healthy from pathological slides. Since the patient information was omitted during preprocessing, it would be promising to include the available metadata through the metadata interface of AUCMEDI in this only image-based approach. The neurology dataset represents a large as well as class-balanced set of images. This is why the model was able to achieve an exceptional classification performance of an F1-score of 0.9 and an AUC of 1.0. In the CT-based radiology dataset, the model achieves remarkable classification performance on the task to detect COVID-19 infected regions. The task of the X-ray-based radiology dataset was to distinguish if a patient has pneumonia or not. The calculated F1-score of 0.95 and AUC of 1.0 revealed that the fitted model was able to achieve powerful classification performance, as well. However, a slight class imbalance was also observed in the gynecology dataset, where breast cancer was to be identified using ultrasound images. The

classification performance on this dataset was adequate but favored classifying samples as normal, a class that contains merely 266 images, and was cautious in assigning samples the class benign, which was with 891 samples the most common class in the gynecology dataset. A more balanced dataset or a more complex neural network architecture might improve predictive capabilities. The ophthalmology dataset is the only multi-label task that was evaluated in this study. Due to the complexity of this task, the averaged AUC was approximately 0.8 and notably lower than other datasets, but still strong considering the challenging character of the dataset [158]. Due to the multi-label context, the model did not take one most likely label to assign (softmax activation) but weights each label with a probability for the given sample (sigmoid activation). An experimental analysis of non-optimal predictions also revealed that the fitted model was often capable of correctly identifying other present diseases in the sample demonstrating high sensitivity.

Performance and Adaptability

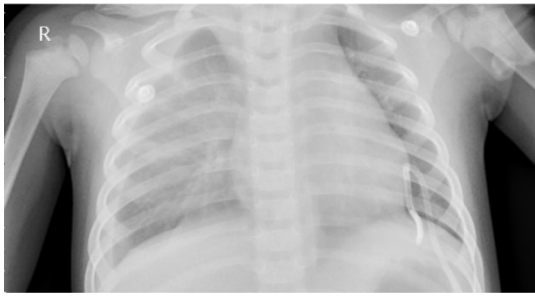
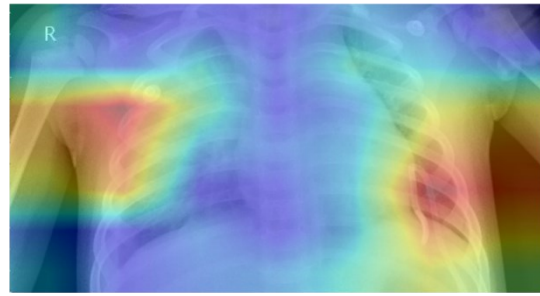
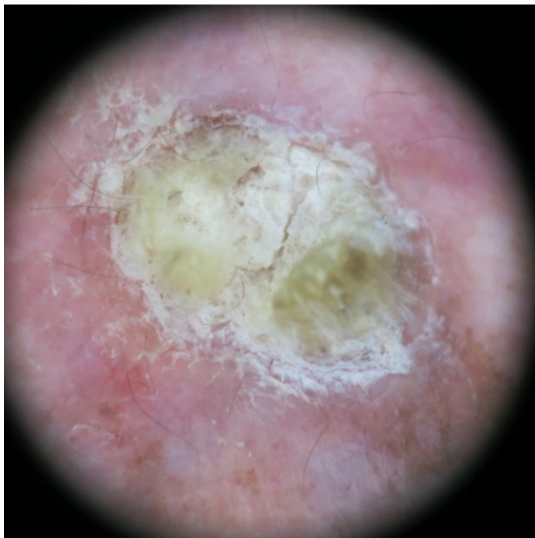
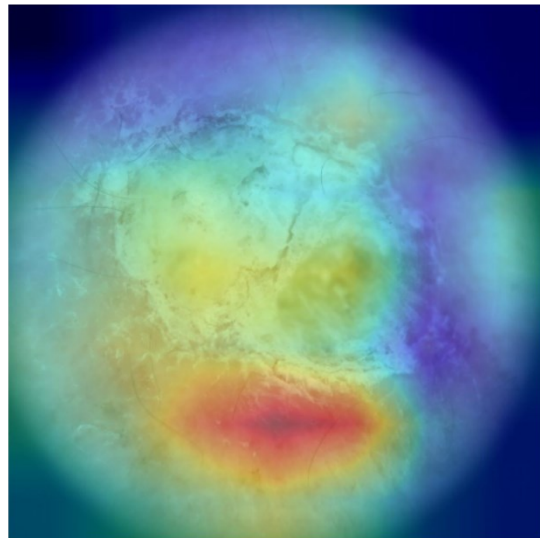
The Author concluded that with an average AUC between 0.8 and 1.0, all fitted models showed a strong performance. That underlines the potential of AUCMEDI: The standardized pipeline was successfully applied to a diverse range of modalities which is highly promising for clinical application. This demonstrates that there is no need for dataset-specific implementations or distinctions between binary, multi-class, and multi-label problems. Instead, clinical researchers are able to design a state-of-the-art pipeline without extensive domain knowledge, apply it to the classification task, and achieve accurate predictions. Thus, the high-level programming interface of AUCMEDI is simple to use even with limited deep learning knowledge or programming experience.

Class Imbalance Influence on Performance

Detailed performance metrics for each dataset revealed, that a well-balanced, large dataset is more likely to generate accurate classifications. Evaluating the two gastroenterology datasets, both subsets had the task to assign anatomical landmarks to images and both contained three distinct classes that were to be identified by the machine. However, the subset relating to the lower GI tract was massively unbalanced. The ileum class contains less than 10 samples, whereas the cecum class contains more than 1,000 samples. In contrast, the upper GI tract subset was more balanced. The stratified sampling split used for the pipeline reflected these proportions. The computed AUC, Accuracy, and F1-scores for both subsets indicated that the landmark classification could work reliably but requires a balanced dataset. A similar phenomenon was observed for the ophthalmology dataset. Its class distribution showed several classes with less than 10 occurrences which had also the lowest performance scores.

Importance, Potential, and Challenges of XAI

Due to the sparsity and diversity of medical data, the capability to understand a machine-made decision is crucial in the medical domain. Particularly promising are insights into images from a modality that displays medical instrumentation, for instance X-rays. The processed X-ray

A : Radiology (X-ray) - Original**B : Radiology (X-ray) - Grad-CAM****C : Dermatology - Original****D : Dermatology - Grad-CAM****Figure 5.19:** XAI visualization of samples from the radiology (X-ray) and dermatology dataset.

dataset in this study, for which a sample is visualized in Figure 5.19 (parts A and B), shows electrodes and what appears to be a drainage. The model identified this sample as having pneumonia based on these instruments and not based on lung tissue. This example illustrates the importance of explainability in the medical context. Furthermore, it is important to note that XAI is not limited to explaining a decision of a model, but can hint humans to image sections, they might not even consider to be relevant for the condition. The dermoscopic images of suspicious-looking skin patches are an adequate example which is illustrated in Figure 5.19 (parts C and D). The figure shows a squamous cell carcinoma and its respective Grad-CAM visualization. The XAI image marks a small section of skin just below the abnormal tissue region which apparently revealed information about the condition. However, the gold-standard to diagnose this type of cancer is an invasive biopsy and the subsequent histopathological examination [372]. With the support of XAI technologies, it may be possible for medical experts to develop more non-invasive diagnostic measures. Nevertheless, XAI still faces various challenges: The smallest cluster of pixels can be enough for a neural network model to make a decision with high confidence. But if the image has a low resolution and details are humanly not possible to identify, XAI would hardly yield any benefits.

5.3.4 Conclusions

In this study, the capabilities of the proposed framework AUCMEDI were evaluated. The pipeline for medical image classification was developed and applied to 8 datasets from a wide range of medical imaging modalities. Each of the trained models achieved an exceptional classification performance. This evaluation illustrates that AUCMEDI is a powerful tool to build modular and reusable classification pipelines without restrictions to specific datasets or medical disciplines. However, the application of a single pipeline to several datasets also emphasizes the relevance of data. Detailed analysis revealed that the dips in classification performance throughout the experiments were caused by a significant class imbalance within the respective dataset. That a standardized pipeline applied to diverse classification tasks was able to generate independently well-performing models, is promising for future clinical applications.

5.4 Study: Multi-Disease Detection in Retinal Imaging

Even if the medical progress in the last 30 years made it possible to successfully treat the majority of diseases causing visual impairment, growing and aging populations lead to an increasing challenge in retinal disease diagnosis [373]. The World Health Organization (WHO) estimates the prevalence of blindness and visual impairment to be 2.2 billion people worldwide, of whom at least 1 billion affections could have been prevented or is yet to be addressed [374]. Early detection and correct diagnosis are essential to forestall disease course and prevent blindness.

The use of clinical decision support systems for diagnosis has been increasing over the past decade [24]. Recently, modern deep learning models allow automated and reliable classification of medical images with remarkable accuracy comparable to physicians [13]. Nevertheless, these models often lack capabilities to detect rare pathologies such as central retinal artery occlusion or anterior ischemic optic neuropathy [375, 376].

In this study, the Author pushed toward creating a highly accurate and reliable multi-disease detection pipeline based on ensemble, transfer, and deep learning techniques. Furthermore, the recently published Retinal Fundus Multi-Disease Image Dataset (RFMiD) was utilized containing various rare and challenging conditions to demonstrate detection capabilities for rare diseases [158].

5.4.1 Dataset

The RFMiD dataset consists of 3,200 retinal images for which 1,920 images were used as training dataset [158]. The fundus images were captured by three different fundus cameras having a resolution of 4288x2848 pixels (277 images), 2048x1536 pixels (150 images), and 2144x1424 pixels (1,493 images), respectively.

Table 5.7: Annotation frequency for each class in the RFMiD dataset.

Disease	Samples	Disease	Samples	Disease	Samples
Disease Risk	1,519	DR	376	ARMD	100
MH	317	DN	138	MYA	101
BRVO	73	TSLN	186	ERM	14
LS	47	MS	15	CSR	37
ODC	282	CRVO	28	TV	6
AH	16	ODP	65	ST	5
AION	17	PT	11	RT	14
RS	43	CRS	32	EDN	15
RPEC	22	MHL	11	RP	6
OTHER	34				

The images were annotated with 46 conditions, including various rare and challenging diseases, through adjudicated consensus of two senior retinal experts. These 46 conditions are represented by the following classes, which are also listed in Table 5.7: An overall normal/abnormal class ('Disease Risk'), 27 specific condition classes, and one 'OTHER' class

consisting of the remaining extremely rare conditions. Besides the training dataset, the dataset authors hold 1,280 images back for organizing the RIADD challenge in which the remaining data is used for external validation and testing datasets to ensure robust evaluation [158, 173].

The following list contains the full disease names of all class acronyms in the RFMiD dataset. All classes which are not represented in Table 5.7 had less than 10 samples in the public as well as hidden/hold-out dataset and were merged as ‘OTHER’. More information and details on the dataset can be obtained from Pachade et al. [158, 173].

List of Class Acronyms in RFMiD

Diabetic retinopathy (DR), age-related macular degeneration (ARMD), media haze (MZ), drusen (DN), myopia (MYA), branch retinal vein occlusion (BRVO), tessellation (TSLN), epiretinal membrane (ERM), laser scar (LS), macular scar (MS), central serous retinopathy (CSR), optic disc cupping (ODC), central retinal vein occlusion (CRVO), tortuous vessels (TV), asteroid hyalosis (AH), optic disc pallor (ODP), optic disc edema (ODE), shunt (ST), anterior ischemic optic neuropathy (AION), parafoveal telangiectasia (PT), retinal traction (RT), retinitis (RS), chorioretinitis (CRS), exudation (EDN), retinal pigment epithelium changes (RPEC), macular hole (MHL), retinitis pigmentosa (RP), cotton wool spots (CWS), coloboma (CB), optic disc pit maculopathy (ODPM), preretinal hemorrhage (PRH), myelinated nerve fibers (MNF), hemorrhagic retinopathy (HR), central retinal artery occlusion (CRAO), tilted disc (TD), cystoid macular edema (CME), post traumatic choroidal rupture (PTCR), choroidal folds (CF), vitreous hemorrhage (VH), macroaneurysm (MCA), vasculitis (VS), branch retinal artery occlusion (BRAO), plaque (PLQ), hemorrhagic pigment epithelial detachment (HPED) and collateral (CL).

5.4.2 Application

In order to build a complete and efficient MIC pipeline, the framework AUCMEDI was utilized. In the following three subchapters, the AUCMEDI and pipeline configuration are described. The structure of the implemented MIC pipeline is illustrated in Figure 5.20. The workflow is starting with the retinal imaging dataset (RFMiD) and ends with computed predictions for novel images.

Preprocessing and Image Augmentation

In order to simplify the pattern-finding process of the deep learning model, as well as to increase data variability, several preprocessing methods were applied.

Extensive image augmentation was utilized by up-sampling to balance class distribution and by on-the-fly (real-time) augmentation during training to obtain novel as well as unique images in each epoch. The augmentation techniques consisted of rotation, flipping, and altering brightness, saturation, contrast, and hue. Through the up-sampling, it was ensured that each label occurred at least 100 times in the dataset which increased the total number of training images from 1,920 to 3,354.

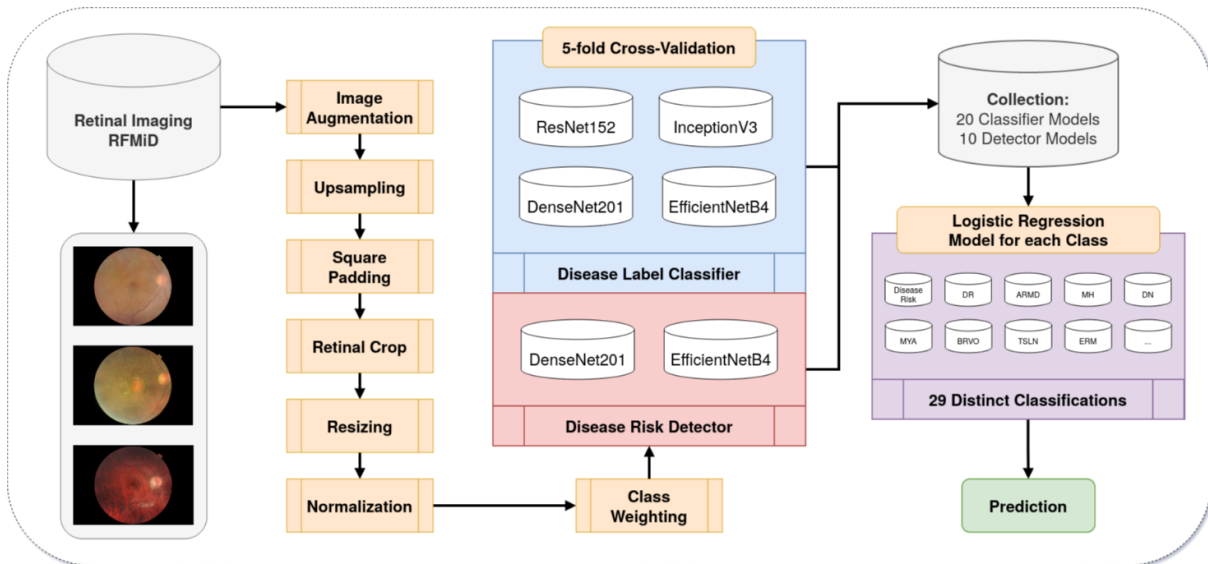


Figure 5.20: Flowchart diagram of the implemented MIC pipeline for multi-disease detection in retinal imaging.

Afterward, all images were square-padded in order to avoid aspect ratio loss during posterior resizing. The retinal images were also cropped to ensure that the fundus is center-located in the image. The cropping was performed individually for each microscope resolution and resulted in the following image shapes: 1424x1424, 1536x1536, and 3464x3464 pixels. The images were then resized to model input sizes according to the neural network architecture, which was 380x380 pixels for EfficientNetB4, 299x299 pixels for InceptionV3, and 244x244 pixels for all remaining architectures [79, 80, 87, 90].

Before feeding the image to the deep convolutional neural network, value intensity normalization was applied as last preprocessing step. The intensities were zero-centered via the Z-Score normalization approach based on the mean and standard deviation computed on the ImageNet dataset [76].

Neural Network Models

The state-of-the-art for medical image classification is the unmatched deep convolutional neural network [13, 36]. Nevertheless, the hyperparameter configuration and architecture selection are highly dependent on the required computer vision task [13, 34]. Thus, the proposed pipeline combines two different types of image classification models: The disease risk detector for binary classifying normal/abnormal images and the disease label classifier for multi-label annotation of abnormal images.

Both model types were pretrained on the ImageNet dataset [76]. For the fitting process, transfer learning training, with frozen architecture layers except for the classification head, and a fine-tuning strategy with unfrozen layers were applied. Whereas the transfer learning fitting was performed for 10 epochs using the Adam optimization [201] with an initial learning rate of $1E^{-4}$, the fine-tuning had a maximal training time of 290 epochs and using a dynamic learning rate for the Adam optimization starting from $1E^{-5}$ to a maximum decrease to $1E^{-7}$ (decreasing factor

of 0.1 after 8 epochs without improvement on the monitored validation loss) [201]. Furthermore, an early stopping and model checkpoint technique was utilized for the fine-tuning process, stopping after 20 epochs without improvement (only active after epoch 60 was reached) and saving the best model measured according to the validation loss. Instead of defining an epoch as a cycle through the full training dataset, it was established that an epoch has 250 iterations. The images for a batch were randomly drawn, considering that as many samples as possible are used based on the number of iterations. This allowed to increase the number of seen batches and, thus, to increase the information given to the model during the fitting process of an epoch. As training loss function, the weighted Focal loss from Lin et al. [213] was utilized:

$$Focal_{weighted}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \quad (5.1)$$

In the above formula, p_t is the probability for the correct ground truth class t , γ a tunable focusing parameter (which was set to 2.0), and α_t the associated weight for class t [213].

The disease risk detector was established as a binary classifier of the disease risk class for general categorizing between normal and abnormal retinal images. Thus, this model type was trained using only the ‘Disease Risk’ class and ignoring all multi-label annotations. Rather than using a single model architecture, multiple models were trained based on the DenseNet201 and EfficientNetB4 architecture [79, 90]. For class weight computation, the number of samples was divided by the multiplication of the number of classes (2 for binary classification) with the number of class occurrences in the dataset.

In contrast, the disease label classifier was established as multi-label classifier of all 28 remaining classes (excluding the disease risk) and was trained on the one hot encoded array of the disease labels. Furthermore, four different architectures were utilized for this model type: ResNet152, InceptionV3, DenseNet201, and EfficientNetB4 [79, 80, 87, 90]. Identical to the class weight computation of the disease risk detector, the weights were computed individually as binary classification for each class. Even if this classifier is provided with all classes, the binary weights balance the decision for each label individually.

Ensemble Learning Strategy

The concept of ensemble learning is to combine predictions from multiple models which increases the overall performance of the pipeline. An in detail definition of, discussion, and studies about ensemble learning for MIA can be found in Chapter 6.

Next to the utilization of multiple architectures, a 5-fold cross-validation was also applied as a Bagging approach for ensemble learning. The aim was to create a large variety of models which were trained on different subsets of the training data. This approach not only allowed a more efficient usage of the available training data but also increased the reliability of a prediction.

This strategy resulted in an ensemble of 10 disease risk detector models (2 architectures with each 5 folds) and 20 disease label classifier models (4 architectures with each 5 folds).

For combining the predictions of the implemented, in total, 30 models, a Stacking setup was integrated. On top of all deep convolutional neural networks, a binary logistic regression algorithm was applied for each class, individually. Thus, the predictions of all models were utilized as input for computing the classification of a single class. This approach allowed combining the information of all other class predictions to derive an inference for one single class. Overall, this strategy resulted in 29 distinct logistic regression models (1 for the disease risk and 28 for each disease label including the ‘OTHER’ class). The individual predicted class probabilities are then concatenated to the final prediction.

The logistic regression models were also trained with the same 5-fold cross-validation sampling on a heavily augmented version of the training dataset to avoid overfitting as well as to avoid training the logistic regression models on already seen images from the neural network models. As logistic regression solver, the large-scale bound-constrained optimization (short: ‘LBFGS’) from Zhu et al. [377] was utilized.

Code Reproducibility

To ensure full reproducibility, the complete code of this study, including extensive documentation, is available in the following public Git repository: <https://github.com/frankkramer-lab/riadd.aucmedi>.

Furthermore, the trained models, evaluation results, and metadata are available in the following public Zenodo repository: <https://doi.org/10.5281/zenodo.4573990>.

5.4.3 Results and Discussion

The sequential training took 13.5 hours with 63 epochs on average for each deep convolutional neural network model. Logistic Regression training required less than 30 minutes for all class models combined.

The training process is illustrated via fitting curves in Figure 5.21. The lines in the figures were computed via locally estimated scatterplot smoothing and represent the average loss across all folds, whereas the gray areas around the lines represent the confidence intervals. No signs of overfitting were observed for the disease label classifiers through validation monitoring. However, the disease risk detectors showed a strong trend to overfit. A reason for this is that the binary classification results in a too low inductive bias of the model. Due to the transfer learning, a high correlation of ImageNet features with strong visual disease features is plausible resulting in neglecting minor disease features. Especially, the DenseNet architecture reveals a high risk of re-using these starting features resulting in distinct overfitting. However, through the applied strategy to use the earlier models based on validation loss monitoring, it was still possible to obtain powerful models for detection.

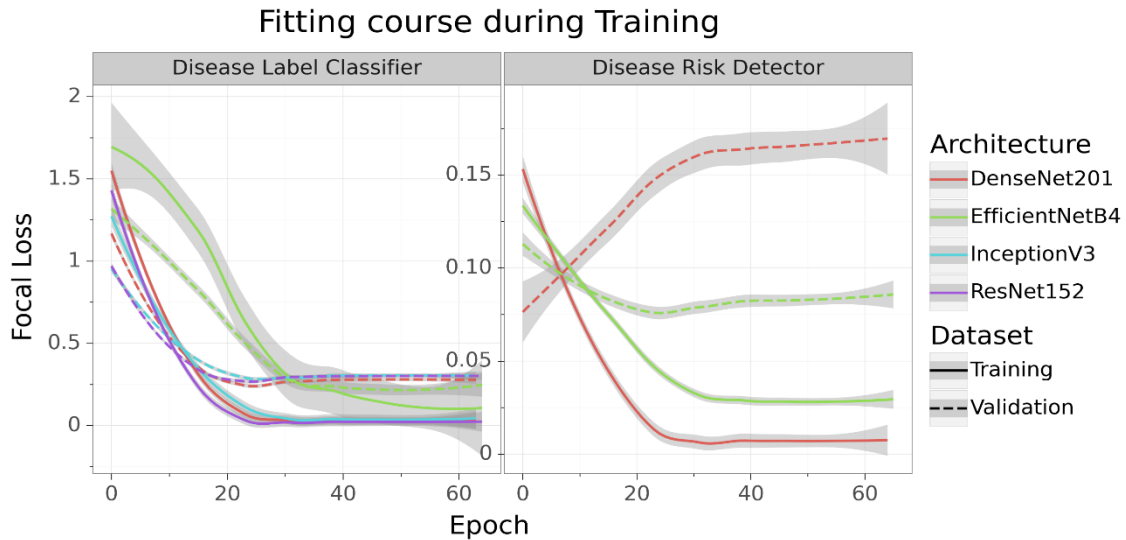


Figure 5.21: Fitting curves showing the training and validation loss during the training process.

Internal Performance Evaluation

For estimating the performance of the proposed pipeline, the validation subsets of the 5-fold cross-validation models were utilized from the heavily augmented version of the described dataset. This approach allowed to obtain testing samples that were never seen in the training process for reliable performance evaluation. For the complex multi-label evaluation, the popular area under the receiver operating characteristic (AUROC) curve, as well as the mean average precision (mAP) were computed. Both scores were macro-averaged over classes and cross-validation folds to reduce complexity. The associated receiver operating characteristics of the models are illustrated in Figure 5.22 and more details on the internal performance evaluation are listed in Table 5.8.

Table 5.8: Achieved results of the internal performance evaluation.

Model Type	Architecture	AUROC	mAP
Classifier	DenseNet201	0.973	0.931
Classifier	EfficientNetB4	0.969	0.929
Classifier	ResNet151	0.970	0.930
Classifier	InceptionV3	0.932	0.663
Detector	DenseNet201	0.980	0.997
Detector	EfficientNetB4	0.993	0.999
Ensembler	Logistic Regression	0.999	0.999

The proposed multi-disease detection pipeline revealed a strong and robust classification performance with the capability to also detect rare conditions accurately in retinal images. Whereas the disease label classifier models separately only achieved an AUROC of around 0.97 and a mAP of 0.93, the disease risk detectors demonstrated to have a really strong predictive power of 0.98 up to 0.99 AUROC and mAP. However, for the classifiers, the InceptionV3

architecture indicated to have the worst performance compared to the other architectures with only 0.93 AUROC and 0.66 mAP.

Training a strong multi-label classifier is in general a complex task, however, the extreme class imbalance between the conditions revealed a hard challenge for building a reliable model [378, 379]. The applied up-sampling and class weighting technique demonstrated to have a critical boost on the predictive capabilities of the classifier models. The Author base this critical boost on a synergy effect between the weighted focal loss, by handicapping samples with very high model confidence or high class frequency, and the up-sampling augmentation, by increasing the probability of a minority class to be present in a randomly drawn batch from the dataset.

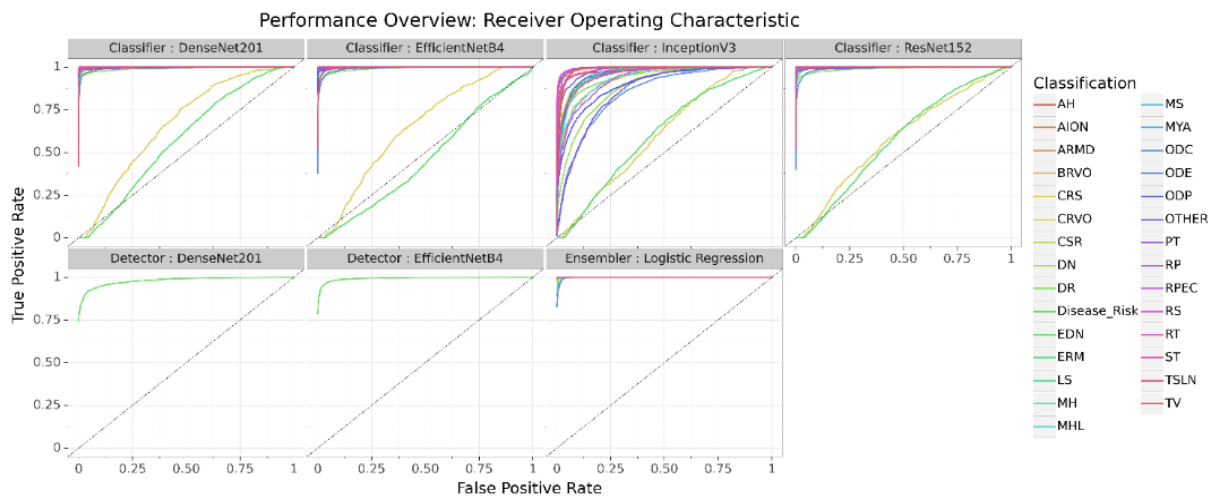


Figure 5.22: Receiver operating characteristic curves for each model type applied in the proposed pipeline.

Nearly all labels were able to be accurately detected, including the ‘OTHER’ class consisting of various extremely rare conditions. Nevertheless, the two classes ‘EDN’ and ‘CRS’ were the most challenging conditions for all classifier models. Both classes belong to very rare conditions, combined with 47 occurrences (1.2%) in the original and 209 occurrences (2.5%) in the up-sampled dataset. Still, the stacked logistic regression algorithm was able to balance this issue and infer the correct ‘EDN’ and ‘CRS’ classifications through context. Overall, the applied ensemble learning strategies resulted in a significant performance improvement compared to the individual deep convolutional neural network models.

External Evaluation through the RIADD Challenge

Furthermore, the Author participated in the RIADD challenge which was organized by the authors of the RFMiD dataset [158, 173]. The challenge participation allowed not only an independent evaluation of the predictive power of the proposed pipeline on an unseen and unpublished testing set but also the comparison with the currently best retinal disease classifiers in the world.

In the participation, the proposed pipeline reached rank 19 from a total of 59 teams in the first evaluation phase and rank 7 in the final phase. In the independent evaluation by the challenge

organizers, an AUROC of 0.95 for the disease risk classification was achieved. For multi-label scoring, the challenge organizers computed the average between the macro-averaged AUROC and the mAP, for which a score of 0.70 was reached by the proposed pipeline. The top performing ranks shared only a marginal scoring difference which is why the proposed pipeline had only a final score difference of 0.05 to the first-ranked team.

Table 5.9: Ranking of the top-7 participants for the final evaluation phase of the RIADD challenge.

Rank	Team	Final Score
1.	KAMATALAB	0.8848
2.	Schulich Applied Computing inMed	0.8821
3.	BNUAA	0.8724
4.	Nekaravuru	0.8634
5.	WWW	0.8572
6.	IGSTfencing	0.8449
7.	MISIT (proposed pipeline)	0.8300

In a retrospectively hosted workshop by the challenge organizers at the IEEE ISBI 2021 International Symposium on Biomedical Imaging conference, it was possible to compare the proposed pipeline with the strategies and implementations of the higher ranking teams. Zhengbo et al. [380] (from the team KAMATALAB) used multiple EfficientNet architectures with an input size of 960x960 pixels, Ho et al. [381] (from the team Schulich Applied Computing inMed) used an ensemble of the SE-ResNeXt [382], DenseNet121, InceptionV3, as well as EfficientNet architecture with an input size of up to 512x512 pixels, and Wang et al. [383] (from the team BNUAA) used Bagging based ResNeXt [88] models with an input size of 512x512 pixels. Whereas all top-performing teams implemented an ensemble learning strategy, the major difference resulting in higher performance was the utilization of larger input sizes (resolutions) for the neural network models.

5.4.4 Conclusions

In this study, a powerful multi-disease detection pipeline for retinal imaging was implemented which exploits ensemble learning techniques to combine the predictions of various deep convolutional neural network models. Next to state-of-the-art strategies, such as transfer learning, class weighting, extensive real-time image augmentation, and Focal loss utilization, a 5-fold cross-validation as Bagging technique was applied and used multiple convolutional neural network architectures to create an ensemble of models. With a Stacking approach of class-wise distinct logistic regression models, the knowledge of all neural network models was combined to compute highly accurate and reliable retinal condition predictions. Next to an internal performance evaluation, the precision and comparability of the proposed pipeline were also proved through participation in the RIADD challenge. As future work, the Author is interested in further improving the proposed pipeline by using increased input sizes and validating the medical gain for automated multi-disease detection in retinal imaging as clinical decision support through a clinical study.

5.5 Study: COVID-19 Infection and Severity Prediction

In reaction to the rapid spread of the coronavirus at the beginning of the year 2020, many scientists quickly reacted and developed various approaches based on deep learning to contribute to the efforts against COVID-19. The scientific community focused its efforts on the development of models for COVID-19 classification, because X-ray and CT images of infected patients could be collected without further annotations [296, 298]. These classification algorithms can be categorized through their objectives: 1) Classification of COVID-19 from non-COVID-19 (healthy) patients, which resulted in models achieving a Sensitivity of 94.1%, Specificity of 95.5%, and AUC of 0.979 by Jin et al. [384]. 2) Classification of COVID-19 from other pneumonia, which resulted in models achieving a Sensitivity of 100.0%, Specificity of 85.18%, and AUC of 0.97 by Abbas et al. [385]. 3) Severity assessment of COVID-19, which resulted in a model achieving a true positive rate of 91.0%, true negative rate of 85.8%, and Accuracy of 89.0% by Tang et al. [386].

Especially the severity assessment of patients is essential for treatment decisions and disease course monitoring. However, the course of the coronavirus pandemic showed that one of the most critical factors for COVID-19 treatment is the capacity of intensive care units (ICU) at hospitals [297, 387, 388]. Through the rapid but inconsistent development of infection severity, predicting the future severity of a patient within a month (prognosis) for capacity planning of ICUs is challenging but crucial [387, 388]. Furthermore, even though a variety of deep learning based MIC approaches to support clinical decision finding for COVID-19 was developed and demonstrated significant performance in research, the transition of these AI models into clinical environments presented significant difficulties [32, 39]. Reproducibility as well as reusability of developed AI models are often only limited for clinical research and unfeasible for routine usage [32, 39].

In this study, a joint effort of multiple labs from the University of Augsburg (including the Author) pushed toward creating an accurate MIC pipeline for predicting COVID-19 presence and severity prognoses utilizing ensemble, transfer, as well as deep learning techniques. To ensure reusability in clinical environments, the proposed pipeline was implemented in a Docker environment. For performance validation, the predictive capabilities of the pipeline were internally as well as externally evaluated by participation in the STOIC challenge [137, 389].

5.5.1 Dataset

For model training and internal performance evaluation, the dataset from the STOIC (Study of Thoracic CT in COVID-19) project was used [137, 389]. The publicly available dataset consists of 2,000 patients with thorax CT scans which is a subcohort of the 10,735 patients in total that were gathered in the project. The patients were recruited from 20 university hospitals (primarily from Paris) and were categorized according to age as well as sex which was also provided as metadata in the dataset [137]. The age categories were grouped by decades (<40 years, 40-50 years, 50-60 years, 70-80 years, and >80 years), whereas the sex was defined as boolean feature (male and female). A summary of the dataset characteristics can be seen in Table 5.10.

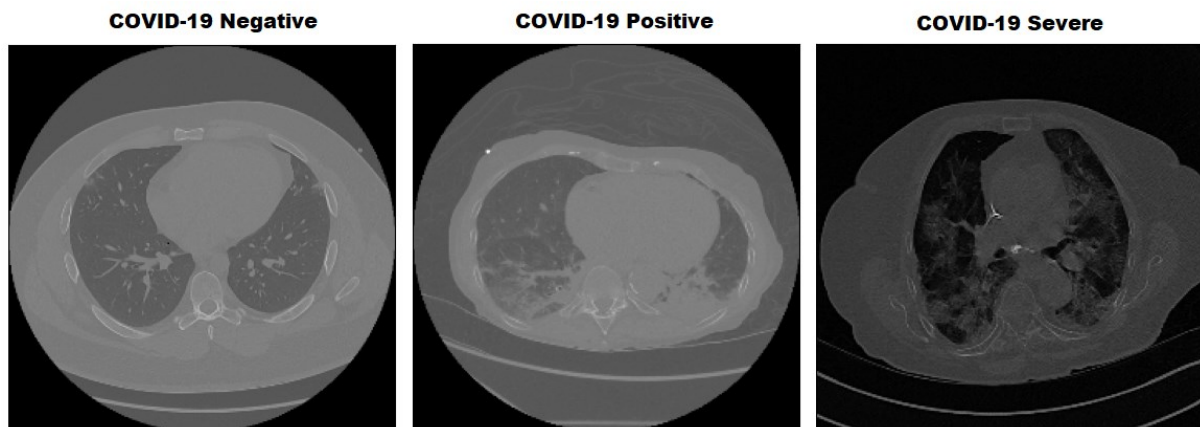
Table 5.10: Summary of the STOIC dataset characteristics and proposed sampling.

Dataset Characteristics			
	Female	Male	Total
Samples	852	1148	2000
COVID-19 Negative	369	426	795
COVID-19 Positive	391	513	904
COVID-19 Severe	92	209	301
Age \geq 65	490	689	1179
Age $<$ 65	362	459	821
Proposed Sampling for Internal Performance Evaluation			
	COVID-19 Negative	COVID-19 Positive	COVID-19 Severe
Total	795	904	301
Training (70%)	556	633	211
Validation (10%)	80	90	30
Testing (20%)	159	181	60

The samples were annotated with two features [137, 389]: 1) The infection with SARS-CoV-2 is also called ‘COVID-19 Positive’ and was assessed by PT-PCR testing (binary feature). 2) The 1-month follow-up outcome of a patient with COVID-19. The outcome, which was also called ‘COVID-19 Severe’, was defined as severe or non-severe (binary feature). A patient was categorized as severe by death or the need for intubation within one month.

The thorax CT scans had a resolution of 512x512 pixels with a number of slices of 434 by mean and 432 by median. The lowest number of slices in a volume was 124, whereas the highest number was 1199. The dataset consists of three groups: Patients without COVID-19 who were classified as ‘COVID-19 Negative’, patients with an active SARS-CoV-2 infection but without a severe outcome who were classified as COVID-19 Positive, and patients with an active infection as well as a severe outcome within the 1-month follow-up who were classified as COVID-19 Severe. An example of each class is visualized in Figure 5.23.

The remaining 8,735 patients from the STOIC project were held back as testing set for organizing the STOIC2021 COVID-19 challenge endorsed by MICCAI [389].

**Figure 5.23:** Visualization of CT scans categorized by COVID-19 classification from the STOIC dataset.

5.5.2 Application

In order to setup a state-of-the-art and effective MIC pipeline, the framework AUCMEDI was utilized. The workflow of the implemented MIC pipeline is illustrated in Figure 5.24. The workflow is completely implemented in a reproducible Docker [350] environment. It takes as input thorax CT scans including the patient's age as well as sex as metadata and outputs the predicted probabilities for COVID-19 presence and a severe outcome within a month.

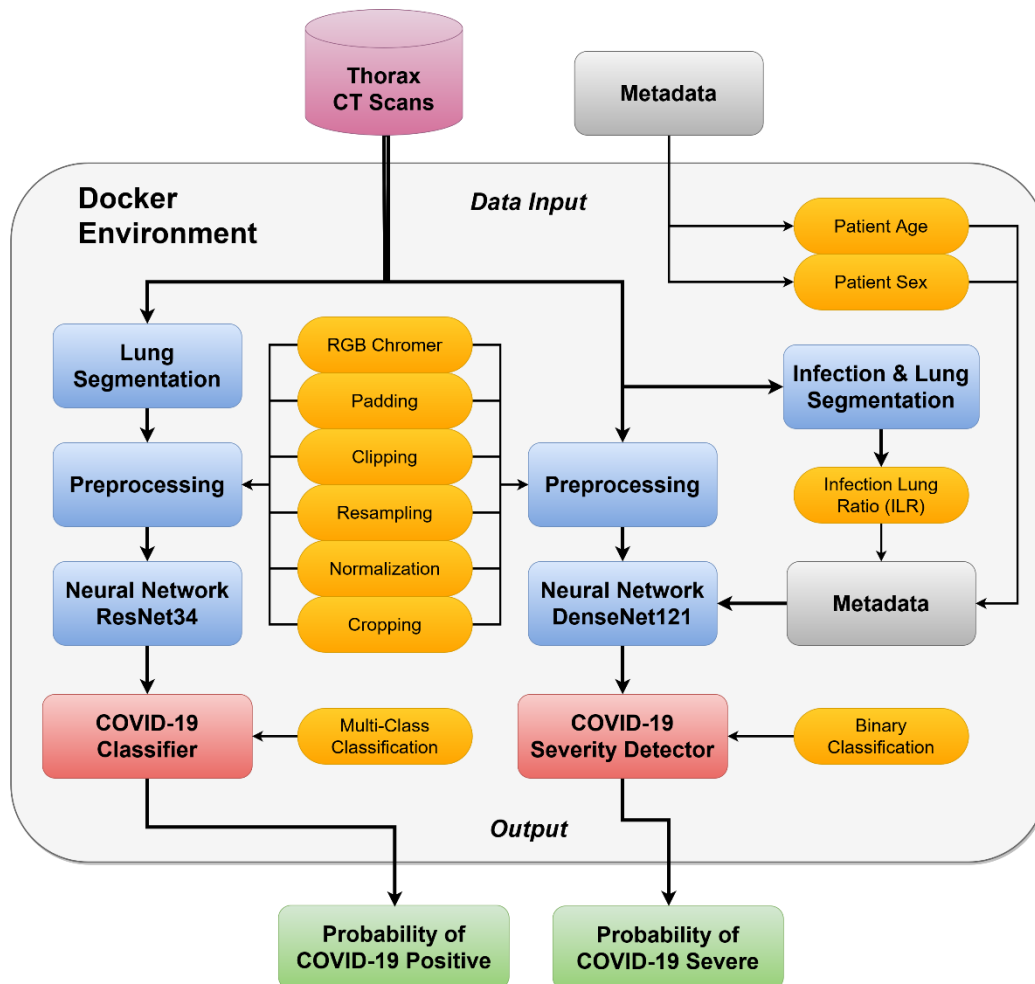


Figure 5.24: Workflow diagram of the implemented pipeline for COVID-19 presence and severity prediction.

To assess the presence and severity of COVID-19 based on computed tomography scans of the lung, an ensemble approach was applied, in which independent models were used for the individual prediction of presence with the ‘COVID-19 Classifier’ and severity with the ‘COVID-19 Severity Detector’. The following subchapter describes the implemented pipeline and AUCMEDI configurations.

Preprocessing and Image Augmentation

In line with current state-of-the-art approaches, several on-the-fly image augmentation methods were applied for the training process. These included rotation, flipping, scaling, gamma modification, and elastic deformations. The following processing methods were used for the

model training and are used for inference. In contrast to the COVID-19 Severity Detector, a segmentation of the lung is applied for the COVID-19 Classifier workflow before preprocessing. Pixels outside of the segmented lung region are excluded and the volume is cropped to a minimal shape around the lung. For the preprocessing of both models, the CT volume is resampled to a voxel spacing of 1.48x1.48x2.10 mm and clipped to the range -1024 HU to $+100$ HU in order to exclude irrelevant tissue types as well as to reduce complexity [145]. Subsequently, the pixel intensities of the volume were standardized to a grayscale range. Samples that might exceed the accepted input image size of $148 \times 224 \times 224$ pixels are center cropped or padded if undersized in order to match the required image size. For training, random cropping was applied instead of center cropping. To enable transfer learning, the grayscale images were converted to RGB. As the last preprocessing step, another normalization is applied via the Z-Score normalization approach based on the mean and standard deviation computed on the ImageNet dataset [76].

Infection Lung Ratio

The severity of lung diseases is commonly quantified in radiology by the percentage of lung area which is affected. This ratio between affected and healthy lung tissue provides intuitive insights into the current state and severity of the disease. For pneumonia as well as COVID-19, a similar approach can be applied. Equivalently to the POI, the infection-lung ratio (ILR) describes the ratio between infected and healthy tissue in the lung. Studies by Wang et al. [390], Xiong et al. [391], and Raoufi et al. [392] proved that the ILR assessed on CT scans has a high correlation to severity measured by labor biomarkers as well as the survival rate. Thus, the ILR was estimated and integrated as metadata in the COVID-19 Severity Detector of the proposed pipeline. For computing the ILR, the pipeline for lung and COVID-19 infection segmentation, which was introduced in Chapter 4.5, was utilized [110].

$$ILR = \frac{|Infection|}{|Infection| + |Lung|} \quad (5.2)$$

The ILR was defined as the number of pixels that were segmented as infection divided by the sum of pixels that were segmented as infection or lung.

Neural Network Models

For the COVID-19 Classifier, a 3D version of the ResNet34 [80] architecture is used, whereas the COVID-19 Severity Detector is based on a 3D version of the DenseNet121 [90] architecture. For the COVID-19 Severity Detector, the classification head was modified to additionally take metadata into account. The metadata consists of three parts: Patient age, sex, and the ILR of each sample.

For the training process of both models, transfer learning was applied to the classification head and a fine-tuning strategy on all layers. The transfer learning was conducted for 10 epochs,

using the Adam optimizer [201] with an initial learning rate of $1E^{-4}$ and a batch size of 4 for the DenseNet as well as 8 for the ResNet. The fine-tuning run for a maximum of 240 epochs, using a dynamic learning rate starting from $1E^{-5}$ to a maximum decrease of $1E^{-7}$ (decreasing factor of 0.1 after 8 epochs without improvement on the monitored validation loss). Furthermore, an early stopping technique was utilized, stopping after 36 epochs without improvement. As a loss function, the sum of the F1-score and the weighted Focal loss from Lin et al. [213] was utilized. The formula of the weighted Focal loss is defined in Chapter 5.4.2, in Equation (5.1).

For inference, the model with the best validation loss of the training process is used. The COVID-19 Classifier predicts the following three classes: COVID-19 Negative, COVID-19 Positive, and COVID-19 Severe. The probability of a COVID-19 infection, which is returned as output, equals the sum of the COVID-19 Positive and COVID-19 Severe classes. In contrast, the COVID-19 Severity Detector does not utilize the COVID-19 Severe prediction of the COVID-19 Classifier but instead individually predicts the probability of a severe outcome within a month.

Sampling and Ensemble Learning Strategy

For both models, a 5-fold cross-validation was applied as a Bagging approach for ensemble learning. The aim was to create a variety of models which were trained on different subsets of the training data. Depending on the type of performance evaluation, two sampling strategies were performed. For the internal performance evaluation, the combined cross-validation folds for training were split into a training (70%) and validation subset (10%), whereas the remaining cross-validation fold (20%) was utilized as testing subset. This allowed the performance assessment of all samples in the dataset without introducing bias by computing predictions on the validation subset. However, for the final pipeline that was submitted to the STOIC challenge (external evaluation), a regular 5-fold cross-validation was performed in which an 80% training and a 20% validation subset were used for the training process in order to utilize the complete dataset. The prediction of the final COVID-19 Severity Detector pipeline comprises the mean-averaged sum of all five predictions from the five models of the cross-validation. This approach not only allows for more efficient usage of the available data but also increases the reliability of the prediction. However, for the COVID-19 Classifier, only the model with the best-monitored validation loss was used for predicting the COVID-19 Positive class in the final pipeline. Internal experiments revealed that this allowed achieving the best testing performance for COVID-19 presence prediction compared to utilizing a similar pooling-based prediction strategy like the COVID-19 Severity Detector.

Docker Environment

One of the key features of the proposed pipeline is full reproducibility in external IT environments. This was achieved by integrating the pipeline in a Docker environment [350]. The virtualization software Docker is recommended for reproducible research by various meta-analyses and guidelines to ensure reproducible research in machine learning as well as data science [42, 351–353]. The proposed pipeline can be utilized in two Docker containers

(packages). Whereas the first container is for re-training the pipeline to obtain fitted neural network models and to ensure reproducibility, the second container is for inference as well as to ensure reusability. The training container needs a thorax CT dataset with annotations about classification as well as metadata, and provides a directory with fitted models which can be passed to an inference container. The input of the inference container, which is illustrated in Figure 5.24, is a thorax CT scan and the patient's age as well as sex as metadata, whereas the output is the two predicted probabilities of COVID-19 infection and severity outcome within one month.

5.5.3 Results and Discussion

As dataset exploration, the provided features from the metadata as well as the computed ILR were analyzed in terms of correlation to the target variables (COVID-19 infection and severity). Therefore, the spearman correlation coefficients were computed and summarized in Table 5.11.

Table 5.11: Spearman correlation coefficients between patient features and target variables.

Feature / Class	COVID-19 Positive	COVID-19 Severe
Patient Age	0.0286	0.1503
Patient Sex	0.0627	0.1024
Infection-Lung Ratio (ILR)	0.3725	0.3005

The feature analysis revealed a small correlation between patient age and severe outcome with a coefficient of 0.15, whereas the ILR demonstrated a moderate correlation to infection and severity with coefficients of 0.37 and 0.30, respectively. This confirmed the value of computing the ILR and integrating it as metadata to the COVID-19 Severity Detector. Still, the ILR provided a stronger correlation to the infection status which is why it would have been a reasonable addition to the COVID-19 Classifier, as well.

The sequential training took less than 120 hours with 57.6 epochs on average for the COVID-19 Classifier and 61.8 epochs on average for the COVID-19 Severity Detector. The utilized hardware for the development and validation process was two NVIDIA QUADRO RTX 6000 with 24GB VRAM, an Intel Xeon Gold 5220R using 4 CPUs, and 40GB RAM. The implemented Docker containers were additionally reproduced on the following hardware setup: Two Tesla V100 GPUs of 32 GB VRAM each and 16 CPUs with a total of 128G RAM.

Internal Performance Evaluation

In order to reliably estimate the performance of the proposed pipeline, a 5-fold cross-validation was performed in which each fold was utilized as a hold-out set for testing. This allowed the computing of performance metrics on the complete dataset due to each sample was able to be used as testing sample when located in the hold-out fold and not be used in the training process. For the COVID-19 Classifier, a multi-class evaluation was performed in which the class with the highest predicted probability was used as outcome in the evaluation. As metrics, the Accuracy, F1-Score, and AUC were computed. For the final performance assessment, the

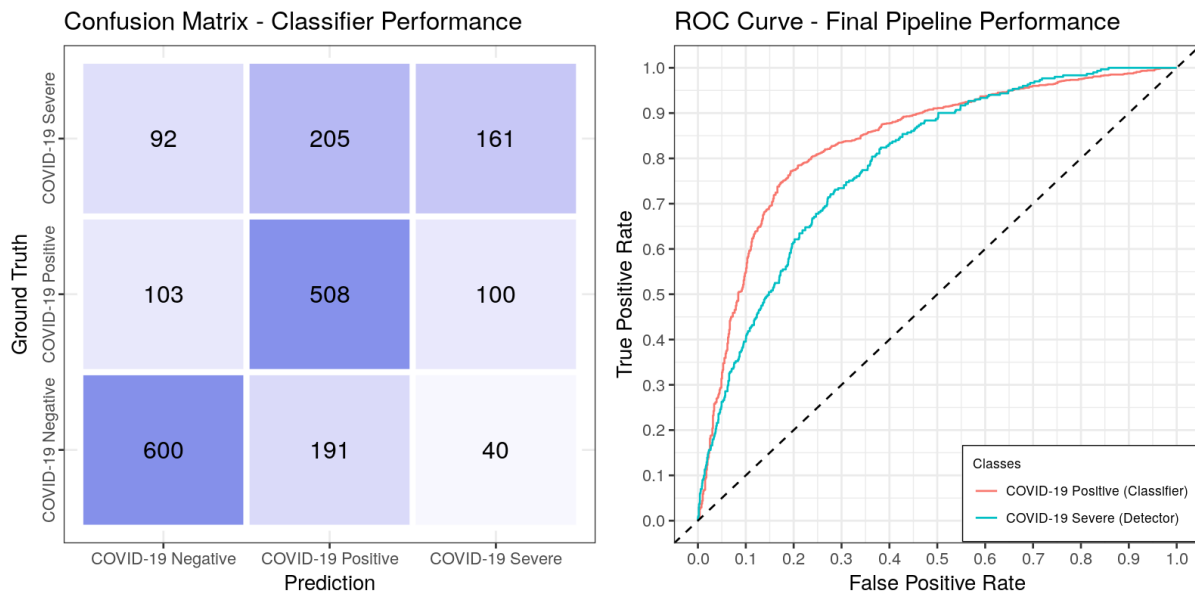


Figure 5.25: Results of the performance evaluation for COVID-19 infection and severe outcome prediction.

predicted probabilities for the COVID-19 infection and severe outcome were evaluated by the AUC metric. The computed results are summarized in Table 5.12 and visualized in Figure 5.25. The figure shows the performance illustrated through a confusion matrix (left plot) for the COVID-19 Classifier and ROC curves for the final predictions of the pipeline to COVID-19 infection and severe outcome.

The proposed pipeline demonstrated strong and robust classification performance for predicting COVID-19 infection and severe outcome within a month. The COVID-19 infection prediction achieved an AUC of 0.837, whereas the severe COVID-19 outcome prediction had an AUC of 0.790. In contrast, the COVID-19 Classifier obtained an AUC of 0.788 for severity outcome prediction. This is why a separate model for severity prediction with the COVID-19 Severity Detector was implemented instead of utilizing the inferior COVID-19 Severe class prediction of the COVID-19 Classifier. Overall, the accurate prediction of the severe outcome within a month is challenging. Analyzing the prediction results of the COVID-19 Classifier revealed that especially the differentiation between COVID-19 Positive and COVID-19 Severe patients is a hard and complex task.

Table 5.12: Computed performance results of the COVID-19 Classifier and Severity Detector.

COVID-19 Classifier			
Classes / Metrics	Accuracy	F1	AUC
COVID-19 Negative	0.787	0.782	0.837
COVID-19 Positive	0.701	0.629	0.766
COVID-19 Severe	0.782	0.424	0.788
Combination of COVID-19 Classifier and Severity Detector			
Method	AUC		
COVID-19 Classifier – Probability of COVID-19 infection	0.837		
COVID-19 Severity Detector – Probability of severe COVID-19 outcome	0.790		

External Evaluation through the STOIC Challenge

Next to the internal evaluation of the proposed pipeline, an external evaluation was performed by participation in the large STOIC2021 COVID-19 AI Challenge which is also endorsed by MICCAI [389]. The authors of the STOIC project released the dataset of 2,000 samples, which was also used for the training and internal evaluation, and hold back the remaining 8,735 samples for this challenge. The remaining data was divided into 200 samples for the first validation phase, around 800 samples for the second validation phase, and the remaining 7,000+ samples for the final testing phase [389]. Whereas the results of the first and second validation phases are already published, the testing phase is still active at this time (23.09.2022) [389]. Participation in the STOIC challenge allowed the independent evaluation of the predictive power of the proposed pipeline on an unseen and large testing set by objective external scientists. Furthermore, it also allowed the comparison with other state-of-the-art implementations from various labs around the world.

The results of the external performance evaluation are summarized in Table 5.13. The organizers defined the ranking order according to the achieved performance of the COVID-19 severity prediction measured by the AUC metric. The proposed pipeline was submitted as ‘uaux2’ with the affiliation ‘University of Augsburg’. The pipeline achieved rank 8 for the first validation phase with an achieved severity prediction performance of 0.789 AUC and rank 5 for the second validation phase with an achieved severity prediction performance of 0.766. For COVID-19 infection prediction, the 3rd best performance in the first validation phase with an AUC of 0.802 and the 2nd best in the second validation phase with an AUC of 0.826 was achieved. The collaboration in Augsburg implemented and submitted two pipelines: The proposed pipeline (uaux2) based on a standardized implementation with AUCMEDI and a custom implemented pipeline (‘Code 1055’) based on pure PyTorch [393].

Table 5.13: Top-10 ranking of the STOIC challenge including achieved prediction results (AUC).

1. Validation Phase				2. Validation Phase			
Rank	Team	COVID-19 Positive	COVID-19 Severe	Rank	Team	COVID-19 Positive	COVID-19 Severe
1.	Keya Medical (Flying Bird)	0.792	0.825	1.	Unknown Affiliation: Simon J (simon.j)	0.832	0.804
2.	Dublin City University (tmukande12)	0.771	0.822	2.	University of Augsburg (Code 1055)	0.607	0.787
3.	Korea Advanced Institute of Science and Technology (Letsur)	0.711	0.814	3.	NVIDIA (titericz)	0.809	0.784
4.	Unknown Affiliation: Simon J (simon.j)	0.801	0.813	4.	Vrije Universiteit Brussel (etro)	0.779	0.775
5.	Deakin University (deakin_team)	0.813	0.797	5.	University of Augsburg (uaux2)	0.826	0.766
6.	Unknown Affiliation: John Mani (ee13b088)	0.807	0.795	6.	Dublin City University (hal9000)	0.814	0.766
7.	Dublin City University (hal9000)	0.750	0.795	7.	Keya Medical (Flying Bird)	0.782	0.748
8.	University of Augsburg (uaux2)	0.802	0.789	8.	Synlab SDN IRCSS (SYNLAB-SDN)	0.796	0.737
9.	China: Southern Medical University (lichuanpugm@gmail.com)	0.801	0.789	9.	Shanghai Jiao Tong University (SJTU426)	0.782	0.729
10.	Shanghai Jiao Tong University (SJTU426)	0.774	0.784	10.	University of Alberta (UofA)	0.792	0.723

The external evaluation proved that the proposed pipeline has a strong performance and predictive capabilities which are comparable to the currently best implementations in the field [137, 389]. Furthermore, through the AUCMEDI standardization and Docker container deployment, the proposed pipeline successfully demonstrated its integration capabilities into clinical research IT infrastructure, provided by the challenge organizers, in which the majority of COVID-19 AI implementations today revealed to have significant problems [32]. Another noticeable point of the current challenge results is the ‘head-to-head’ performance for severity outcome prediction between the proposed standardized AUCMEDI implementation and a specialized as well as pure PyTorch implementation (Code 1055), which demonstrates the adaptability and powerfulness of the AUCMEDI framework.

5.5.4 Conclusions

In this study, a powerful pipeline for COVID-19 infection and severe outcome prediction was implemented which utilized the AUCMEDI framework and integrated ensemble, transfer, and deep learning techniques. The pipeline consisted of a multi-model workflow for individual prediction of the outcome variables to ensure utilization of the best-suited model design. The internal as well as external evaluation proved that it was possible to significantly contribute and the state-of-the-art of COVID-19 infection and severe outcome prediction in terms of performance and reproducibility. Furthermore, utilizing the Docker technology, the pipeline demonstrated strong reproducibility and reusability capabilities through successful integration into external clinical research IT infrastructure. Still, the pipeline can be further improved by integrating methods and applied configurations of the few higher ranking pipeline in the participated STOIC challenge. As future work, additional validation and integration in a clinical study are needed to identify actual medical gain in a clinical workflow.

5.6 Study: Informative Value of Explainable AI

Modern MIA models in histopathology were able to show impressive predictive capabilities and can be used for the automated recognition and classification of both learned and new patterns [187, 286, 394, 395]. In this application field, examinations for tumor diseases and predictive markers (biomarkers) are conducted using digitized tissue sections. Due to the high case numbers of more than 68,000 initial diagnoses per year in Germany [396] and an increasingly standardized diagnostic procedure, prostate cancer is well suited for deep neural network training, validation, and possible clinical application. Nagpal et al. [395] as well as Steiner et al. [394] have developed and evaluated a deep learning system for Gleason scoring [397] of prostate adenocarcinomas based on over 1,500 annotated, digitized tissue sections.

A fundamental problem in the development and application of AI algorithms, however, is that they resemble a ‘black box’ and it is difficult for the user to understand which criteria are used to make a machine decision [398]. In an AI-supported MIA, the selection of the image parts used by the algorithm (ROI) and the specification of the (un)safety factor used to classify an image or an ROI is of particular interest to the user. For example, the explainable AI algorithm Grad-CAM [344] provides a representation of which image areas the neural network is ‘looking at’ by overlaying a heatmap on the input image. Algorithmic, computer-aided methods offer the possibility to assist pathologists in decision-making, whereby a transparent justification for the assessment of the tissue plays a decisive role in the acceptance of such a CDS system. Without intuitive, transparent visualizations of decision factors, users will always be skeptical about deep learning models, which makes it difficult to establish and build trust in AI-supported CDS systems in clinical routine.

In this clinical study, a joint effort of the University of Augsburg, in which the Author represents the head development for AI as well as XAI, and the University Hospital Augsburg pushed forward developing an adaptive deep learning algorithm that can use image data from prostate tissue sections to classify the Gleason score [397] but focusing on the informative value analysis of explainable AI methods for clinicians. The following interim report is a summary by the Author about the currently ongoing clinical study that is approved by the ethics committee of the Ludwig Maximilian University of Munich and funded by the Faculty of Medicine of the University of Augsburg.

This project aims to contribute to the following research questions:

- How reproducible are the results of Nagpal et. al. [395]?

Although the authors have successfully demonstrated the application of AI algorithms for Gleason scoring, they have not published the deep learning model itself or a proof-of-concept application due to intended commercial exploitation. This project would like to test whether it is possible to reproduce and achieve high-quality classification of tumor tissue using a self-developed, open-source deep learning pipeline on an initially smaller, manually annotated data set of prostate carcinoma tissue sections.

- Which explainable AI approaches are suitable for the use case of prostate cancer?

There are a variety of different algorithms to highlight image pixels used for classification by a deep neural network model. This project aims to identify the methods suitable for the use case of prostate cancer and implement them in a proof-of-concept application for evaluation.

- How understandable, transparent, and helpful are the visualizations by explainable AI methods for pathologists in terms of their own decision-making?

This project aims to systematically evaluate, involving pathologists, whether the diagnostic or prognostic assessment of the deep neural network model is comprehensible using user-friendly implementations of explainable AI visualizations of the relevant criteria and thus analyze the research question, whether a CDS system based on this algorithm would experience acceptance.

5.6.1 Clinical Study Design

The project, which has the code title ‘EKIPRO’ (an abbreviation for the German project title “*Klinische Entscheidungshilfen dank erklärbarer Künstlicher Intelligenz am Beispiel des Prostata-Karzinoms*”), includes the development of an AI-based system that can make diagnostic and prognostic statements based on histopathological images. These statements should be presented in a precise, user-friendly, and comprehensible manner for the user. The evaluation of prostate carcinoma tissue according to the Gleason score will serve as an application. In addition to the development and implementation of a deep neural network algorithm for use on digitized tissue sections, the focus is on transparent and intuitive representations of the decision criteria used by the algorithm. These are evaluated iteratively by practicing pathologists and are thus continuously adapted to the needs of routine clinical practice with regard to possible use as a medical device. The study is organized into the following four modules representing major steps in the workflow.

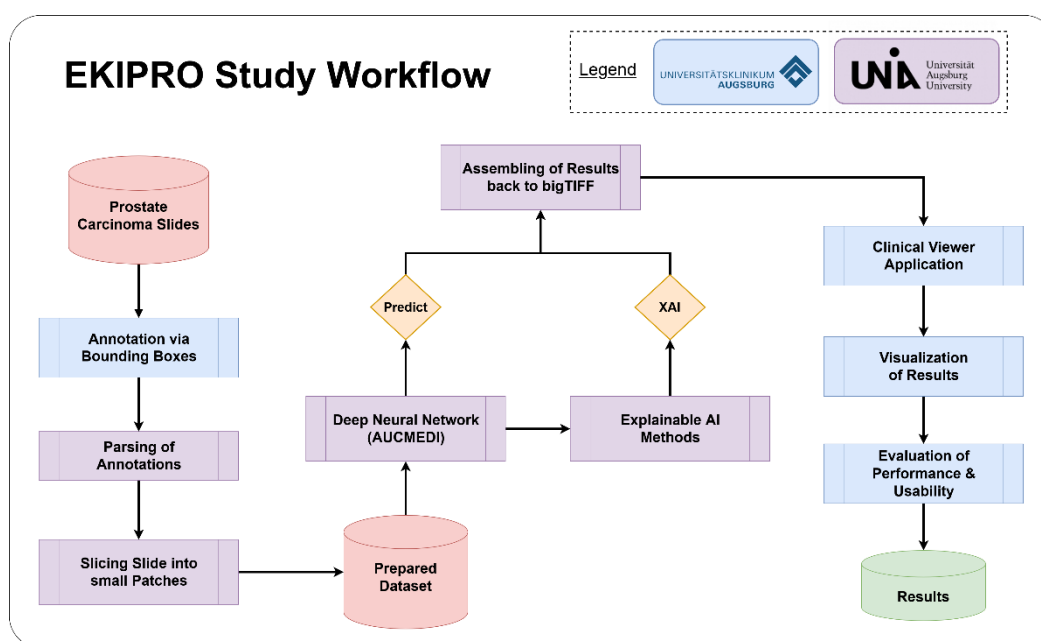


Figure 5.26: Flowchart diagram of the EKIPRO project.

Module: Dataset Creation and Annotation

It is planned to extract at least 1,000 suitable tissue samples from the archives of the Institute for Pathology and Molecular Diagnostics at the University Hospital Augsburg. Afterward, the extracted tissue samples are stained if necessary (with p63 or RaceMase for invasive carcinoma or preliminary stages), digitalized, and annotated.

Module: Medical Image Classification

The resulting annotated microscopy slides are converted into a suitable data format which includes a slicing procedure in order to obtain adequate patches which are able to fit into a common deep learning model. Utilizing the AUCMEDI framework, a state-of-the-art deep neural network based medical image classification pipeline is developed, trained, and validated on the annotated imaging dataset.

Module: Explainable AI and Visualization Software

Based on a trained deep neural network model, several explainable AI algorithms are applied including Grad-CAM [344], Grad-CAM++ [345], Saliency Maps [347], Guided Backpropagation [346], and Integrated Gradients [348]. Furthermore, a proof-of-concept software for visualization is implemented which allows user-friendly evaluation by pathologists and will be published as open-source code.

Module: Evaluation

Finally, the deep neural network model performance, the explainable AI approaches, the XAI visualization, and the proof-of-concept implementation are evaluated iteratively by pathologists in the context of the suitability for prostate carcinoma application. For this purpose, consortium partners, in particular the NCT-WERA association (National Center for Tumor Diseases association between Würzburg, Erlangen, Regensburg, and Augsburg in Germany) and the BZKF (Bavarian Cancer Research Center), are involved. The detailed results of this evaluation will be published in a standardized report according to STARE-HI [399].

5.6.2 Dataset

In the retrospective data extraction process, it was possible to initially identify 325 prostate carcinoma cases from 11.05.2016 – 10.05.2021 in the laboratory information system of the Institute for Pathology and Molecular Diagnostics at the University Hospital Augsburg. A total of 1,212 slides of H&E-stained (hematoxylin- and eosin-stained) histological sections from prostate samples are assigned to these cases, with Gleason scores from the available reports being annotated for each slide. 620 of these slides were extracted from the pathology archive and digitized. Further slides can be digitized at short notice if additional image data is required as the project progresses. 369 of the 620 digitized slides were comprehensively annotated by a pathologist from the Institute for Pathology and Molecular Diagnostics at the University Hospital Augsburg. Here, image regions were marked via bounding boxes and assigned to one

of the following classes. Regular representing normal tissue, carcinoma tissue according to Gleason score 3, 4, or 5, artefacts such as air pockets or contamination of the slide, and questionable regions that cannot be clearly assigned to any other class.

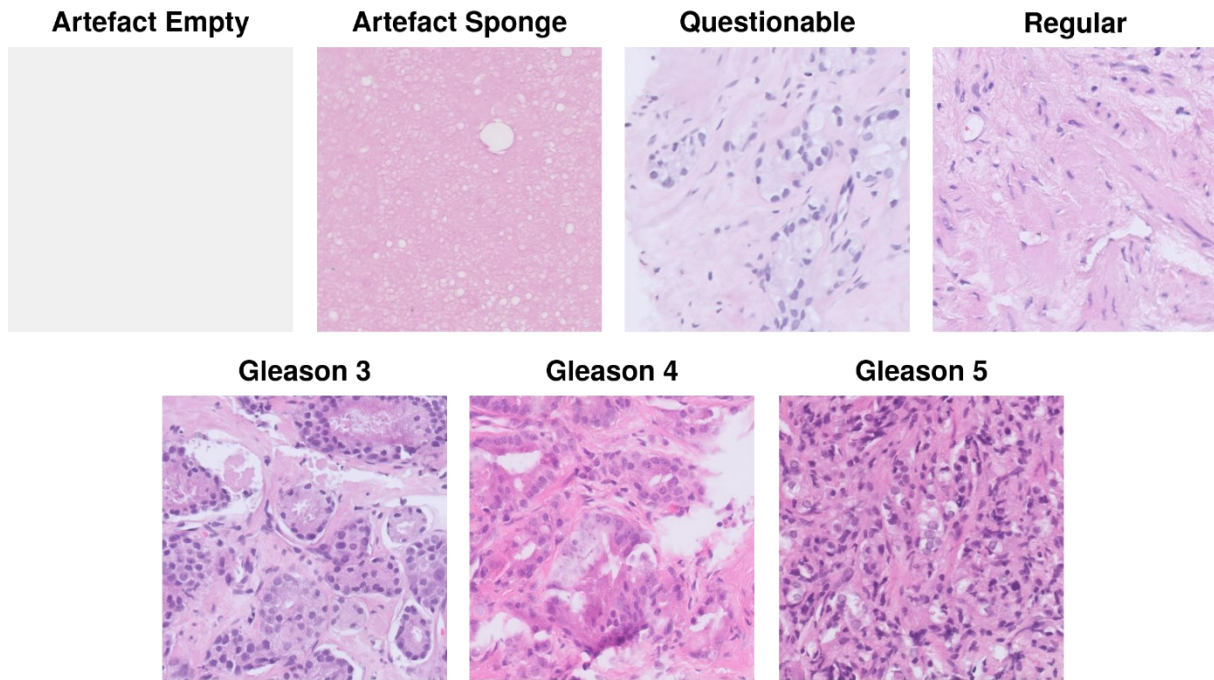


Figure 5.27: Visualization of tiles from histological sections generated in the EKIPRO project.

Preprocessing and Sampling

The annotated high-resolution digitized slides were preprocessed for the data interface of the deep neural network model. Thereby, each slide image was divided via patching into tiles with a size of 1024x1024 pixels (corresponds to 256x256 μm on the slide). Each tile was assigned a class according to the annotated bounding boxes if more than half of the tile’s area was annotated with a single class. This procedure allowed featuring of superordinate as well as also fine-detailed spatial structures in the corresponding tiles. As a result, it was possible to generate 78,564 patches from the 369 slides. For reliable performance validation later, the set of patches was sampled according to a 62-15-23 stratified percentage split resulting in 48,396 patches for model training, 12,100 for validation, and 18,068 as a hold-out set for testing.

Table 5.14: Summary of the EKIPRO dataset annotation and proposed sampling.

Annotation	Training Set	Validation Set	Testing Set	Frequency	Total
Regular	5,031	1,258	1,627	10.1 %	7,916
Gleason 3	600	150	234	1.3 %	984
Gleason 4	1,819	455	604	3.7 %	2,878
Gleason 5	1,778	445	344	3.3 %	2,567
Questionable	210	53	119	0.5 %	382
Artefact Empty	1,888	472	2,127	5.7 %	4,487
Artefact Sponge	37,070	9,267	13,013	75.5 %	59,350
Sum	48,396	12,100	18,068	100.0%	78,564

5.6.3 Application

Based on the AUCMEDI framework, a system for the classification of ROIs in histological cross-sections based on the Gleason score utilizing deep neural networks could be developed. The AI pipeline consists of the following core modules: Preprocessing, neural network models, and explainable artificial intelligence. To reduce the complexity and increase the overview of the procedures applied in this clinical study, only methods relevant to the AUCMEDI application are described in this subchapter.

Preprocessing and Image Augmentation

In order to further increase the performance and reliability, multiple preprocessing methods were exploited before passing the images to the neural network model.

For the training process, an extensive on-the-fly image augmentation procedure was applied including flipping and rotation augmentations. Other augmentation methods were excluded to avoid introducing artificial bias into the images. For preprocessing, the tiles were resized to 224x224 pixels according to the pretrained neural network architecture input shape. Subsequently, the pixel value intensities of the RGB tiles were normalized via the Z-Score normalization based on the ImageNet [76] mean and standard deviation.

Neural Network Models

For the development of a reliable and high-performing MIC model, a total of four different neural network architectures were validated by comparing the predictive abilities. The following architectures were utilized: DenseNet121 [90], EfficientNetB2 [79], ResNet101 [80], and ResNeXt101 [88].

For the training process of the four pipelines, the following state-of-the-art procedure was applied. Transfer learning fitting of the classification head was conducted for 10 epochs based on weights obtained from the ImageNet dataset [76] using the Adam optimization [201] with an initial learning rate of $1E^{-4}$. Afterward, a fine-tuning fitting was conducted on the complete architecture with a maximal training time of 1,000 epochs and a dynamic learning rate starting from $1E^{-5}$ to a maximum decrease of $1E^{-7}$ (decreasing factor of 0.1 after 5 epochs without improvement on the monitored validation loss). Moreover, an early stopping technique was applied that stopped the training process after 15 epochs without validation loss improvement. The training was performed with a batch size of 28 samples, utilized the traditional epoch definition, and applied the weighted Focal loss by Lin et al. [213].

Explainable Artificial Intelligence

The explainable AI module of AUCMEDI allowed not only the deep neural network classification of the tiles obtained from the prostate samples but the generation of heatmaps for these tiles to highlight the important regions for the decision-making of the model. The five explainable AI methods Grad-CAM [344], Grad-CAM++ [345], Saliency Maps [347], Guided

Backpropagation [346], and Integrated Gradients [348] were integrated into the developed pipeline of the clinical study. Subsequently to the model prediction process, the XAI methods were applied to evaluate visualization methods for the graphical user interface of the proof-of-concept software. As an output of the image analysis pipeline, the classification predictions and XAI heatmaps are generated for the 1024x1024 pixel tiles.

5.6.4 Current Outcomes and Insights

The EKIPRO project is a still ongoing clinical study which is why the following presented interim outcomes are not final and represent insights into the current state of the project. As the module ‘Dataset Creation and Annotation’ has been completed for the most part and is described in Chapter 5.6.2, this subchapter focuses on the ‘Medical Image Classification’ and ‘Explainable AI and Visualization Software’ modules.

Module: Medical Image Classification

Through the proposed framework AUCMEDI, it was possible to setup a powerful prostate cancer classification pipeline based on the Gleason score. The developed models based on four different neural network architectures were successfully trained and evaluated on the hold-out testing set (18,068 tiles). For performance assessment, the Accuracy, F1-score, and AUC were computed for each class as well as for each architecture, individually. The achieved results for the performance evaluation of the architectures are illustrated by ROC curves in Figure 5.28 and by an overview of the detailed scores in Table 5.15.

In the architecture comparison, the ResNeXt demonstrated superior performance with a macro-averaged F1-score of 0.756, AUC of 0.989, and Accuracy of 0.982. In contrast, the EfficientNetB2 performed the worst, whereas the ResNet101 also revealed low prediction capabilities. The DenseNet121 was able to also achieve strong performance and even the best class-wise F1-score for ‘Gleason 5’ detection with 0.826 compared to 0.805 by the ResNeXt. According to the scores, the most challenging classes to identify by the AI model classifier were ‘Gleason 3’ and ‘Questionable’. The ‘Questionable’ class was expected to be more difficult or inexplicable as the pathologist annotator of the dataset labeled any region that could not be

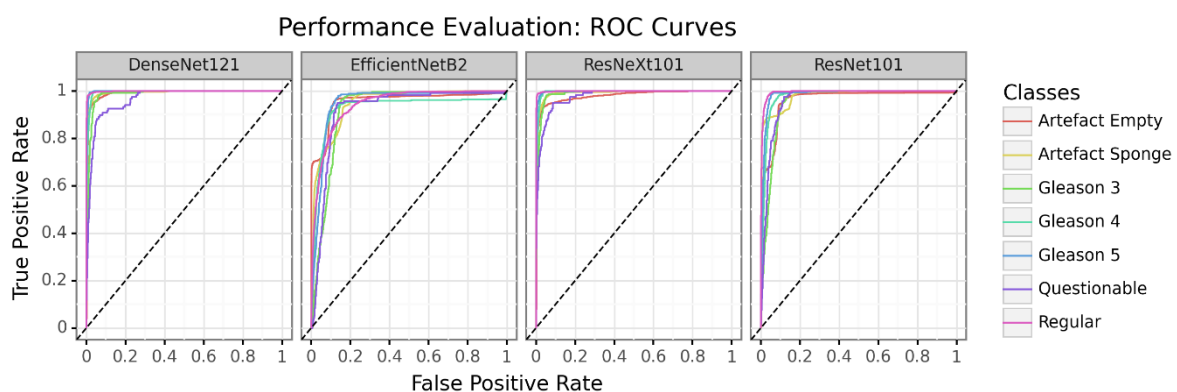


Figure 5.28: Receiver operating characteristic curves for each architecture validated in the EKIPRO study.

annotated to a specific class to this corresponding class. This is why the detailed analysis of the ResNeXt predictive capabilities in Figure 5.29 showed that the model almost evenly classified tiles annotated as ‘Questionable’ to the remaining classes (excluding the artefact classes). In contrast, the ResNeXT model mistook the ‘Gleason 3’ class, which also showed lower performance than the other classes, for ‘Gleason 4’ or ‘Regular’. This scenario can be explained through the ordinal grading of the Gleason score and revealed that the regions share high similarities with the adjacent Gleason classes representing a challenging task for reliable distinguishment.

Table 5.15: Achieved results of the proposed image classification for prostate cancer.

Class	DenseNet121			EfficientNetB2			ResNeXt101			ResNet101		
	Accuracy	F1	AUC	Accuracy	F1	AUC	Accuracy	F1	AUC	Accuracy	F1	AUC
Regular	0.992	0.958	0.999	0.921	0.578	0.941	0.993	0.962	0.999	0.974	0.865	0.995
Gleason 3	0.986	0.447	0.985	0.987	0.000	0.920	0.989	0.541	0.988	0.979	0.204	0.956
Gleason 4	0.985	0.777	0.995	0.963	0.354	0.925	0.985	0.785	0.995	0.970	0.389	0.978
Gleason 5	0.993	0.826	0.998	0.980	0.132	0.947	0.993	0.805	0.997	0.981	0.528	0.986
Artefact Empty	0.959	0.803	0.990	0.951	0.739	0.954	0.960	0.806	0.978	0.925	0.677	0.961
Artefact Spoon	0.959	0.972	0.995	0.911	0.942	0.944	0.959	0.972	0.991	0.920	0.945	0.979
Questionable	0.991	0.205	0.965	0.993	0.000	0.917	0.994	0.418	0.976	0.988	0.122	0.962

The clinical study design intended to train multiple deep neural network architectures for the later selection of the best-performing one according to the quantitative performance assessment on the hold-out set testing set instead of utilizing multiple models in an ensemble learning pipeline structure. As the majority of explainable AI methods, in particular gradient-based methods, are based on analyzing the neural network model, the study design purposed the utilization of a single model in order to reduce complexity. This is why the ResNeXt architecture was selected as the final model for the pipeline.

The methodologies of Nagpal et al. [395] and the proposed MIC pipeline share fundamental techniques in the design of the deep neural network but also revealed multiple differences. The authors utilized more extensive image augmentation with random saturation, contrast, brightness, and hue modifications. This was neglected in the proposed pipeline design to avoid bias introduction but could be further experimented with in the future study progress. Furthermore, the tile shape, which is processed by the neural network model, is covering a large area of the slide with 911x911 μm in contrast to 256x256 μm of the proposed pipeline which could lead to a considerable information loss through the low resolution of 911x911 pixels. Moreover, the performance of the pipeline is further improved through the utilization of ensemble learning which was avoided by the proposed pipeline in EKIPRO to avoid the increase of explainable AI complexity. As architecture, Nagpal et al. [395] utilized the InceptionV3 [87] which is inferior to more recent architectures like ResNeXt [88] according to the ImageNet benchmarking [76]. Nevertheless, the integration of the InceptionV3 architecture into the pipeline is currently work in progress for ensuring better comparability. Another difference in the pipeline by Nagpal et al. [395] is the multi-phase setup with a subsequent k-

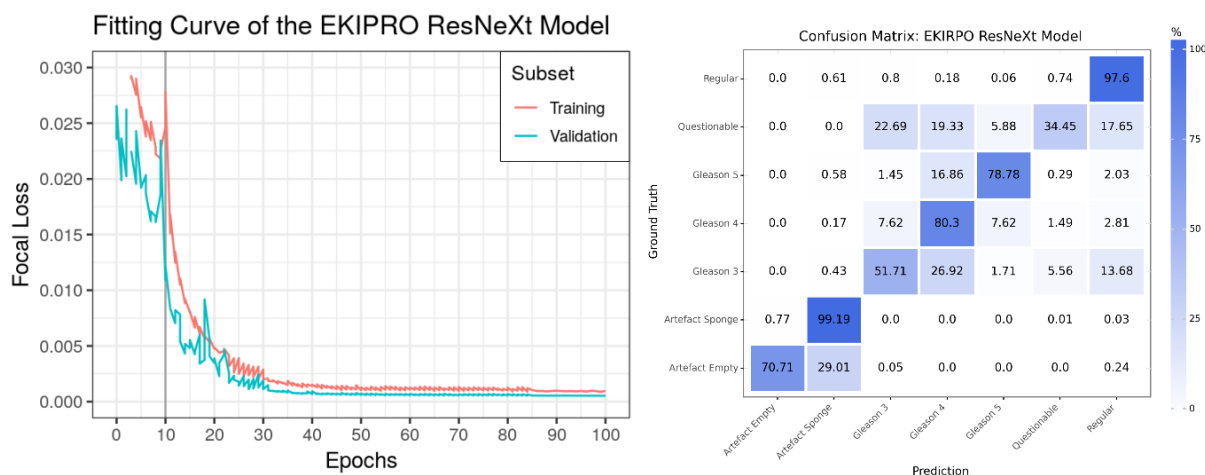


Figure 5.29: Fitting curve and confusion matrix of the EKIPRO ResNeXt model.

nearest neighbor classification for the final Gleason grading of a whole slide by combing the predicted classifications of multiple regions. Such workflow is capable of fully automated Gleason grading for a whole slide instead of just patch-based Gleason grading as currently only supported by the proposed pipeline. Comparing the dataset between the studies showed that Nagpal et al. [395] focused on purely tiles annotated with Gleason scores, whereas the EKIPRO study also included realistic use cases like artefacts or questionable regions. The comparison of the Gleason grading performance with the Nagpal et al. [395] approach is complex due to the different tile resolutions as well as annotation procedures. The authors reported a tile-wise Accuracy of 0.720 indicating that the proposed model is superior with an Accuracy of 0.99. However, as already mentioned, the difference in the annotation procedure highly biases these results making them incomparable. It has to be noted that these evaluation results represent only current insights and are not final. These pre-evaluation results were generated through quantitative evaluation by assessing the performance of the tiles from the in-house microscopy sections. For reliable comparison, a qualitative evaluation by pathologists is mandatory and is part of the ‘Evaluation’ module which is still work in progress.

Module: Explainable AI and Visualization Software

Through the successful setup of a high-performing MIC pipeline for reliable prostate carcinoma prediction, it was possible to start generating XAI heatmaps for the selected XAI methods. However, the generation process is more time-intensive to compute as each tile for a whole slide has to be processed by the neural network model as well as by the XAI method in order to reassemble the generated XAI tiles into the original slide shape. Thus, this process is currently work in progress. The first experiments indicated that this assembling process represents a challenge that still has to be overcome, as the heatmap weighting of each tile is local and independent of other tiles. Therefore, four methods are currently being developed to normalize the local weighting of the individual tiles to a global weighting for the overall picture.

Simultaneously apart from the AI and XAI development by the Author, a browser-based proof-of-concept application is currently being developed for the display and evaluation of the AI-

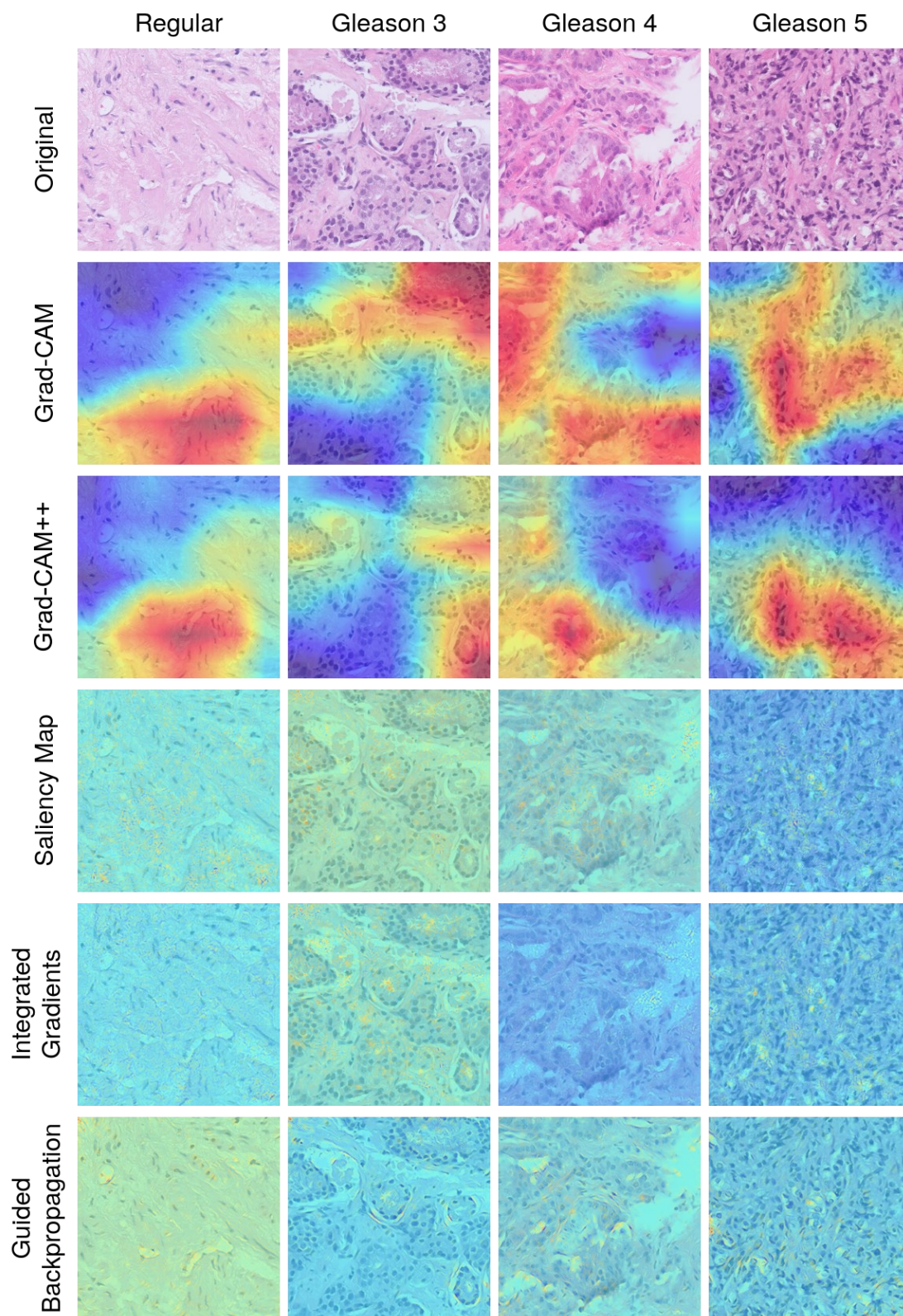


Figure 5.30: Generated XAI heatmaps based on the ResNeXt model for prostate carcinoma classification.

supported image classification as well as the explainable AI visualizations. To determine the expectations and requirements of the users, structured interviews with assistant physicians, specialists, and senior physicians in the pathology department of the UK Augsburg were conducted. Accordingly, a list of the most important features was created and conceptual

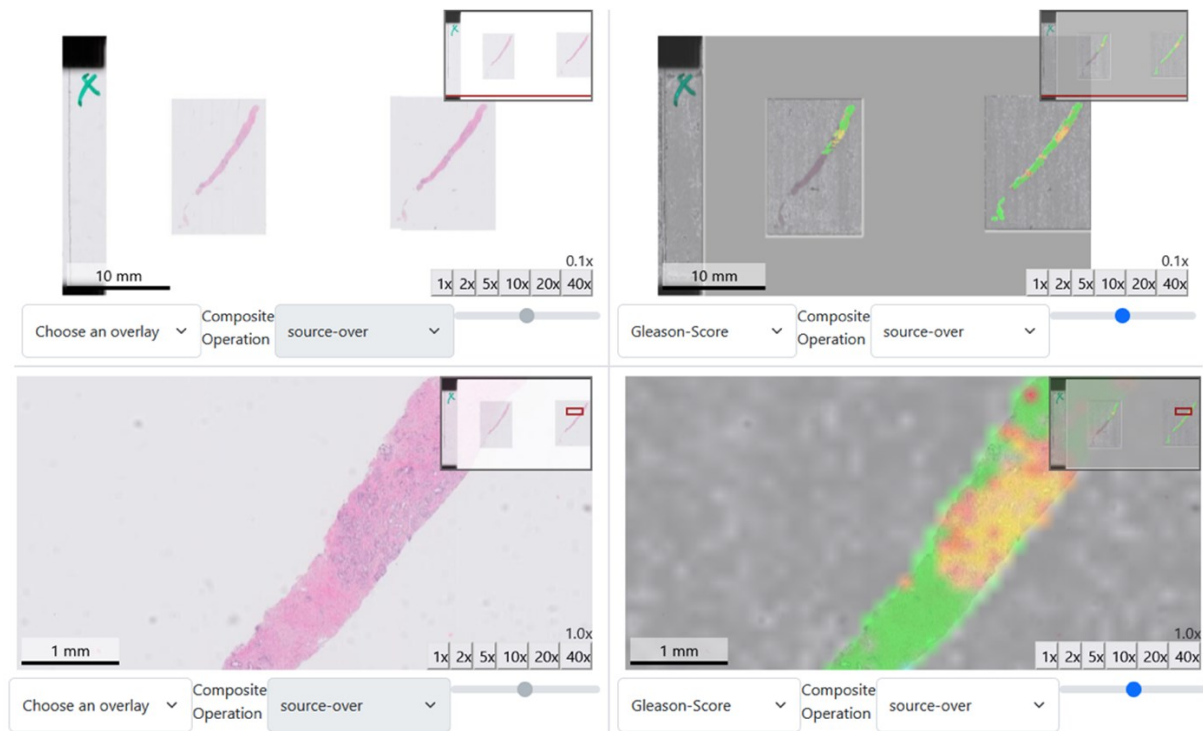


Figure 5.31: Prototype of the EKIPRO browser application for visualizing microscopy and XAI imaging.

designs for the graphical user interface were created. The application design is based on these drafts, whereby utilizing existing, freely available tools and frameworks for rapid development. Additionally, the application has a modular design so that it can also be used for other projects in the field of digital pathology with minor modifications.

5.6.5 Future Outlook

It was possible to develop an MIC pipeline utilizing AUCMEDI that demonstrated excellent Gleason grading performance as well as was able to reproduce the results by Nagpal et al. [395] in terms of predictive capabilities and general deep neural network based methodology. The major difference is currently the tile-based (patch-based) analysis compared to the whole-slide approach by Nagpal et al. [395]. However, focusing on the tile-based approach with a single deep neural network model is more reasonable as EKIPRO studies gradient-based explainable AI methods. Still, the proposed pipeline lacks capabilities for fine-distinguishment of ‘Gleason 3’ regions. This could be further improved in this study by integrating more detailed imaging data which can be achieved by a higher zoom factor like $128 \times 128 \mu\text{m}$ or smaller tiles resulting in reduced information loss through resizing. Furthermore, it is planned to apply the InceptionV3 [87] architecture, as well, to increase comparability with Nagpal et al. [395].

The ‘Explainable AI and Visualization Software’ and ‘Evaluation’ modules are still actively developed and currently work in progress which is why more detailed insights and possible outcome reports are defined as future work. Overall, the EKIPRO study offers promising insights contributing to the field of XAI and clinical application. The project can have a significant impact on the future design of clinical applications by revealing the true informative value of the studied XAI methods for clinicians.

6

Ensemble Learning

In the field of machine learning, the aim is to find a suitable hypothesis that maximizes prediction correctness. However, finding the optimal hypothesis is difficult which is why the strategy evolved to combine multiple hypotheses into a superior predictor closer to an optimal hypothesis. In the context of deep convolutional neural networks, hypotheses are represented through fitted neural network models. Thus, ensemble learning is defined as the combination of models to yield better prediction performance. The integration of ensemble learning strategies in a deep learning based pipeline is called deep ensemble learning. Recent studies showed that the most successful and accurate medical image analysis pipelines are heavily based on ensemble learning strategies [187, 189, 400–404].

In this chapter, the Author provides a quick introduction to ensemble learning and presents two studies analyzing the performance impact of ensemble learning strategies on deep learning based medical image analysis. The studies contribute to the active research field of deep ensemble learning.

6.1 Idea

In contrast to the regular approach of preprocessing and model enhancement for obtaining an improved model performance, the contrary approach is to enhance the inference via postprocessing methods. One of the most popular strategies to enhance inference accuracy is ensemble learning. Ensemble learning can be defined as building an ensemble of multiple predictions and combining them into a single prediction. In the classic ensemble learning definition, the multiple predictions originate from different machine learning algorithms [338–341]. However, reasonable input augmentation, different hyperparameter configurations, or diverse training data also allow variable inference based on the same algorithm. The ensemble of predictions for a single sample impacts the overall performance through improved inference robustness, variance as well as bias reduction, search space decomposition, better data utilization, and exploiting the strengths of multiple models [338–341]. Deep ensemble learning is traditionally defined as building an ensemble of multiple predictions originating from different deep convolutional neural network models [338]. However, recent novel techniques necessitate redefining ensemble learning in the deep learning context as combining information, most commonly predictions, for a single inference. This information or predictions can either originate from multiple distinct models or just a single model.

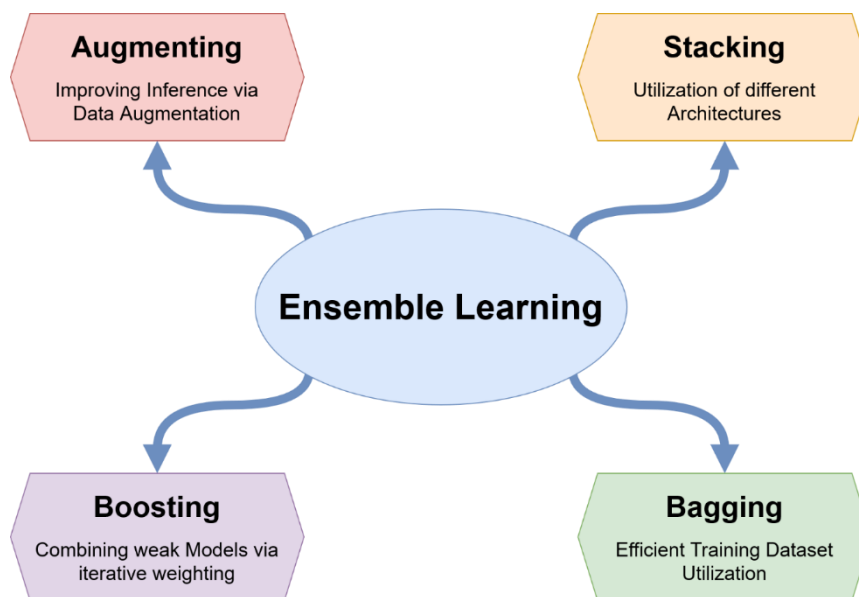


Figure 6.1: Overview of strategies in the field of ensemble learning.

Various recent studies successfully utilized ensemble learning strategies to improve the performance and robustness of their MIA pipeline [187, 189, 247, 340, 400–406]. The underlying techniques of these deep ensemble learning based pipelines are ranging from the combination of different model types like in the studies Rajaraman et al. [407] and Hameed et al. [187] to inference improvement of a single model like in Galdran et al. [408]. Furthermore, medical imaging datasets are commonly quite small, which is why ensemble learning techniques for efficient training data usage are especially popular as demonstrated in Bibaut et al. [340] and Müller et al. [159].

6.2 Methods

The field of ensemble learning describes the process of combining multiple predictions generated by one or commonly multiple machine learning models. The process of ensemble learning consists of two phases: The applied ensemble learning strategy defines from which models or sources the predictions are generated, and the applied pooling function defines the method of how the predictions are merged into a single one. This chapter introduces the prevalent ensemble learning strategies for deep neural networks and provides a brief overview of popular pooling functions.

6.2.1 Ensemble Learning Strategies

Over the years, the field of general machine learning has proposed various strategies based on multi-model prediction. Still, the majority of these methods can be categorized into three strategies: Stacking, Boosting, and Bagging. Depending on the definition of the term ensemble learning, the Augmenting strategy represents an exception due to its single model concept. The Boosting technique, which is also commonly used in ensemble learning for general machine learning, was excluded from this summary. The reason for this is that Boosting is not feasibly applicable for image analysis with deep convolutional neural networks due to the extreme increase in training time [338, 339].

Augmenting

The Augmenting technique, often called test-time data augmentation, can be defined as the application of reasonable image augmentation prior to inference [163, 409–413]. Through augmentation, multiple images of the same sample can be generated and then used to compute multiple predictions. Contrary to other ensemble learning strategies, Augmenting is based on a single model instead of the traditional multi-model structure. Even so, the classification of

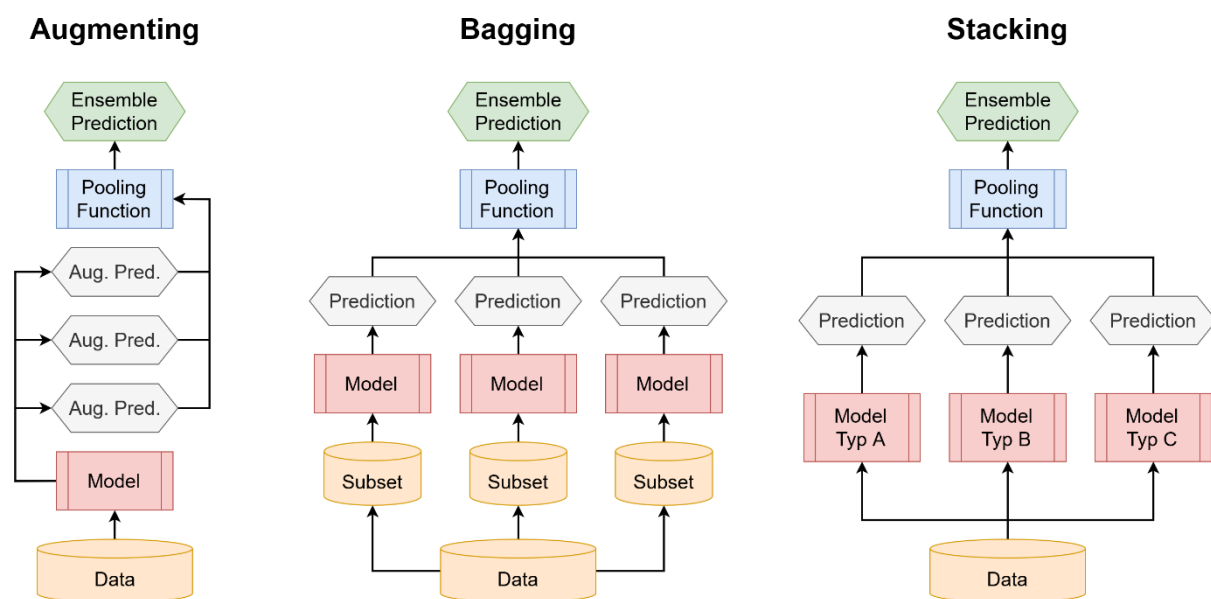


Figure 6.2: Workflow illustration of deep ensemble learning strategies.

Augmenting as an ensemble learning technique is debatable and depending on the used definitions, it is still one of the most popular applied multi-prediction combination strategies for deep learning based MIA pipelines [109, 409, 411, 412] and, thus, should be compared next to other similar approaches for generating multiple predictions for a single sample. Popular image augmentation methods for inference augmenting are rotations and mirroring. The aim of augmenting is to reduce the risk of incorrect predictions based on overfitting or too strict pattern learning [409, 410, 412]. In theory, this should be already avoided with standard data augmentation during the training process. However, strong performing medical image classification and segmentation pipelines revealed that model performance is still able to benefit from inference augmenting [163, 409–413].

Bagging

Homogeneous ensemble learning can be defined as multiple models consisting of the same algorithm, hyperparameters, or architecture [338, 406]. The Bagging technique is based on improved training dataset sampling and a popular homogeneous ensemble learning technique. In contrast to a standard single training/validation split, which results in a single model, Bagging consists of training multiple models on randomly drawn subsets from the dataset. In practice, k -fold cross-validation is applied to the dataset resulting in k models [414]. Afterward, each model can be used to compute an equally weighted prediction for a single sample. The benefit of Bagging is that it allows more efficient usage of the available data, which results in variance reduction and improved inference robustness.

Stacking

In contrast to single algorithm approaches, the ensemble of different deep convolutional neural network architectures (also called inhomogeneous ensemble learning) showed strong benefits for overall performance [338, 339, 341, 400, 403]. This kind of ensemble learning is more complex and can consist of even different computer vision tasks [338, 341, 403]. For medical image analysis, there are three ensemble learning techniques based on this principle. Traditional Stacking combines multiple independent models based on different architectures or hyperparameters, often including another machine learning algorithm on top. This technique of Stacking is utilized in the further studies of this thesis. Still, in a Stacking approach, each model is able to provide an independent prediction for the sample. In contrast, the one-vs-one ensemble approach is a popular binarization strategy for multi-class problems in which a binary model is trained for each class. Only the combination of all binary models in a one-vs-one ensemble produces a valid prediction. Furthermore, complex ensembles based on the cascading technique have become quite popular since recent years [109, 111, 148, 248, 415, 416]. Cascading ensembles are defined as a pipeline or sequential alike structure in which models use the predicted output of prior models. A popular application of cascading ensembles is the combination of classification and segmentation architectures. These pipelines showed that prior image classification on a sample can reveal important information for a further segmentation model [416, 417].

6.2.2 Pooling Functions

In order to combine the ensemble of predictions into a single one, several different methods and algorithms can be used. This subchapter provides a quick overview of commonly utilized methods for prediction combination, which are defined as ‘pooling functions’ in this thesis. A summary of the presented pooling functions can be found in Table 6.1.

Table 6.1: Overview of pooling functions for ensemble learning.

Pooling Function	Function Type	Applicable in Task
Best Model	Metalearner Function	Classification & Segmentation
Decision Tree	Metalearner Function	Classification
Gaussian Process	Metalearner Function	Classification
Global Argmax	Aggregate Function	Classification & Segmentation
Logistic Regression	Metalearner Function	Classification
Majority Voting Hard	Aggregate Function	Classification & Segmentation
Majority Voting Soft	Aggregate Function	Classification & Segmentation
Mean Unweighted	Aggregate Function	Classification & Segmentation
Mean Weighted	Metalearner Function	Classification & Segmentation
Median	Aggregate Function	Classification & Segmentation
Naive Bayes	Metalearner Function	Classification
Neural Network	Metalearner Function	Classification & Segmentation
Support Vector Machine	Metalearner Function	Classification
k-Nearest Neighbor	Metalearner Function	Classification

Aggregate Functions

Aggregate functions are homogenous pooling functions in which each model is equally weighted and each prediction equally contributes to the combined result. Typically, aggregate functions are more simple and intuitive pooling functions like averaging. The advantage of aggregate functions is that no further data or model training is required, whereas the disadvantage is that low-performing or biased models can decrease the overall performance of the ensemble.

Metalearner Functions

Metalearner functions are heterogenous pooling functions in which the contribution of the individual predictions to the combined result is unequal. This means that predictions of some models are more important than others and, thus, have a higher influence on the outcome prediction. Furthermore, metalearner functions are models which require fitting or weighting before usage. Typically, metalearner functions are machine learning models. The advantage of metalearner functions is that complex prediction patterns can be detected to identify the correct prediction, whereas the disadvantage is further data and model training are required.

Methods

A pooling function can be defined as a function, which given multiple predictions represented through 1D vectors returns a single vector in the same dimension but not necessarily in the same shape. Accordingly, there is a wide spectrum of functions that are able to be utilized for pooling. However, only a handful are commonly applied in deep ensemble learning [339–341]. Theoretically, the definition of the corresponding strategies also determines the function type and, thus, if the ensemble learning strategy can utilize aggregate or metalearner functions. However, any pooling function can be implemented for any ensemble learning strategy in practice.

A prediction in MIA is represented through one or multiple percentage values and determines the classification of the element. In image classification, predicted classifications are stored in a 1D vector in which each element represents the probability of the sample to be assigned to the corresponding class, whereas in image segmentation, predicted masks are stored in a 2D or 3D vector in which each element represents the probability of the pixel to be assigned to the corresponding class. Commonly, these percentage values result from the softmax or sigmoid activation function of the neural network model. However, the pooling function is applied to each element, individually, which means that pooling is applied on a sample for image classification and on a pixel for image segmentation. Even so, there are complex pooling functions that allow global pooling of multiple pixels in image segmentation (like cascading segmentation models) [109, 148, 415], this chapter focuses on traditional pooling functions for ensemble learning.

The majority of ensemble learning pipelines heavily utilize averaging and majority voting [340, 341]. Averaging approaches are the Unweighted Mean which straightforwardly averages the class probabilities across elements, and the Weighted Mean which performs a weighted averaging according to prior provided weights for the classifier. Similar to the Unweighted Mean, Median averaging can also be applied. For majority voting, there are two implementations depending on vote accumulation. The Majority Vote Soft variant sums up all probabilities per class and then softmax normalizes them across all classes, whereas the Majority Vote Hard variant utilizes traditional class voting in which the class with the highest probability is used for each element as a vote. Besides averaging and majority voting, Global Argmax and Best Model are also more intuitive pooling functions. Global Argmax is defined as selecting the class with the highest probability across all elements and zeroing the remaining classes. The Best Model technique is selecting the best scoring model according to a prior evaluated metric. In contrast to the intuitive pooling functions, it is also possible to utilize an additional machine learning classifier for pooling. Commonly used machine learning models are: Decision Tree [405], Gaussian Process classifier [377], Logistic Regression, Naïve Bayes, Support Vector Machine [418], and k-Nearest Neighbors [342]. Especially for Stacking based MIS pipelines, the utilization of another deep convolutional neural network model for further processing of the original predictions can improve prediction quality (cascading ensembles) [109, 248, 415].

6.3 Challenges

The field of current MIA research is divided into various strategies how to further improve performance. Whereas one trend is to modify and enhance architectures like the U-Net, another trend is to utilize ensemble learning techniques [13, 74]. Empirically, ensemble learning based pipelines tend to be superior according to the assumption that the assembling of diverse models has the advantage to combine their strengths in focusing on different features whereas balancing out the individual incapability of a model [338, 340, 419, 420].

Multiple authors provide extensive reviews on general ensemble learning like Ganaiea et al. [338] but only a handful of works started to survey the deep ensemble learning field. While Cao et al. reviewed deep learning based ensemble learning methods specifically in bioinformatics [421], Sagi et al. [339], Ju et al. [340], and Kandel et al. [341] started to provide descriptions or analyses on general deep ensemble learning methods.

However, it is still an open question to what extent as well as which ensemble learning strategies are beneficial in deep learning based MIA pipelines. Even so, the field and idea of general ensemble learning are not novel, the impact of ensemble learning strategies in deep learning based MIC and MIS has not been adequately analyzed in the literature, yet. In order to setup standardized as well as optimal-performing pipelines for MIA, knowledge about application recommendations for different imaging modalities and tasks of deep ensemble learning methods is essential. To achieve such knowledge, experiments to quantify the performance impact of deep ensemble learning are needed. These experiments must include detailed comparisons of different ensemble learning strategies and pooling functions in diverse dataset settings. This would allow enabling meta-analyses for providing comprehensive insights and reliable recommendations.

6.4 Study: Performance Impact in Image Segmentation

In this study, the Author pushed towards analyzing the impact of ensemble learning techniques on MIS performance with deep convolutional neural networks. By computing the performance of two ensemble learning techniques, the Author was able to compare them to a baseline pipeline and, thus, identify possible performance gain. The experiments aim to help understand the beneficial as well as unfavorable influences of different ensemble learning techniques on model performance. This study contributes to the field of deep ensemble learning and provides the missing insights of ensemble learning impact on state-of-the-art MIS pipelines.

6.4.1 Dataset

Since studies showed that CT imaging of the thorax is more sensitive for COVID-19 screening compared to RT-PCR testing and can be used for accurate severity assessment, COVID-19 infection segmentation in CT scans has become an active research field [306]. Due to the morphological variance of lesions, the precise segmentation of infected regions in lungs is a challenging task but of great interest.

For this study, the COVID-19 Lung CT Lesion Segmentation Challenge 2020 dataset was used [422, 423]. The dataset contains 245 thorax CT scans which have publicly available annotations. All samples were retrieved from patients with a positive RT-PCR test for SARS-CoV-2 and originated from China and the USA. An example of a sample is visualized in Figure 6.3. The highlighted ROI in red points to an infected region of the right lung caused by COVID-19. The annotated ground truth segmentation was generated by a two-step process. In the first step, initial segmentation masks were predicted by a model from NVIDIA and the National Institutes of Health, whereas in the second step, these annotations were manually adjudicated and corrected by a board of certified radiologists [422, 423]. Finally, the dataset was sampled into a training subset consisting of 199 samples and testing subset of 46 samples which were also utilized in the proposed ensemble learning study.

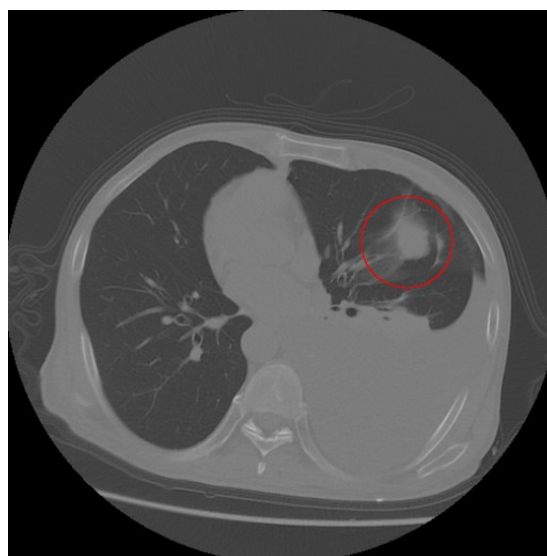


Figure 6.3: Computed tomography scan of the thorax from a Sars-CoV-2 positive patient.

6.4.2 Application

For implementation, the proposed framework MIScnn was utilized. In this subchapter, the general pipeline design and used configurations of MIScnn are described.

The deployed pipeline utilized multiple preprocessing methods to reduce feature complexity. In order to decrease the pixel intensity range as well as to constrain the image representation on relevant tissue types, a clipping of Hounsfield units was applied to a minimum of -1258 HU and a maximum of +255 HU. Afterward, the CT scans were resampled to a uniform voxel spacing of 1.58x1.58x2.70 mm. As the last preprocessing step, the volumes were normalized by Z-Score. For data augmentation, an on-the-fly strategy was applied during the training process with the following augmentation techniques: Scaling, rotations, elastic deformation, mirroring, gaussian noise, and adjustments to brightness, contrast as well as gamma. The resulting volumes were processed into batches according to a patch-wise cropping analysis. This means that the CT scans were sliced into smaller cuboid patches based on a grid for inference, and randomly cropped into a single patch for training in order to introduce another type of data augmentation as well as additionally decrease the overfitting risk. A patch size of 180x180x40 pixels was selected. Furthermore, a patch overlapping strategy of 80x80x40 pixels was integrated for inference.

For the neural network model, a standard U-Net architecture [108] with the Tversky index [241] as loss function was used. The model was trained for a maximum of 1,000 epochs with an Adam [201] optimizer using an initial weight decay of $1E^{-3}$. Multiple callbacks were integrated into the training process: A dynamic learning rate with a reduction factor of 0.1 if no validation loss decrease has been observed for 15 epochs (minimal learning rate was $1E^{-5}$) and the early stopping technique if no validation loss decrease has been observed after 100 epochs. A traditional epoch definition (see Chapter 3.4.1) was utilized with a batch size of 2.

The created pipeline was established as a baseline on which the further ensemble learning techniques were applied: Augmenting and Bagging. For the Augmenting technique, horizontal mirroring was applied to the inference resulting in four predicted segmentation masks per sample. For the Bagging technique, a model ensemble was generated by performing a 5-fold cross-validation on the training subset resulting in five predicted segmentation masks per sample. As pooling function, the Majority Vote Hard method was applied. If the Augmenting resulted in a tie vote, the pixel was labeled as a COVID-19 infection.

6.4.3 Results and Discussion

A pipeline for MIS of COVID-19 infection, which was able to analyze ensemble learning impact on performance, was successfully setup. After the training, the fitted models were utilized to predict segmentation masks highlighting infected regions. The predictions were evaluated in comparison to the ground truth masks annotated by radiologists. For performance assessment, the Dice Similarity Coefficient and the Intersection-over-Union were selected. The performance impact (increase) was defined as the ratio between the mean-averaged DSCs of

the Baseline and the corresponding ensemble learning technique resulting in an intuitive percentage value.

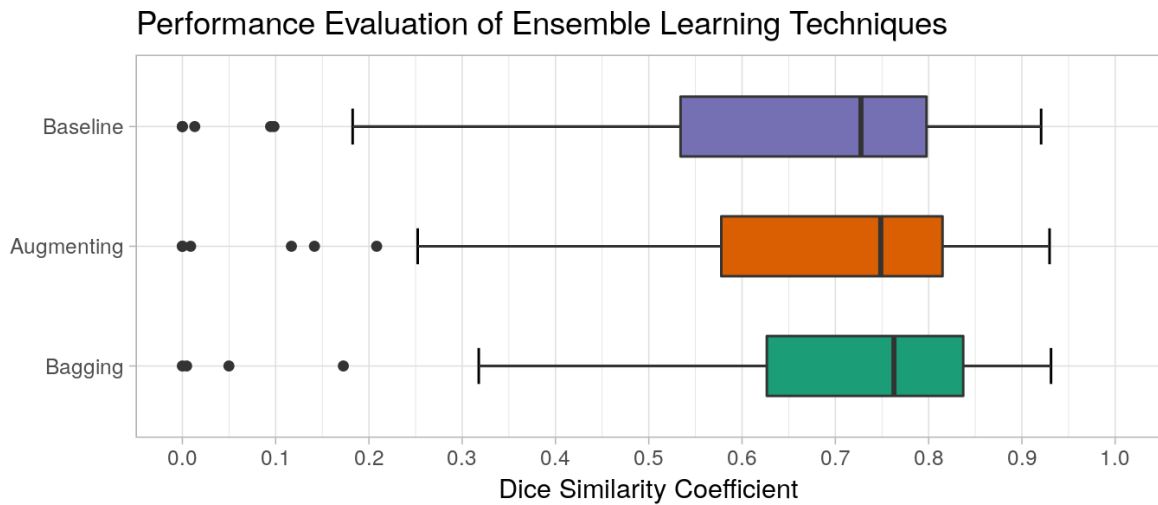


Figure 6.4: Box plot showing the performance distribution of all experiments.

The resulting outcomes revealed that it was possible to predict highly accurate segmentation masks for COVID-19 infected regions. The models achieved the following median-averaged DSC scores on the testing set: 0.728 for the Baseline, 0.749 for the Augmenting, and 0.763 for the Bagging technique. The results of all computed metrics are summarized in Table 6.2. Additionally, the distribution of the achieved DSCs for the testing set is visualized in Figure 6.4. Through the evaluation of the experiments, it was possible to compute performance differences between the Baseline and the ensemble learning techniques by assessing the performance increase. The comparison revealed a positive performance increase for both ensemble learning techniques. Whereas Augmenting showed a small improvement of a performance increase of +2.3%, Bagging achieved an increase of +8.4% compared to the Baseline. A visualization of the performance increase can be seen in Figure 6.5.

Table 6.2: Achieved results of all experiments on ensemble learning based MIS.

Pipeline	Mean DSC	Median DSC	Mean IoU	Median IoU	Performance Increase
Baseline	0.630	0.728	0.502	0.572	X
Augmenting	0.645	0.749	0.520	0.598	2.286
Bagging	0.683	0.763	0.557	0.617	8.352

The performance increase of Augmenting can be explained through the reduction of spatial bias by mirror augmentations of the samples. On the basis of a high prevalence of samples with one-sided feature representation in the training dataset like more patients with infected regions in the right lung, the model learns incorrect spatial patterns. In this case, mirroring the image allows the model to detect these spatial patterns despite the biased fitting. In contrast to Bagging, an ensemble of multiple models trained on different cross-validation folds

demonstrated a stronger performance increase. Through the cross-validation strategy, a more efficient data sampling is possible which ensures that data needed for validation can also be incorporated into the model. Thereby, the model ensemble gains the advantage of additional information.

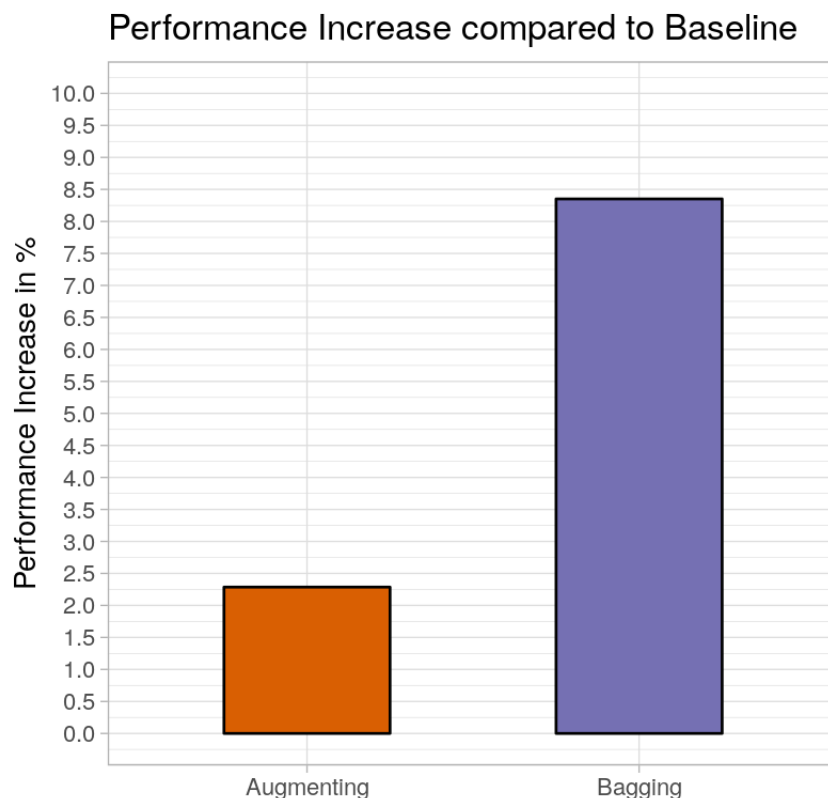


Figure 6.5: Average performance increase by ensemble learning compared to the baseline performance.

In the future, further analysis of ensemble learning impact on segmentation is planned. Next to different ensemble learning methods like Stacking, experimentation with other baseline configurations and datasets are needed to validate the achieved findings.

6.4.4 Conclusions

In order to analyze the performance impact of ensemble learning techniques on medical image segmentation, a comprehensive pipeline for COVID-19 infection segmentation was setup. This allowed the evaluation of the performance increase by the Augmenting and Bagging technique. It was revealed that both ensemble learning techniques improve MIS performance, in which Bagging was able to achieve the strongest performance gain.

6.5 Study: Performance Impact in Image Classification

In this study, the Author pushed towards setup a reproducible analysis pipeline to reveal the impact of ensemble learning techniques on MIC performance with deep convolution neural networks. By computing the performance of multiple ensemble learning techniques, the Author was able to compare them to a baseline pipeline and, thus, identify possible performance gain. Furthermore, the possible performance impact on multiple medical datasets from diverse modalities ranging from histology to X-ray imaging was explored. The experiments aim to help understand the beneficial as well as unfavorable influences of different ensemble learning techniques on model performance. This study contributes to the field of deep ensemble learning and provides the missing overview of state-of-the-art ensemble learning techniques for deep learning based MIC.

6.5.1 Datasets

For increased result reliability and robustness, multiple public MIC datasets were analyzed. The datasets differ in sample size, modality, feature type of interest, and noisiness. The noisiness of a dataset is a subjective impression based on the best-achieved performance in the literature for this dataset. An overview of all datasets can be seen in Table 6.3, as well as exemplary samples in Figure 6.6.

Table 6.3: Overview of utilized datasets with descriptive details and sampling distributions.

Dataset	Modality	Noisiness	Classes	Number of Samples			
				model-train	model-val	ensemble-train	testing
CHMNIST	Histology	Small	8	3,250	501	500	749
COVID	X-ray	Small	3	1,889	291	290	435
ISIC	Dermoscopy	Medium	8	16,466	2,533	2,533	3,799
DRD	Ophthalmoscopy	Strong	5	22,832	3,513	3,513	5,268

CHMNIST

The image analysis of histological slides is an essential part in the field of pathology. The CHMNIST dataset consists of image patches generated from histology slides of patients with colorectal cancer [424, 425]. These patches were annotated in 8 distinct classes: Tumor epithelium, simple stroma (homogeneous composition), complex stroma (containing single tumor cells and/or immune cells), immune cells, debris (including necrosis, hemorrhage, and mucus), normal mucosal glands, adipose tissue and background (no tissue) [424, 425]. The dataset contains in total 5,000 images in RGB color encoding with 625 images for each class and a unified resolution of 150x150 pixels. The slides were generated via an Aperio ScanScope microscope with a 20x magnification from the pathology archive of the University Medical Center Mannheim and Heidelberg University [424].

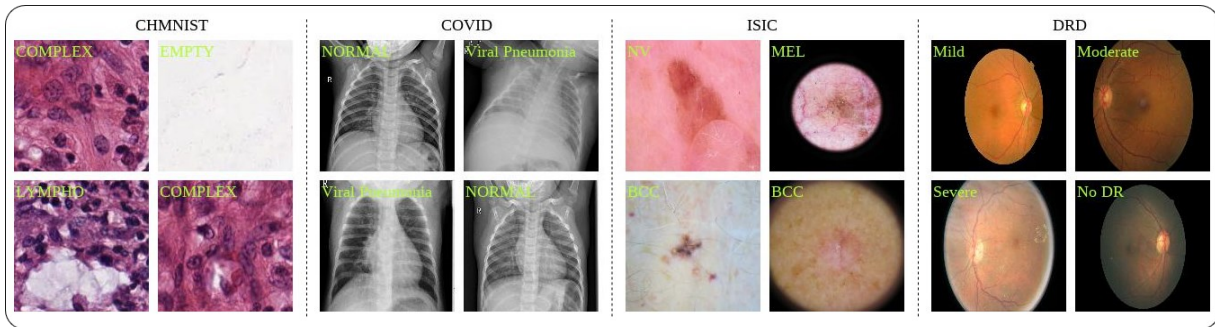


Figure 6.6: Exemplary samples of the four used datasets used in the analysis.

COVID

X-ray imaging is one of the key modalities in the field of medical image analysis and is crucial in modern healthcare. Furthermore, X-ray imaging is a widely favored alternative to reverse transcription polymerase chain reaction testing for the coronavirus disease (COVID-19) [426, 427]. Researchers from Qatar, Doha, Dhaka, Bangladesh, Pakistan, and Malaysia have created a dataset of thorax X-ray images for COVID-19 positives cases along with healthy control and other viral pneumonia cases [426]. The X-ray scans were gathered and annotated from 6 different radiographic databases or sources like the Italian Society of Medical and Interventional Radiology (SIRM) COVID-19 Database [322, 426]. The dataset consists of in total 2,905 grayscale images with 219 COVID-19 positives, 1,345 viral pneumonia, and 1,341 control cases.

ISIC

Melanoma, appearing as pigmented lesions on the skin, is a major public health problem with more than new 300,000 cases per year and is responsible for the majority of skin cancer deaths [285]. Dermoscopy is the field of early melanoma detection, which can be either performed manually by expert visual inspection or automatically by MIC via high-resolution cameras. The International Skin Imaging Collaboration (ISIC) hosts the largest publicly available collection of quality-controlled images of skin lesions [285]. The 2019 release of their archive consists of in total 25,331 RGB images which were classified into the following 8 classes: Melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), vascular lesion (VASC) and squamous cell carcinoma (SCC) [57–59].

DRD

Diabetic retinopathy is the leading cause of blindness and is estimated to affect over 93 million people worldwide [428]. The detection of diabetic retinopathy is mostly done via a time-consuming manual inspection by a clinician or ophthalmologist with the help of a fundus camera [159]. In order to contribute to research for automated diabetic retinopathy detection (DRD) algorithms, the California Healthcare Foundation and EyePACS created a public dataset consisting of 35,126 RGB fundus images [428, 429]. These were annotated in the following

five classes according to disease severity: No DR, Mild, Moderate, Severe, and Proliferative DR. It has to be noted that the authors pointed out the real-world aspect of this dataset which includes various types of noise-like artifacts, out of focus, under-/overexposed images and incorrect annotations [428].

6.5.2 Application

For implementation, the proposed framework AUCMEDI was utilized which is built on TensorFlow [72]. In the following subchapters, the configurations of AUCMEDI and the general pipeline are described. A configuration overview of the applied preprocessing techniques and neural network models is summarized in Table 6.4.

Table 6.4: Configuration overview of implemented MIC pipelines.

Preprocessing			
Online Image Augmentation		Subfunctions	
Flipping	Random Probability: 50%	Padding	To square ratio
Rotations	Random 90° Probability: 50%	Resize	To 224 ² or 299 ² or 380 ² (depending on architecture)
Brightness, Contrast, Saturation, and Hue	-0.1 to +0.1 factor range Probability: 50%	Standardize	Z-Score normalization
Neural Network - General			
Loss	Focal Loss	Class Weights	Computed on train set via $n_samples / (n_classes * \text{bincount}(y))$
Activation Output	Softmax	Optimizer	Adam
Learning Rate	Initialized at 1e-04 for frozen-layer epochs Initialized at 1e-05 for unfrozen-layer epochs	Dynamic Learning Rate	Decreasing up to 1e-07 by a factor of 0.1 each time after 8 epochs without validation loss improvement
Epochs	Max. 1,000	Batch size	28
Transfer Learning	ImageNet	Number of frozen Epochs	10
Model Checkpoints	Best computed loss on validation set (model-val) during training	Early Stopping	After 15 epochs without validation loss improvement
Training Monitoring	Tensorboard and CSV dumps for logging		
Neural Network - Architecture dependent Input Size			
DenseNet121	244 x 244 x 3	MobileNetV2	244 x 244 x 3
ResNeXt101	244 x 244 x 3	ResNet101	244 x 244 x 3
VGG16	244 x 244 x 3	EfficientNetB4	380 x 380 x 3
InceptionResNetV2	299 x 299 x 3	Xception	299 x 299 x 3
Vanilla	244 x 244 x 3		

Sampling and Preprocessing

In order to ensure a reliable evaluation of the models, each dataset was sampled with the following distribution strategy: For model training, 65% of each dataset was used (called ‘model-train’) whereas 10% of all samples were used as a validation set during the training process (called ‘model-val’) to allow validation monitoring for callback strategies. The only

exception for this ‘model-train’ and ‘model-val’ sampling strategy occurred in the Bagging experiment, in which the two sets were combined and sampled according to a 5-fold cross-validation (75% in total of a dataset with 60% as training and 15% as validation for each fold). For possible training of ensemble learning pooling methods, another 10% of the total dataset was reserved (called ‘ensemble-train’). For the final in detail evaluation on a separate hold-out set, the remaining 15% of each dataset was sampled as testing set (called ‘testing’).

The following preprocessing methods were applied for enhancement of the pattern-finding process of the deep learning models as well as to increase data variability. The implemented pipeline utilized extensive real-time (also called on-the-fly) image augmentation during the training phase to allow the model seeing novel and unique images in each epoch. The augmentation consisted of the following techniques: Flipping, rotations as well as alterations in brightness, contrast, saturation, and hue. Furthermore, all images were squared padded for avoiding aspect ratio loss. In the posterior resizing, the image resolutions were reduced to the model architecture default input sizes, which were commonly 224x224 pixels except for EfficientNetB4 with 380x380 pixels, as well as InceptionResNetV2 and Xception with 299x299 pixels [79, 87, 97]. Before passing the images into the model, value intensity normalization was applied. The intensities were zero-centered via Z-Score normalization based on the mean and standard deviation computed on the ImageNet dataset [76].

Neural Network Models

For computer vision tasks like image classification, deep convolutional neural networks are state-of-the-art and unmatched in accuracy and robustness [11, 30, 263, 398]. Rather than focusing on a single model architecture for the analysis, diverse classification architectures were trained to ensure result reliability. The following architectures were selected: DenseNet121 [90], EfficientNetB4 [79], InceptionResNetV2 [87], MobileNetV2 [92], ResNeXt101 [88], ResNet101 [80], VGG16 [84], Xception [97] and a custom Vanilla architecture for comparison. The Vanilla architecture consisted of 4 convolutional layers with each followed by a max-pooling layer. The utilized classification head for all architectures applied a global average pooling, a dense layer with linear activation, a dropout layer, and another dense layer with a softmax activation function for the final class probabilities. The selected architectures represent the large diversity of popular and widely applied types of deep learning models for image classification. These strongly vary in the number of model parameters as well as neural network layers, input sizes, underlying composition techniques as well as functionality principles, and overall complexity. This allows a clearer analysis of the ensemble learning impact without architecture-related biases. In terms of general complexity, the utilized architectures have the following numbers of model parameters: Vanilla with $0.5E^6$, DenseNet121 with $7.0E^6$, EfficientNetB4 with $17.7E^6$, InceptionResNetV2 with $54.3E^6$, MobileNetV2 with $2.3E^6$, ResNet101 with $42.7E^6$, ResNeXt101 with $42.2E^6$, VGG16 with $14.7E^6$, and Xception with $20.8E^6$. Further details on the architectures and their differences can be found in the excellent reviews of Bressen et al. [329] and Alzubaidi et al. [74].

A transfer learning strategy was utilized by pretraining all models on the ImageNet dataset [76]. For the fitting process, the architecture layers were frozen at first except for the classification head and unfrozen, again, for fine-tuning. Whereas the frozen transfer learning phase was performed for 10 epochs using the Adam optimization with an initial learning rate of $1E^{-4}$, the fine-tuning phase stopped after a maximal training time of 1,000 epochs (including the 10 epochs for transfer learning). The fine-tuning phase also utilized a dynamic learning rate for the Adam optimization [201] starting from $1E^{-5}$ to a maximum decrease to $1E^{-7}$ by a decreasing factor of 0.1 after 8 epochs without improvement on the monitored validation loss. As loss function for model training, the weighted Focal loss from Lin et al. [213] was used. The formula of the weighted Focal loss is defined in Chapter 5.4.2, in Equation (5.1). The class weights were computed based on the class distribution in the corresponding ‘model-train’ sampling set. Furthermore, an early stopping and model checkpoint technique was applied for the fine-tuning phases, stopping after 15 epochs without improvement and saving the best model measured based on validation loss monitoring.

The complete analysis was performed with a batch size of 28 and run parallelized on a workstation with 4x NVIDIA Titan RTX with each 24GB VRAM, Intel(R) Xeon(R) Gold 5220R CPU @ 2.20GHz with 96 cores and 384GB RAM.

Ensemble Learning Techniques

In this analysis, the performance impact of the following ensemble learning techniques was explored: Augmenting, Bagging, and Stacking. For comparison, Baseline models were setup for all architectures to identify possible performance gain or loss tendencies through the ensemble learning techniques. For Augmenting, the Baseline models were reused and random rotations as well as mirroring on all axes were applied to images for inference. For each sample, 15 randomly augmented images were created, and their predictions were combined through an unweighted Mean as pooling function. For Stacking, the Baseline models consisting of various architectures were reused as an ensemble in order to stack the pooling functions directly on top of these inhomogeneous models. For Bagging, a 5-fold cross-validation was applied as described in the previous subchapter Sampling and Preprocessing, which resulted in five models for each architecture. The predictions of these five models for a single sample were combined via multiple pooling functions.

The following pooling functions, which were applied for the Stacking and Bagging technique, were analyzed: Best Model (according to F1-score), Decision Tree trained with Gini impurity as information gain function [430], Gaussian Process classifier based on Laplace approximation with a ‘one-vs-rest’ multi-class strategy, Global Argmax, Logistic Regression using the ‘newton-cg’ solver and L2 regularization with a multinomial multi-class strategy [377], Majority Vote Soft and Hard, Unweighted and Weighted Mean (weights according to F1-score), Naïve Bayes implemented as the Complement variant described by Rennie et al. [431], Support Vector Machine based on the standard implementation from LIBSVM [418], and k-Nearest Neighbors initialized with five neighbors. For the Augmenting technique, only the Unweighted

Mean was used as pooling function. Pooling functions that required prior fitting were trained on the ‘ensemble-train’ sampling set.

Evaluation

For evaluation, the packages pandas [432], scikit-learn [342], and plotnine [433] for visualization were utilized.

The performance scores were calculated class-wise and averaged by the unweighted mean. The following community-standard scores were used: Accuracy, F1-score (also called DSC), Sensitivity, False Positive Rate, and area under the receiver operating characteristic curve (AUC & ROC). The metrics are defined and described in Chapter 3.5.1.

Code Reproducibility

The code for this study was implemented in Python (platform independent) and is available under the GPL-3.0 License at the following GitHub repository: <https://github.com/frankkramer-lab/ensmic>.

All data generated and analyzed during this study is available in the following Zenodo repository: <https://doi.org/10.5281/zenodo.6457912>.

6.5.3 Results

The total training time of the complete analysis took around 1,215 hours with the following distribution per technique: Baseline 213 hours (17.5%), Augmenting 0.00 hours (0%), Stacking less than 0.09 hours (0%), and Bagging 1,002 hours (82.5%). It has to be noted that the Augmenting and Stacking techniques were based on the Baseline models but did not require extensive additional training time. The Baseline revealed an average training time by mean across all architectures of 45 minutes for COVID, 47 minutes for CHMNIST, 302 minutes for ISIC, and 1,026 minutes for DRD, whereas the Vanilla architecture had the lowest training time on average across datasets with 246 minutes and ResNet101 the highest with 522 minutes. Further details on training times for all architectures and phases can be found in the supplementary.

All training processes for the deep learning convolutional neural network models did not require the entire 1,000 epochs and instead were early stopped after an average of 51 epochs. On median the epoch distribution looked like the following: For Baseline CHMNIST 54, COVID 48, ISIC 64, and DRD 37. For Bagging CHMNIST 53, COVID 48, ISIC 68, and DRD 37. Through validation monitoring during the training, no overfitting was observed. The training and validation loss function revealed no significant distinction from each other.

Baseline

The Baseline revealed the performance of various state-of-the-art architectures without the usage of any ensemble learning technique. This resulted in an average F1-score by a median of 0.95 for CHMNIST, 0.96 for COVID, 0.72 for ISIC, and 0.43 for DRD. The architectures shared an overall similar performance depending on the dataset noisiness. According to their F1-score, the best architectures were EfficientNetB4 and ResNet101 in CHMNIST, ResNeXt101 in COVID, ResNet101 and ResNeXt101 in ISIC, as well as EfficientNetB4 and ResNet101 in DRD. The smaller architectures like Vanilla and MobileNetV2 performed the worst. More details are shown in Table 6.5.

Table 6.5: Achieved results of the Baseline approach grouped by architecture and dataset.

Method	CHMNIST				COVID				ISIC				DRD			
	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC
DenseNet121	0.99	0.95	0.95	1.0	0.98	0.97	0.97	1.0	0.95	0.72	0.76	0.96	0.84	0.43	0.48	0.78
EfficientNetB4	0.99	0.96	0.96	1.0	0.97	0.95	0.96	0.99	0.95	0.74	0.79	0.97	0.84	0.45	0.53	0.81
Inception-ResNetV2	0.99	0.95	0.95	1.0	0.98	0.96	0.96	1.0	0.95	0.72	0.74	0.96	0.84	0.42	0.5	0.78
MobileNetV2	0.98	0.9	0.9	0.99	0.97	0.95	0.96	0.99	0.94	0.68	0.73	0.95	0.82	0.32	0.41	0.72
ResNeXt101	0.99	0.95	0.95	1.0	0.99	0.99	0.98	1.0	0.96	0.76	0.78	0.97	0.83	0.43	0.52	0.79
ResNet101	0.99	0.96	0.96	1.0	0.98	0.98	0.98	1.0	0.95	0.76	0.79	0.97	0.84	0.45	0.48	0.79
VGG16	0.99	0.95	0.95	1.0	0.98	0.98	0.97	1.0	0.94	0.71	0.74	0.96	0.82	0.43	0.49	0.78
Vanilla	0.95	0.82	0.82	0.98	0.91	0.79	0.79	0.94	0.86	0.27	0.34	0.8	0.74	0.16	0.27	0.59
Xception	0.98	0.94	0.94	1.0	0.97	0.96	0.95	1.0	0.95	0.74	0.77	0.96	0.82	0.4	0.48	0.78

The performance results are also visualized in Figure 6.7. The left side of the figure shows the achieved F1-score with its confidence interval on classification for each architecture, whereas the right side shows class-wise ROC curves for the best performing architecture (according to F1-score) on each dataset. The ROC curves revealed only marginal performance differences of classes in the CHMNIST, COVID, and ISIC dataset. However, DRD showed significant differences in Accuracy between classes whereas the detection of ‘Mild’ samples had the lowest performance.

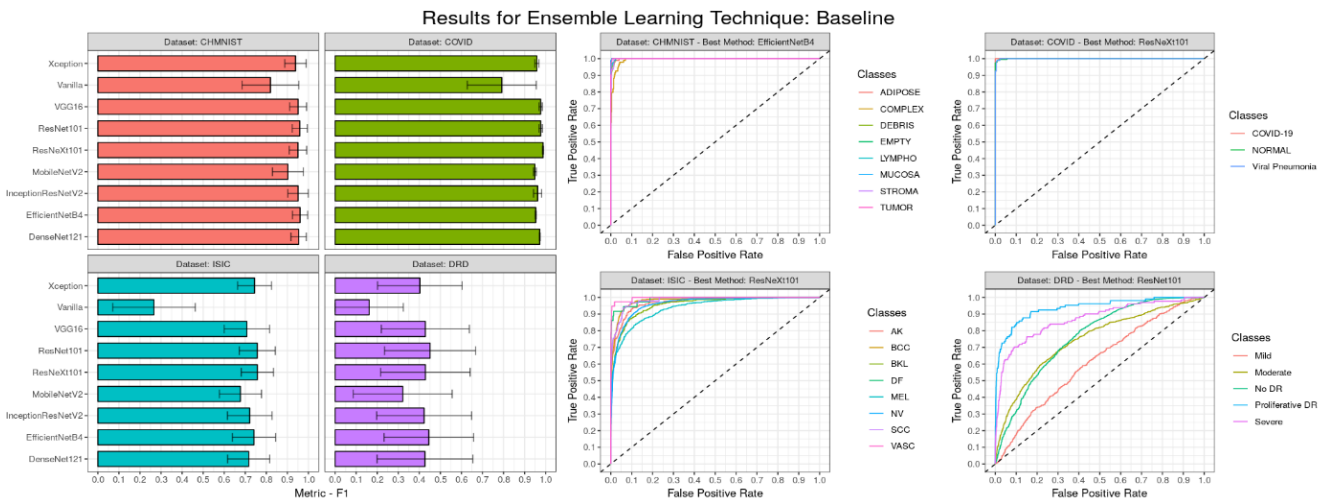


Figure 6.7: Visualization of performance results for the Baseline pipeline.

Augmenting

By integrating the ensemble learning technique Augmenting for inference based on the Baseline models, it was possible to obtain the following average F1-scores by median: 0.95 for CHMNIST, 0.97 for COVID, 0.74 for ISIC, and 0.43 for DRD. More details are shown in Table 6.6. Thus, there was only a marginal performance increase for the CHMNIST and ISIC datasets compared to the Baseline.

Table 6.6: Achieved results of the Augmenting approach grouped by architecture and dataset.

Method	CHMNIST				COVID				ISIC				DRD			
	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC
DenseNet121	0.99	0.95	0.95	1.0	0.99	0.98	0.98	1.0	0.95	0.73	0.77	0.97	0.85	0.43	0.5	0.79
EfficientNetB4	0.99	0.96	0.96	1.0	0.97	0.95	0.95	0.99	0.96	0.77	0.81	0.97	0.85	0.45	0.55	0.82
Inception-ResNetV2	0.99	0.95	0.95	1.0	0.98	0.97	0.97	1.0	0.95	0.74	0.76	0.97	0.85	0.44	0.52	0.79
MobileNetV2	0.98	0.92	0.92	0.99	0.97	0.96	0.96	0.99	0.94	0.71	0.76	0.96	0.84	0.34	0.42	0.74
ResNeXt101	0.99	0.95	0.95	1.0	0.99	0.98	0.98	1.0	0.96	0.77	0.8	0.97	0.84	0.43	0.52	0.8
ResNet101	0.99	0.95	0.95	1.0	0.98	0.98	0.98	1.0	0.95	0.77	0.8	0.97	0.85	0.47	0.51	0.8
VGG16	0.99	0.95	0.95	1.0	0.98	0.98	0.97	1.0	0.95	0.72	0.75	0.96	0.83	0.45	0.51	0.79
Vanilla	0.95	0.82	0.82	0.98	0.9	0.77	0.8	0.93	0.86	0.27	0.35	0.8	0.74	0.16	0.26	0.59
Xception	0.99	0.95	0.95	1.0	0.98	0.97	0.97	1.0	0.95	0.76	0.77	0.97	0.83	0.42	0.5	0.8

However, in the comparison of the best possible score between Augmenting and Baseline, a performance impact of 0% for CHMNIST, -1% for COVID, +3% for ISIC, and +4% for DRD was measured according to the F1-score.



Figure 6.8: Visualization of performance results for the Augmenting pipeline.

The performance results are visualized in Figure 6.8 in an equal manner to Figure 6.7. The ranking of best-performing architectures revealed no drastic change. Especially, the EfficientNetB4 and ResNet101 achieved the highest performance similar to the Baseline, as well as the smaller architectures like Vanilla and MobileNetV2 the lowest. The ROC curves resulted in equivalent model Accuracy variance between classes and datasets as the Baseline.

Stacking

For the Stacking technique, several pooling functions were successfully applied for combining the predictions of all Baseline architectures and resulted in the following average F1-scores by median: 0.96 for CHMNIST, 0.98 for COVID, 0.81 for ISIC, and 0.48 for DRD. More details are shown in Table 6.7. Compared with the median F1-score of the Baseline, a performance impact of +1% for CHMNIST, +2% for COVID, +13% for ISIC, and +12% for DRD was measured. Additional to the median performance comparison, the pooling function ‘Best Model’ was also used as a benchmark without the usage of ensemble learning, which was inferior by up to 0.08 in Accuracy, 0.06 in F1, 0.06 in Sensitivity, and 0.04 in AUC compared with the best pooling function.

Table 6.7: Achieved results of the Stacking approach grouped by architecture and dataset.

Method	CHMNIST				COVID				ISIC				DRD			
	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC
Best Model	0.99	0.96	0.96	1.0	0.98	0.97	0.97	1.0	0.96	0.76	0.78	0.97	0.84	0.45	0.48	0.79
Decision Tree	0.98	0.94	0.94	0.97	0.98	0.96	0.97	0.98	0.94	0.67	0.69	0.82	0.87	0.41	0.42	0.64
Gaussian Process	0.99	0.96	0.96	0.99	0.99	0.98	0.98	1.0	0.96	0.82	0.79	0.97	0.92	0.48	0.44	0.83
Global Argmax	0.99	0.96	0.96	1.0	0.99	0.98	0.98	1.0	0.96	0.79	0.82	0.94	0.85	0.44	0.54	0.71
Logistic Regression	0.99	0.96	0.96	1.0	0.99	0.98	0.98	1.0	0.96	0.81	0.79	0.98	0.92	0.48	0.45	0.83
Majority Voting Hard	0.99	0.95	0.95	0.99	0.99	0.98	0.98	1.0	0.96	0.8	0.81	0.96	0.87	0.48	0.52	0.79
Majority Voting Soft	0.99	0.96	0.96	1.0	0.99	0.99	0.98	1.0	0.96	0.82	0.82	0.97	0.87	0.49	0.53	0.81
Mean Unweighted	0.99	0.96	0.96	1.0	0.99	0.99	0.98	1.0	0.96	0.82	0.82	0.98	0.87	0.49	0.53	0.82
Mean Weighted	0.99	0.96	0.96	1.0	0.99	0.99	0.98	1.0	0.96	0.82	0.82	0.98	0.87	0.49	0.53	0.82
Naive Bayes	0.99	0.97	0.97	0.99	0.99	0.99	0.98	1.0	0.96	0.81	0.81	0.97	0.9	0.36	0.41	0.79
Support Vector Machine	0.99	0.96	0.95	1.0	0.99	0.98	0.98	1.0	0.96	0.82	0.8	0.97	0.92	0.47	0.42	0.8
k-Nearest Neighbor	0.99	0.96	0.96	0.99	0.99	0.99	0.98	0.99	0.96	0.81	0.79	0.93	0.91	0.43	0.4	0.74

According to their F1-score, the best pooling functions were Naïve Bayes in CHMNIST, Majority Voting Soft, Mean Un-/Weighted, Naïve Byes and k-Nearest Neighbor in COVID, Gaussian Process, Majority Voting Soft, Mean Un-/Weighted, and Support Vector Machine in ISIC, as well as Majority Voting Soft and Mean Un-/Weighted in DRD. The Decision Tree

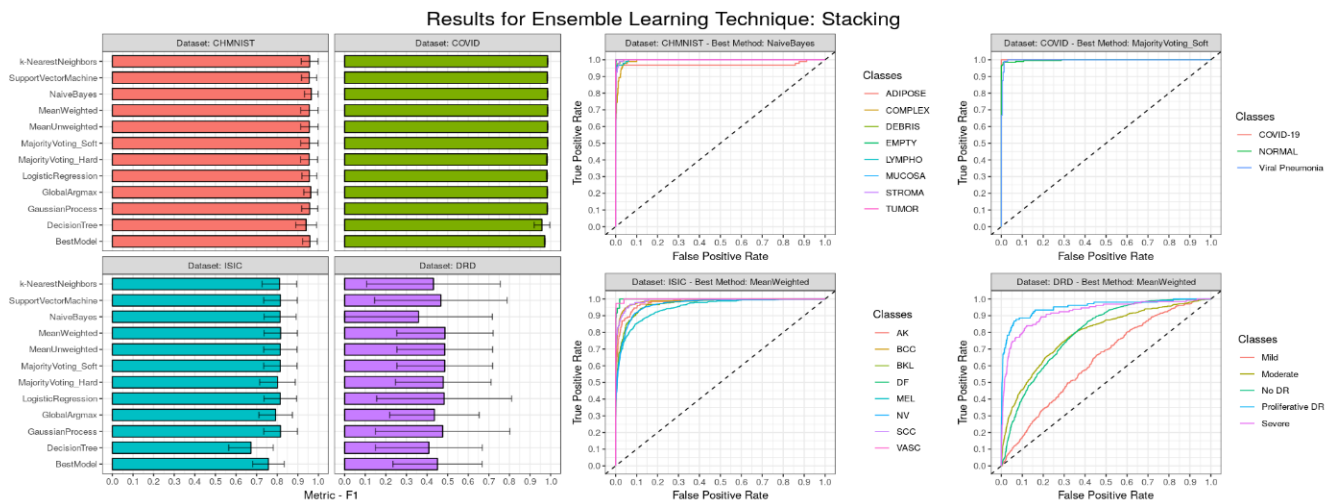


Figure 6.9: Visualization of performance results for the Stacking pipeline.

pooling function performed the worst and had a performance difference of up to -0.02 Accuracy, -0.09 F1, -0.09 Sensitivity, and -0.15 AUC compared to the ‘Best Model’ from the Baseline.

The performance results are visualized in Figure 6.9 in an equal manner to previous figures. The ROC curves of the Stacking approach showed the same trend of class-wise performance differences as the Baseline, but with better precision results especially in the ISIC and DRD dataset.

Bagging

By training new models based on a 5-fold cross-validation, it was possible to analyze the effects of Bagging on prediction capability. The predictions of five models per architecture were combined using various pooling functions. In this experiment, the five models of the EfficientNetB4 architecture archived the highest F1-scoring and were selected for further result reporting and representation of the Bagging approach. The evaluation of the merged predictions of these models showed the following averaged F1-score results by median: 0.96 for CHMNIST, 0.98 for COVID, 0.8 for ISIC, and 0.47 for DRD. In comparison with the Baseline, the following performance impact was measured: +1% for CHMNIST, +2% for COVID, +11% for ISIC, and +9% for DRD. More details for the Bagging results can be seen in Table 6.8.

Table 6.8: Achieved results of the Bagging approach grouped by architecture and dataset.

Method	CHMNIST				COVID				ISIC				DRD			
	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC	Acc	F1	Sens	AUC
Best Model	0.99	0.96	0.96	1.0	0.99	0.98	0.98	1.0	0.95	0.76	0.78	0.97	0.86	0.46	0.53	0.82
Decision Tree	0.98	0.92	0.93	0.96	0.97	0.96	0.95	0.96	0.94	0.68	0.66	0.81	0.86	0.4	0.41	0.63
Gaussian Process	0.99	0.96	0.96	0.99	0.99	0.98	0.98	1.0	0.96	0.81	0.8	0.97	0.92	0.47	0.46	0.83
Global Argmax	0.99	0.96	0.96	0.99	0.99	0.98	0.98	1.0	0.96	0.78	0.82	0.95	0.84	0.45	0.56	0.72
Logistic Regression	0.99	0.96	0.96	0.99	0.99	0.98	0.98	1.0	0.96	0.81	0.8	0.97	0.92	0.47	0.46	0.84
Majority Voting Hard	0.99	0.96	0.96	0.98	0.99	0.98	0.98	0.99	0.96	0.8	0.81	0.94	0.85	0.48	0.55	0.77
Majority Voting Soft	0.99	0.96	0.96	1.0	0.99	0.98	0.98	1.0	0.96	0.8	0.82	0.97	0.85	0.48	0.56	0.81
Mean Unweighted	0.99	0.96	0.96	1.0	0.99	0.98	0.98	1.0	0.96	0.8	0.82	0.98	0.85	0.48	0.56	0.83
Mean Weighted	0.99	0.96	0.96	1.0	0.99	0.98	0.98	1.0	0.96	0.8	0.82	0.98	0.85	0.48	0.56	0.83
Naive Bayes	0.99	0.96	0.96	0.99	0.99	0.98	0.98	1.0	0.96	0.8	0.82	0.97	0.9	0.37	0.43	0.8
Support Vector Machine	0.99	0.96	0.96	0.99	0.99	0.98	0.98	1.0	0.96	0.79	0.76	0.95	0.92	0.46	0.43	0.79
k-Nearest Neighbor	0.99	0.96	0.95	0.99	0.99	0.98	0.98	0.99	0.96	0.79	0.78	0.92	0.91	0.44	0.43	0.74

Contrary to the previous ensemble learning approaches, the ‘Best Model’ pooling function represents not the best validation scoring Baseline model but instead the best model from the 5-fold cross-validation. The ranking between best-performing pooling functions for the EfficientNetB4 5-fold cross-validation revealed close grouping around the same score. In the CHMNIST and COVID sets, all pooling functions except for Decision Trees achieved an F1-score of 0.96 and 0.98, respectively. Overall, the pooling based on Mean, Majority Voting, Gaussian Process, and Logistic Regression resulted in the highest performance on average. On the other hand, Decision Tree and Naïve Bayes obtained the lowest F1-scores.

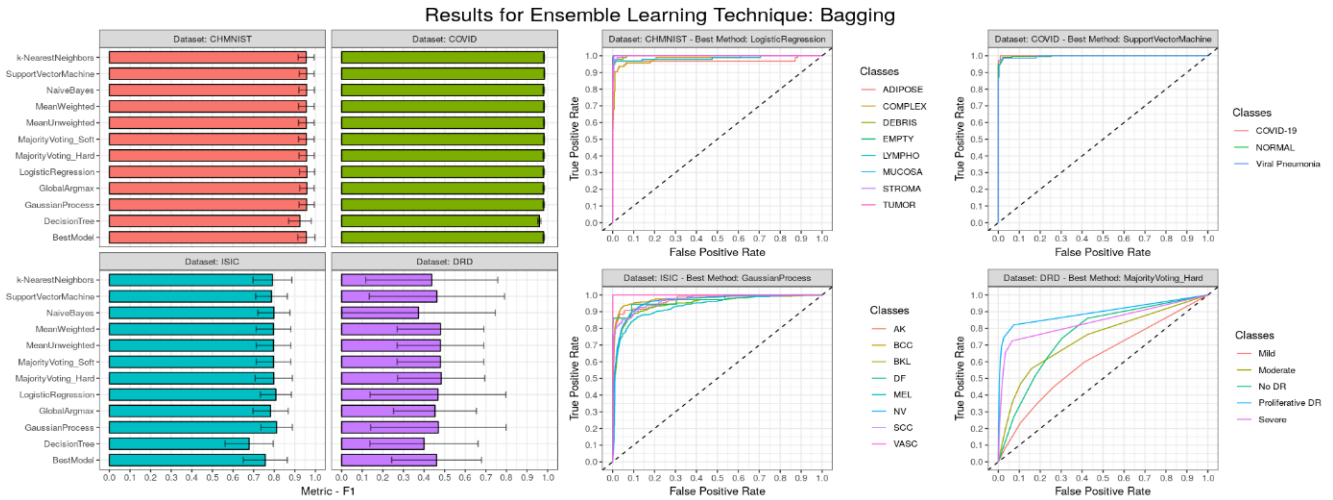


Figure 6.10: Visualization of performance results for the Bagging pipeline.

The performance results are visualized in Figure 6.10 in an equal manner to previous figures. The ROC curves indicate an overall equal or superior performance compared to the Baseline, but a slightly inferior performance in CHMNIST dataset. Notably, the ISIC dataset indicates a stronger model Accuracy variance between classes compared to the Baseline ROC curves.

6.5.4 Discussion

In this study, a reproducible pipeline for analyzing the impact of ensemble learning techniques on MIC performance with deep convolutional neural networks was setup. Augmenting, Bagging as well as Stacking were implemented and compared to a Baseline to compute performance gain on various metrics like F1-score, Sensitivity, AUC, and Accuracy. The analysis proved that the integration of ensemble learning techniques can significantly boost classification performance from deep convolutional neural network models. As summarized in Figure 6.11, the results showed a performance gain ranking from highest to lowest for the following ensemble learning techniques: Stacking, Bagging, and Augmenting. The left side of the figure illustrates bar plots showing the maximum achieved Accuracy across all methods for each ensemble learning technique and dataset: Baseline (red), Augmenting (blue), Bagging (green), and Stacking (purple). Additionally, the distribution of achieved F1-scores by the various methods is illustrated with box plots. The right side of the figure illustrates the computed performance impact between the best scoring method of the Baseline and the best scoring method of the applied ensemble learning technique for each dataset. The performance impact is represented as performance gain in % between the F1-score (top-right side of the figure) as well as Accuracy (bottom-right side of the figure). The color mapping of the ensemble learning techniques is equal to the bar plots on the left side of the figure (Augmenting: Blue; Bagging: Green; Stacking: Purple).

Stacking

The ensemble learning technique with the highest performance gain was Stacking, which applies pooling functions on top of different deep convolutional neural network architectures.

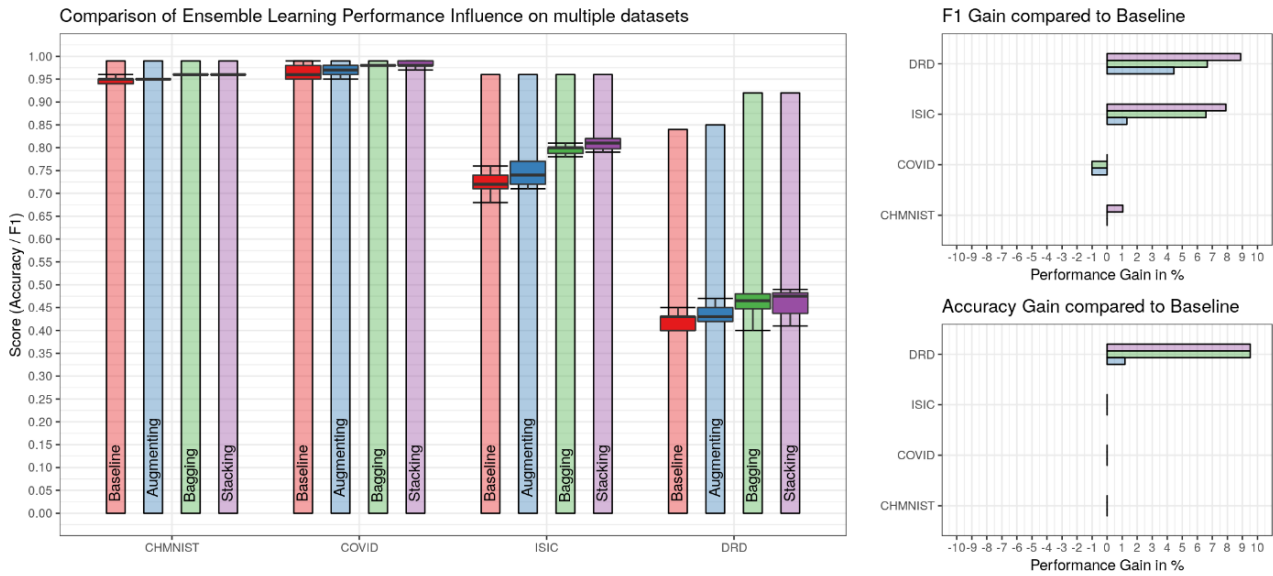


Figure 6.11: Summary of all experiments on performance impact of ensemble learning techniques in MIC.

Various state-of-the-art MIC pipelines heavily utilize a Stacking based pipeline structure to optimize performance by combining novel architectures or differently trained models [159, 257, 400, 403, 404]. This results in higher inference quality and bias or error reduction by using the prediction information of diverse methods. The performed analysis also revealed that, according to F1-score results, simple pooling functions like averaging by Mean or a Soft Majority Vote result in an equally strong or even higher performance gain compared to more complex pooling functions like Support Vector Machines or Logistic Regressions. However, according to Accuracy results, the more complex pooling functions obtained higher scores. This indicates that more simple pooling functions are still based on the penalty strategy of the models which were trained with a class weighted loss function in the performed experiments. Thus, the results of simple pooling functions still optimize for class-balanced metrics like F1-score or Sensitivity. On the other hand, more complex pooling functions with a separate training process focused on optimizing overall true cases including true negatives which resulted in better scoring on unbalanced metrics like Accuracy. Apart from that, other recent studies which analyzed the impact of Stacking also support the proposed hypothesis that Stacking can significantly improve individual deep convolutional neural network model performance by up to 10% [189, 338, 340, 341]. With a similar experimental design as the proposed study, Kandel et al. demonstrated Stacking impact on a musculoskeletal fracture dataset and analyzing pooling functions based on statistics as well as probability [341].

Augmenting

The Augmenting technique was demonstrated to be an efficient ensemble learning approach. In nearly all performed experiments, it was possible to improve the performance by another few percent through reducing overfitting bias in predictions. In theory, this should be already avoided with standard data augmentation during the training process. Although, the experiments indicated that the increased image variability through Augmenting could lead to

adverse performance influences if applied to models based on small-sized datasets with a high risk of being overfitted. Especially in medical imaging, in which small datasets are common, this effect should be considered if Augmenting is applied and can also act as a strong indicator for overfitting. Nevertheless, strong performing MIC pipelines revealed that model performance can be significantly boosted with inference Augmenting [410–413]. Recent studies by Kandel et al. [412] and Shanmugam et al. [410] also analyzed the performance impact in detail of Augmenting on MIC and proved strong as well as consistent improvement, especially for low-scoring models. In contrast to other ensemble learning techniques, Augmenting can be quickly integrated into pipelines without the need for additional training of various deep convolutional neural network or machine learning models. Thus, also a single model pipeline can benefit from this ensemble learning technique. However, the performance gain from Augmenting is strongly influenced by applied augmentation methods and medical context in a dataset. Molchanov et al. [413] tried to solve this issue with a greedy policy search to find the optimal Augmentation configuration. A limitation of the Author's performed analysis was that Augmenting was implemented and studied utilizing only an unweighted Mean as pooling function. However, other simple pooling functions without model training requirements like Global Argmax or Majority Vote Soft/Hard could have led to a stronger performance increase by Augmenting and should be analyzed as future work.

Bagging

Nowadays, Bagging is one of the most widely used ensemble learning techniques and is utilized in several state-of-the-art pipelines and top-performing benchmark submissions in MIC [159, 187, 338, 340, 405, 434]. In compliance with the performed experiments, Bagging showed a strong performance increase for large datasets and no or marginal performance decrease for small datasets. Similar to Stacking, Bagging was able to significantly improve prediction capability for complex datasets like ISIC and DRD. The Author interpreted possible detrimental effects in COVID and CHMNIST that the fewer data used for model training through cross-validation sampling had a considerable impact on performance in smaller datasets. Especially in small medical datasets with rare and unique morphological cases, excluding these can have a strong negative impact on performance. This is why large datasets like ISIC and DRD with adequate feature presentations in all sampled folds revealed persistent performance improvement. Studies like Dwork et al. [435] analyzed this behavior and concluded that cross-validation based strategies comprise sustainable overfitting risk [436]. Based on the Author's results, Bagging showed to have a high risk of drifting away from an optimal bias-variance tradeoff. According to Geman et al. [437], the bias-variance tradeoff is the right balance between bias and variance in a machine learning model in order to obtain the optimal generalizable model [437]. Whereas increased bias results in the risk of underfitting, increased variance can lead to overfitting. Cross-validation based Bagging boosts efficient data usage and, thus, the variance of a model. However, it has to be noted that the bias-variance tradeoff is still in active discussion in the research community for its correctness in deep learning [438, 439]. Furthermore, Bagging requires extensive additional training time to obtain multiple models. In the field of deep learning, training a higher number of models can lead to an

extremely time-consuming process. For this reason, the performed analysis was specified on a 5-fold cross-validation. Still, the analysis results for Bagging are limited thereby based on the specification on using only 5-folds. Further research is needed on the impact of fold number or sampling size on performance and model generalizability in deep learning based MIC. Nevertheless, it can be concluded that Bagging is a powerful but complex to utilize ensemble learning technique and that its effectiveness is highly depended on sufficient feature representation in the sampled cross-validation folds. To avoid harmful folds with missing feature representation, in-detail dataset analysis is promoted by the Author with manual annotation supported sampling (stratified) or using a higher k-fold to increase training sets and, thus, reduce the risk of excluding samples with unique morphological features.

6.5.5 Conclusions

In this study, the impact of the most widely used ensemble learning techniques was analyzed on medical image classification performance: Augmenting, Stacking, and Bagging. A reproducible experiment pipeline was setup, the performance evaluated through multiple metrics, and the ensemble learning techniques compared with a Baseline to identify possible performance gain. The results revealed that Stacking was able to achieve the largest performance gain in the proposed medical image classification pipeline. Augmenting showed consistent improvement capabilities on non-overfitting models and has the advantage to be applicable to also single model based pipelines. Cross-validation based Bagging demonstrated significant performance gain close to Stacking, but reliant on sampling with sufficient feature representation in all folds. Additionally, it was shown that simple statistical pooling functions like Mean or Majority Voting are equal or often even better than more complex pooling functions like Support Vector Machines. Overall, it was concluded that the integration of ensemble learning techniques is a powerful method for MIC pipeline improvement and performance boosting. As a best practice, Stacking based pipeline builds utilizing multiple architectures showed continuous and strong performance improvement, whereas the gain of other ensemble learning techniques is based on datasets preconditions. As future research, the Author plans to further analyze the impact of the number of folds in cross-validation based Bagging techniques, integrate more pooling functions in Augmenting, and extend the analysis on deep learning Boosting approaches. Furthermore, the applicability of explainable artificial intelligence techniques for ensemble learning based medical image classification pipelines with multiple models is still an open research field and requires further research.

7

Reproducibility of Performance Assessment

Various novel studies demonstrated that MIS models based on deep learning revealed powerful prediction capabilities and achieved similar results as radiologists regarding performance [13, 110]. Clinicians, especially from radiology and pathology, strive to integrate deep learning based MIS methods as clinical decision support systems in their clinical routine to aid in diagnosis, treatment, risk assessment, and reduction of time-consuming inspection processes [13, 110]. Throughout their direct impact on diagnosis and treatment decisions, correct and robust evaluation of MIS algorithms is crucial.

However, in the past years a strong trend of highlighting or cherry-picking improper metrics to show particularly high scores close to 100% was revealed in scientific publishing of MIS studies [32, 44–48]. Studies showed that statistical bias in evaluation is caused by issues reaching from incorrect metric implementation or usage to missing hold-out set sampling for reliable validation [32, 44–48, 225, 227, 440, 441]. This led to the current situation that various clinical research teams are reporting issues on model usability outside of research environments [32, 45, 52, 75, 442–444]. The use of faulty metrics and missing evaluation standards in the scientific community for the assessment of model performance on health-sensitive procedures is a large threat to the quality and reliability of CDS systems.

In this chapter, the Author presents three individual studies to address statistical bias in the performance assessment of medical image segmentation pipelines and to increase reproducibility in the field.

7.1 MISEval: a Metric Library for Medical Image Segmentation Evaluation

Due to the observed widespread statistical bias often caused by incorrect metric implementation, standardized frameworks or APIs for performance assessment in MIS are needed to ensure reproducibility and comparability between studies. However, to the Author's knowledge, there was no universal metric library in Python for standardized and reproducible evaluation.

Therefore, as part of this dissertation, the open-source publicly available Python package MISEval was proposed, which is a metrics library for correct MIS model evaluation. It facilitates an intuitive and fast usage of various popular metrics from literature, as well as ensures implementation functionality and stability.

```

1 # Load libraries
2 import numpy as np
3 from miseval import evaluate
4
5 # Get some ground truth / annotated segmentations
6 np.random.seed(1)
7 real_bi = np.random.randint(2, size=(64,64)) # binary (2 classes)
8 real_mc = np.random.randint(5, size=(64,64)) # multi-class (5 classes)
9 # Get some predicted segmentations
10 np.random.seed(2)
11 pred_bi = np.random.randint(2, size=(64,64)) # binary (2 classes)
12 pred_mc = np.random.randint(5, size=(64,64)) # multi-class (5 classes)
13
14 # Run binary evaluation
15 dice = evaluate(real_bi, pred_bi, metric="DSC")
16 # -> returns single np.float64 e.g. 0.75
17
18 # Run multi-class evaluation
19 dice_list = evaluate(real_mc, pred_mc, metric="DSC", multi_class=True,
20                     n_classes=5)
21 # -> returns array of np.float64 e.g. [0.9, 0.2, 0.6, 0.0, 0.4]
22 # for each class, one score

```

Code Snippet 7.1: Usage example of the MISEval package.

7.1.1 Implementation

The open-source Python module MISEval is a metric library for **Medical Image Segmentation EVALuation**. The library contains various commonly used metrics for image segmentation, which can be easily imported and instantly used for model performance assessment. MISEval is structured as an API with a central core interface for intuitive usage and is implemented in the programming language Python, which is platform-independent and highly popular for computer vision tasks. This allows simple and fast integration of MISEval in commonly used platforms like Tensorflow [72], PyTorch [71], or any NumPy-compatible image segmentation pipeline [138].

Metric library

Over the last decades, the MIS literature introduced a large variety of metrics for evaluation. Especially for semantic segmentation, model performance assessment can be quite complex due to the need for scoring pixel classification as well as localization correctness between predicted and annotated segmentation. The MISeval metric library contains popular metrics like Dice Similarity Coefficient (DSC), Intersection-over-Union (IoU), Sensitivity (Sens), Specificity (Spec), Pixel Accuracy (Acc), AUC, Cohen's Kappa (Kap), and Average Hausdorff Distance (AHD), but also more complex metrics like entropy-based divergence and boundary-based distances. A summary of all metrics in MISeval can be seen in Table 7.1. The in-detail metric definitions and descriptions can be found in Chapter 3.5.1 and in the excellent review by Taha et al. [225].

Core Interface: Evaluate()

The core of the package is the `evaluate()` function, which acts as a simple and intuitive interface to access and run all implemented metrics. The function documentation (docstring) and an example usage can be seen in Code Snippet 7.2, as well as Code Snippet 7.1, respectively. The desired backbone metric for the `evaluate()` function can be defined by passing the name of an already implemented metric or by passing a user-created metric function for uncomplicated integration of custom metrics. Moreover, the core function handles automatically binary as well as multi-class problems. This allows the straightforward passing of any ground truth and predicted segmentation masks to the `evaluate()` function for computing the metric assessment in a single line of code.

```
1 """
2 Arguments:
3     truth (NumPy Matrix):           Ground Truth segmentation mask.
4     pred (NumPy Matrix):           Prediction segmentation mask.
5     metric (String or Function):    Metric function. Either a function directly or encoded as
6                                     String from miseval or a custom function.
7     multi_class (Boolean):         Boolean parameter, if segmentation is a binary or multi-class
8                                     problem. By default False -> Binary mode.
9     n_classes (Integer):           Number of classes. By default 2 -> Binary
10    kwargs (arguments):            Additional arguments for passing down to metric functions.
11
12 Output:
13     score (Float) or scores (List of Float)
14
15     The multi_class parameter defines the output of this function.
16     If n_classes > 2, multi_class is automatically True.
17     If multi_class == False & n_classes == 2, only a single score (float) is returned.
18     If multi_class == True, multiple scores as a list are returned (for each class one score).
19 """
20 def evaluate(truth, pred, metric, multi_class=False, n_classes=2, **kwargs)
```

Code Snippet 7.2: MISeval core function `evaluate()` with docstring documentation.

Table 7.1: Overview and comparison of currently implemented metrics in MISeval.

Group	Metric	scikit-learn	VISCERAL	PyMIA	Tensorflow	TorchMetrics	MISeval
Spatial Overlap	Dice Similarity Coefficient / F1-score	X	X	X	X	X	X
	Intersection-Over-Union / Jaccard Index	X	X	X	X	X	X
	Sensitivity / Recall	X	X	X	X	X	X
	Specificity		X	X	X	X	X
	Precision	X	X	X	X	X	X
Spatial Distance	(Average) Hausdorff		X	X			X
	Bhattacharyya						X
	Canberra						X
	Chebyshev						X
	Chi Square	X					X
	Cosine	X			X		X
	Euclidean	X					X
	Manhattan	X			X		X
	Hamming	X			X	X	X
	Mahanabolis		X				
	Minkowski						X
	MAE / MSE	X		X	X		X
	Pearson						X
	Correlation	Interclass Correlation		X	X		
Matthews Correlation		X			X	X	X
Divergence	Jensen-Shannon						X
	Kullback-Leibler				X	X	X
	Cross-Entropy	X			X		X
	Hinge	X			X	X	X
Probabilistic	AUC	X	X	X	X	X	X
	Cohen Kappa	X	X	X	X	X	X
or	Accuracy / Rand Index	X	X	X	X	X	X
Pairing	Balanced Accuracy	X					X
	Adjusted Rand Index	X	X	X			X
Volume	Volumetric Similarity		X				X

Package Stability

MISeval utilizes modern DevOps strategies to ensure package stability and functionality during ongoing development [445]. After each update, the source code is automatically built in a reproducible environment, extensively tested via unit testing, released, and, finally, deployed in the scientific community's MIS projects.

The unit testing considers functionality, edge cases, and exceptions for each metric. For application (functionality and edge cases), multiple dummy dataset types like empty, full, or random segmentation masks, as well as single and multi-class masks, are tested in all combinations. For exception handling, cases with incorrect parameter usage and non-matching mask shapes are tested.

Package Availability

The MISeval package is hosted, supported, and version-controlled in the Git repository platform GitHub [219]. This allows the utilization of platform-hosted DevOps workflows and a hub for package documentation, community contributions, bug reporting as well as feature requests.

The Git repository is available under the following link: <https://github.com/frankkramer-lab/miseval>.

Furthermore, MISeval is published in the Python Package Index (PyPI), which is the official third-party software repository for Python [364]. Thus, MISeval can be directly installed and immediately used in any Python environment using “*pip install miseval*”.

The source code is licensed under the open-source GNU General Public License Version 3 (GPL-3.0 License), which allows free usage and modification for anyone.

7.1.2 Quantitative Evaluation

For quantitative evaluation, the metric library was compared with other widely used frameworks for machine learning and image analysis. As it can be seen in Table 7.1, MISeval currently provides 28 metrics, which is the highest number of segmentation metrics compared to other analyzed frameworks: scikit-learn [342] with 18, EvaluateSegmentation from VISCERAL [225] with 13, PyMIA [393] with 12, Tensorflow [72] with 16 and TorchMetrics [446] with 12.

7.1.3 Qualitative Evaluation

For qualitative evaluation and functionality demonstration, a deep-learning based MIS pipeline was setup for training a COVID-19 segmentation model for CT scans, computing predictions, and evaluation of model performance using MISeval. The analysis utilized the MIS framework MIScnn [49] with default parameters. As dataset, the annotated computed tomography scans of COVID-19 positive patients from Ma et al. [102] were used. The evaluation results are illustrated in Figure 7.1. The figure compares an untrained model (after 1 epoch during training)

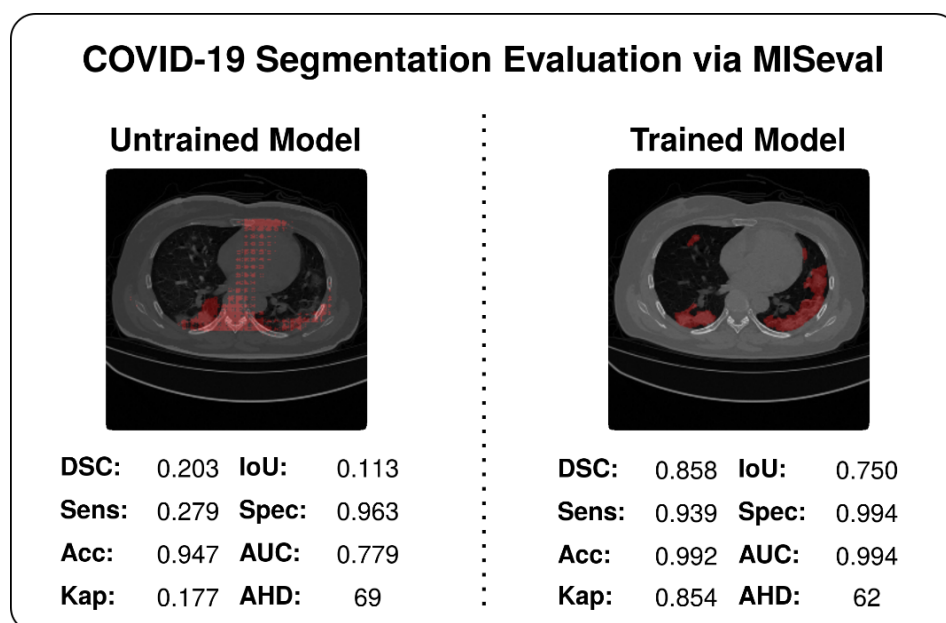


Figure 7.1: Illustration of various selected metrics from the library of MISeval to evaluate model performance on the use case COVID-19 infected region segmentation.

to a fully trained model (after 163 epochs) and shows computed tomography scans for each model with predicted infected regions (red).

7.1.4 Discussion

The proposed package MISEval allows a universal, reproducible, and standardized application of various metrics for MIS evaluation, which hopefully reduces the risk of statistical bias in studies through incorrect custom implementations. By following the state-of-the-art package stability and availability strategies, MISEval has the potential to be integrated into any future scientific performance analysis due to package stability, easy accessibility, and further contribution possibilities.

The road map and future direction for MISEval are to ensure ongoing support, the further extension of the metric library, and to provide guidelines on correct metric usage as well as evaluation.

7.2 Towards a Guideline for Evaluation Metrics

Robust evaluation or model performance assessment strategies are highly distinctive in computer vision between different research fields and applications. These differences range from the computer vision task, like the sample-wise evaluation in classification compared to the pixel-and-sample-wise evaluation in image segmentation, to the contrasts of an acceptable error rate in application fields, like medicine or autonomous driving compared to the manufacturing industry. The principles behind the heterogenous computer vision fields entail different biases for interpretation, which is why evaluation procedures must be adapted to these.

The uprising spread of statistical bias in MIS studies shows a clear need for a standardized MIS evaluation procedure to guide in metric usage and interpretation. Therefore, as part of this dissertation, various interpretation biases, as well as common metric behavior pitfalls, are discussed and a guideline for properly evaluating MIS performance is proposed. The Author's hope is that such guidelines increase research reliability, study comparability, and reproducibility in the field of medical image segmentation.

7.2.1 Bias: Class Imbalance

Medical images are infamous in the field of image segmentation due to their extensive class imbalance [225, 237]. Usually, an image in medicine contains a single ROI taking only a small percentage of pixels in the image, whereas the remaining image is all annotated as background. From a technical perspective for machine learning, this scenario entails that the model classifier must be trained on data composed of a very rare ROI class and a background class with often more than 90% or even close to 100% prevalence. This extreme inequality in class distribution affects all aspects of a computer vision pipeline for MIS, starting from the preprocessing, to the model architecture and training strategy up to the performance evaluation [49].

In MIS evaluation, class imbalance significantly affects metrics which include correct background classification. For metrics based on the confusion matrix, these cases are defined as true negatives. In a common medical image with a class distribution of 9:1 between background and ROI, the possible number of correct classifications is extensively higher for the background class compared to the ROI. Using a metric with equal true positive and true negative weighting results in a high-ranking scoring even if any pixel at all is classified as ROI and, thus, significantly biases the interpretation value. This behavior can be seen in metrics like Accuracy or Specificity which present always significantly high scorings in any MIS context. Therefore, these metrics should be avoided for any interpretation of segmentation performance. Metrics that focus on only true positive classification without a true negative inclusion provide better performance representation in a medical context. This is why the DSC and IoU are highly popular and recommended in the field of MIS.

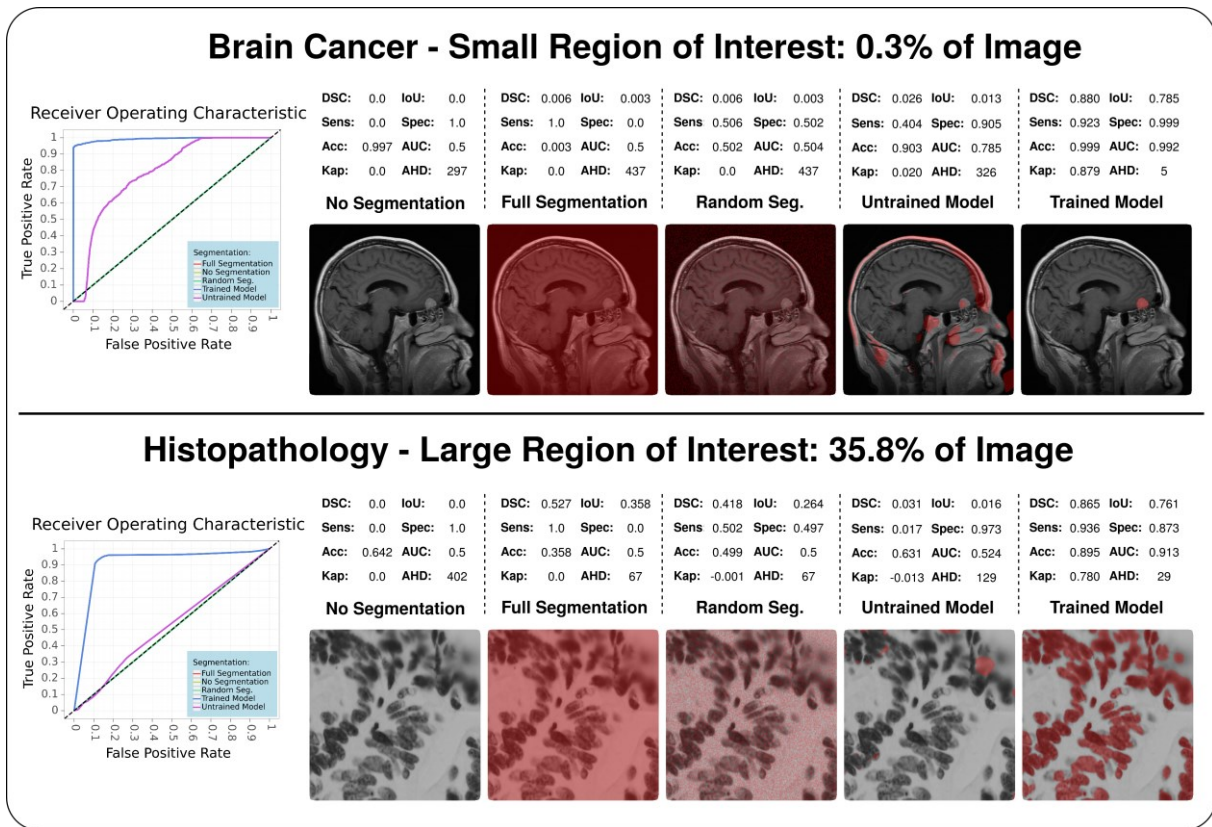


Figure 7.2: Metric behavior demonstration in the context of different-sized ROIs compared to the total image.

7.2.2 Bias: Influence of the Region-of-Interest Size

The size of an ROI and the resulting class imbalance ratio in an image demonstrates an anti-correlation to evaluation complexity for interpretation robustness. In the medical context, the ROI size is determined by the type in terms of the medical condition and the imaging modality. Various types of ROIs can be relevant to segment for clinicians. Whereas organ segmentation, cell detection, or a brain atlas take up a larger fraction of the image and, thereby, represent a more equal background-ROI class ratio, the segmentation of abnormal medical features like lesions commonly reflects the strong class imbalance and can be characterized as more complex to evaluate. Furthermore, the imaging modality highly influences the ratio between ROI and background. Modern high-resolution imaging like whole-slide images in histopathology provides resolutions of $0.25\mu\text{m}$ with commonly $80,000 \times 60,000$ pixels [120, 132] in which an anaplastic (poorly differentiated) cell region takes up only a minimalistic part of the image. In such a scenario, the resulting background-ROI class ratio could typically be around 183:1 (estimated by a 512×512 ROI in an 803×603 slide). Another significant class ratio increase can be observed in 3D imaging from radiology and neurology. Computer tomography or magnetic resonance imaging scans regularly provide image resolutions of 512×512 pixels with hundreds of slices (z-axis) resulting in a typical class ratio of around 373:1 (estimated by a 52×52 ROI in a $512 \times 512 \times 200$ scan) [120]. In order to avoid such extreme imbalance bias, metrics that are distance-based like AHD or exclude true negative rewarding like DSC are recommended. Besides that, patching techniques (splitting the slide or scan into multiple smaller images) are often also applied to reduce complexity and class imbalance [110, 132].

7.2.3 Bias: Influence of the Segmentation Task

For valid interpretation of an MIS performance, it is crucial to understand metric behaviors and expected scores in different segmentation tasks. Depending on the ROI type like a lesion or organ segmentation, the complexity of the segmentation task and the resulting expected score varies significantly [235]. In organ segmentation, the ROI should be located consistently at the same position with low spatial variance between samples, whereas an ROI in lesion segmentation shows high spatial as well as morphological variance in its characteristics. Thereby, optimal performance metrics in organ segmentation are more likely to be possible, even though less realistic in lesion segmentation [109, 125]. This complexity variance implicates expected evaluation scores and should be factored in performance interpretation. Another important influencing factor in the segmentation task is the number of ROIs in an image. Multiple ROIs require additional attention for implementation and interpretation because not only high scoring metrics can be misleading and hiding undetected smaller ROIs but also distance-based metrics are defined only on pairwise instance comparisons [235]. These risks should be considered in any evaluation of multiple ROIs.

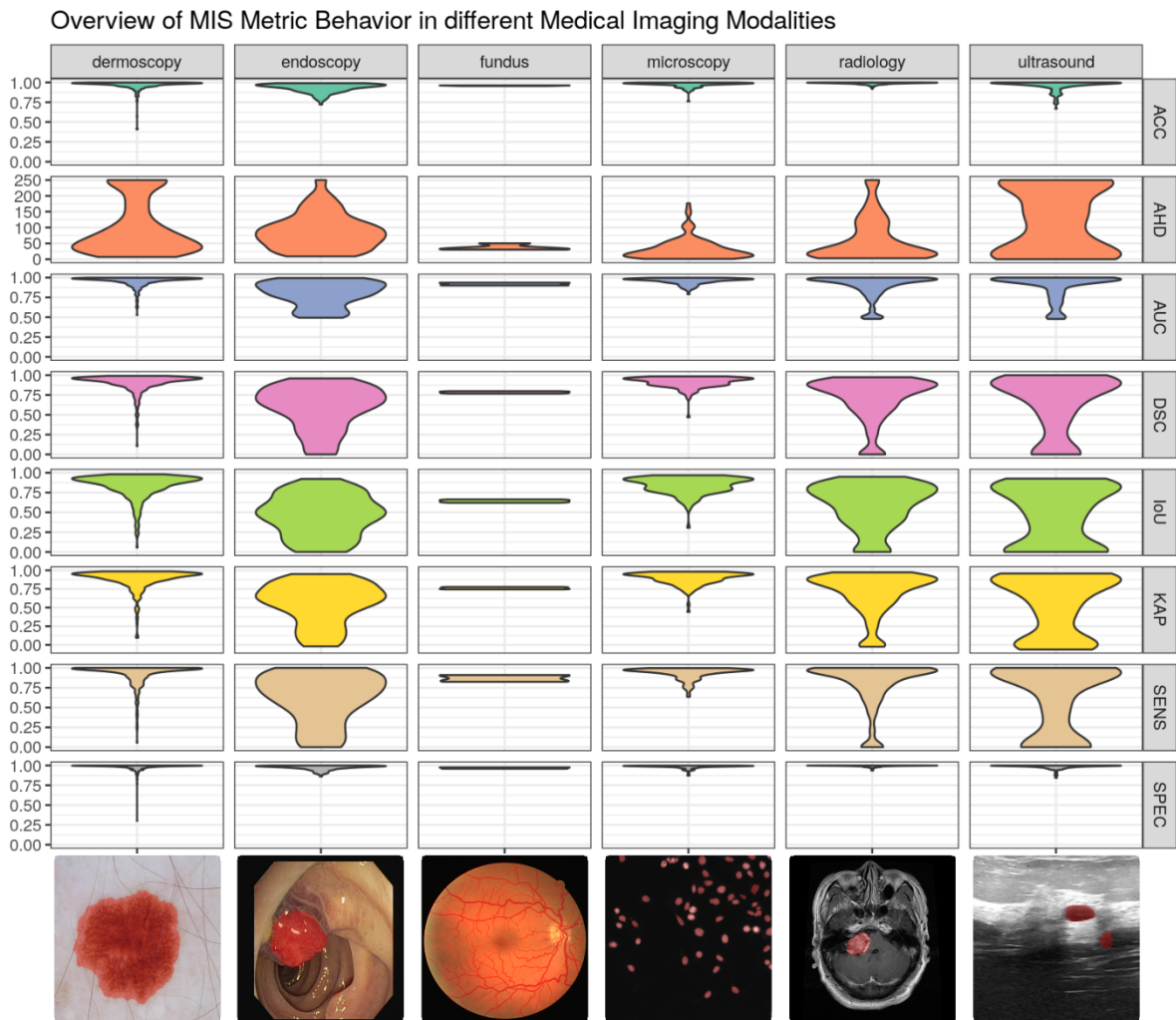


Figure 7.3: Demonstration of metric behavior for trained segmentation models in the context of different medical imaging modalities.

7.2.4 Bias: Multi-class Evaluation

In nearly all cases, evaluation metrics are defined for binary classification or segmentation problems. It is needed to be aware that applying binary metrics to multi-class problems can result in highly biased results, especially in the presence of class imbalance [47]. This can often lead to a confirmation bias and promising-looking evaluation results in scientific publications which, however, are actually quite weak [47]. In order to evaluate multi-class tasks, it is required to compute and analyze the metrics individually for each class. Distinct evaluation for each class is in the majority of cases the most informative and comparable method. Nevertheless, it is often necessary to combine the individual class scores into a single value for improving clarity or for further utilization, for example as a loss function. This can be achieved by micro and macro averaging the individual class scores. Whereas macro-averaging computes the individual class metrics independently and just averages the results, micro-averaging aggregates the contributions of each class for computing the average score.

7.2.5 Proposed Guideline

- Use DSC as main metric for validation and performance interpretation.
- Use AHD for interpretation of point position sensitivity (contour) if needed.
- Watch out for class imbalance and avoid interpretations based on high Accuracy.
- Provide next to DSC also IoU, Sensitivity, and Specificity for method comparability.
- Provide sample visualizations, comparing the annotated and predicted segmentation, for visual evaluation as well as to avoid statistical bias.
- Avoid cherry-picking high-scoring samples.
- Provide histograms or box plots showing the scoring distribution across the dataset.
- Keep in mind variable metric outcomes for different segmentation types.
- Be aware of interpretation risks by multiple ROIs.
- For multi-class problems, provide metric computations for each class individually.
- Avoid confirmation bias through macro-averaging classes which is pushing scores via background class inclusion.
- Provide access to evaluation scripts and results with journal data services or third-party services like GitHub [219] and Zenodo [447] for easier reproducibility.

Sample Visualization

Besides the exact performance evaluation via metrics, it is strongly recommended to additionally visualize segmentation results. Comparing annotated and predicted segmentation allows robust performance estimation by eye. Sample visualization can be achieved via binary visualization of each class (black and white) or via utilizing transparent color application based on pixel classes on the original image. The strongest advantage of sample visualization is that statistical bias, overestimation of predictive power through unsuited or incorrect computed metrics, is avoided.

7.2.6 Experiments on Metric Behavior

Evaluation of semantic segmentation can be quite complex because it is required to measure classification accuracy as well as localization correctness. The aim is to score the similarity between the predicted (prediction) and annotated segmentation (ground truth). Over the last 30 years, a large variety of evaluation metrics can be found in the MIS literature [225]. However, only a handful of scores have proven to be appropriate and are used in a standardized way [225].

Multiple experiments were conducted for supporting the principles of the proposed evaluation guideline as well as demonstrate metric behaviors on various medical imaging modalities. Furthermore, the previously discussed insights are based on the experience during the development and application of the popular framework MIScnn [49] as well as the contribution to currently running or already published clinical studies [110, 159, 448, 449].

The experiments demonstrate the behavior of the following common metrics for evaluation:

- F-measure based metrics like Dice Similarity Coefficient (DSC) and Intersection-over-Union (IoU)
- Sensitivity (Sens) and Specificity (Spec)
- Accuracy / Rand Index (Acc)
- Receiver Operating Characteristic (ROC) and the area under the ROC curve (AUC)
- Cohen's Kappa (Kap)
- Average Hausdorff Distance (AHD)

In detail descriptions of these metrics are presented in Chapter 3.5.1. The behavior of the metrics is illustrated in Figure 7.2 and Figure 7.3 which demonstrate the metric application in multiple use cases.

Figure 7.2 is showing the perks of F-measure based metrics like DSC as well as IoU and the inferiority of Rand index usage. Furthermore, the small ROI segmentation points out that metrics like Accuracy have no value for interpretation in these scenarios, whereas the large ROI segmentation indicates that small percentage variance can lead to a risk of missing whole instances of ROIs. The analysis was performed in the following scenarios and common MIS use cases. Scenarios: No segmentation (no pixel is annotated as ROI), full segmentation (all pixels are annotated as ROI), random segmentation (full random-based annotation), untrained (after 1 epoch during training), and trained model (fully fitted model). Use cases: Small ROIs via brain tumor detection in magnetic resonance imaging and large ROIs via cell nuclei detection in pathology microscopy.

Figure 7.3 is showing the differences between metrics based on distance like AHD, with true negatives like Accuracy, and without true negatives like DSC. Each subplot illustrates a violin plot that visualizes the resulting scoring distribution of all testing samples for the corresponding metric and modality. For visualization purposes, AHD was clipped to a maximum of 250

(affected number of samples per dataset: dermoscopy 2.0%, endoscopy 0.3%, fundus 0.0%, microscopy 0.0%, radiology 0.5%, and ultrasound 2.5%).

The analysis utilized the medical image segmentation framework MIScnn [49] and was performed with the following parameters: Sampling in 64% training, 16% validation, and 20% testing sets; resizing into 512x512 pixel images; value intensity normalization via Z-Score; extensive online image augmentation during training, common U-Net architecture [108] as neural network with focal Tversky loss function [450] and a batch size of 24 samples; advanced training features like dynamic learning rate, early stopping and model checkpoints. The training was performed for a maximum of 1,000 epochs (68 up to 173 epochs after early stopping) and on 50 up to 75 randomly selected images per epoch. For metric computation and evaluation, the framework MISeval was utilized, which provides implementation and an open interface for all discussed evaluation metrics in a Python environment. In order to cover a large spectrum of medical imaging with the experiments, datasets from various medical fields were integrated: Radiology - brain tumor detection in magnetic resonance imaging by Cheng et al. [60, 157], ultrasound - breast cancer detection in ultrasound images [61], microscopy - cell nuclei detection in histopathology by Caicedo et al. [451], endoscopy - endoscopic colonoscopy frames for polyp detection [56], fundus photography - vessel extraction in retinal images [452], dermoscopy - skin lesion segmentation for melanoma detection in dermoscopy images [58].

7.3 Performance Assessment in Presence of Control Samples

Next to the consequent statistical bias related to also incorrect metric usage, it has been proven that the current metrics have limitations in covering certain edge cases like weak labels [46, 47, 225, 226, 453]. Weak labels are sample annotations with no ROI in the segmentation mask. These cases without any present condition are important for evaluation in the medical field, as additional control patients are a standard procedure in any proper clinical trial. However, predictions of weak labels are not taken into account and rated zero by common metrics like the DSC, regardless of the correctness of the prediction [293]. It is crucial that the segmentation masks of control patients are also correctly inferred by a model, which is why a metric covering these edge cases is essential.

This subchapter describes developing an MIS metric that covers the limitations of current widely used metrics in evaluating weakly labeled data.

7.3.1 Proposed Metric: MISm

The MISm has been included in the previously described package MISeval: a Metric Library for Medical Image Segmentation Evaluation. Throughout this subchapter, the definition of a metric is used by Taha et al. [225].

Metric Definition

For solving the weak label scoring issue which is present in most metrics in MIS, the MISm: **Medical Image Segmentation metric**, was proposed.

$$MISm = \begin{cases} \frac{2TP}{2TP + FP + FN} & \text{if } P > 0 \\ \frac{\alpha TN}{(1 - \alpha) FP + \alpha TN} & \text{if } P = 0 \end{cases} \quad (7.1)$$

The operators are based on the computation of a confusion matrix for binary segmentation, which contains the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions. P is the number of the actual positive conditions and therefore the sum of TP and FN. Furthermore, the weighting coefficient $\alpha \in [0, 1]$ was introduced. The first part of the formula is equivalent to the Dice Similarity Coefficient or F1-score (DSC). The second part is the Specificity (Spec), also called True Negative Rate, where additional weights α and $(1 - \alpha)$ were added, which was defined as weighted Specificity ($wSpec_\alpha$). Analogous to the DSC and the Specificity, MISm returns values between $[0, 1]$ with one being equal to the ground truth annotation and zero implying no overlap between ground truth and prediction.

Metric Verification

For the performance measurement of a prediction, common metrics in MIS entail two main limitations. If there is no ground truth annotation, meaning $P = 0$ and, therefore, $TP = FN = 0$, then DSC, which is widely used in MIS [13, 225, 227], takes zero. For $FP = 0$ or close to zero, the segmentation is an actually accurate true negative, which contradicts the DSC being zero or non-defined. To measure the impact of TN and FP in this case, it is possible to estimate the false positive rate (FPR), also called fall-out, with the formula

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \quad (7.2)$$

where N is the number of the actual negative condition. If everything is predicted incorrectly, so $TN = 0$, FPR takes one. To have the common metric range where one means a perfect prediction and zero an incorrect prediction, FPR is reversed and transformed into the Specificity, also called true negative rate, which is commonly used in medicine.

$$1 - FPR = 1 - \frac{FP}{FP + TN} = \frac{FP + TN}{FP + TN} - \frac{FP}{FP + TN} = \frac{TN}{FP + TN} = Spec \quad (7.3)$$

The application of the Specificity results in the second limitation for the performance measurement of a prediction. Assume $N = 60,000$ and $P = 0$. Let $FP = 5,000$, thus $TN = N - FP = 55,000$. As almost 10% were predicted false positive, an MIS prediction can be postulated as inaccurate. However,

$$Spec = \frac{55,000}{5,000 + 55,000} \approx 0.9167 \quad (7.4)$$

implying an acceptable model. In order to fix this inaccuracy, FP and TN were weighted to each other by adding weights α and $1 - \alpha$ to the formula.

$$wSpec_{\alpha} = \frac{\alpha TN}{(1 - \alpha) FP + \alpha TN} \quad (7.5)$$

In the example above, let $\alpha = 0.2$. This yields to

$$wSpec_{0.2} = \frac{0.2 * 55,000}{(1 - 0.2) * 5,000 + 0.2 * 55,000} \approx 0.7333 \quad (7.6)$$

which results in a more insightful scoring and represents the second part of the MISm.

7.3.2 Theoretical Analysis

In the following theoretical analysis, definition gaps, as well as appropriate scoring gradient compared to the DSC and Spec, were investigated.

$$TN + FP = 0 \Leftrightarrow TN = FP = 0 \tag{7.7}$$

The Spec is not defined, if $TP \geq 0, FN \geq 0, TN = 0$ and $FP = 0$. As $P = TP + FN \geq 0$, MISm computes the DSC and is, therefore, defined, although the Spec is not. Analogously, the definition gap of the DSC.

$$2TP + FP + FN = 0 \Leftrightarrow TP = FP = FN = 0 \tag{7.8}$$

Thus, the DSC is not defined for $TN \geq 0$ and the other values being zero, which represents a completely true negative prediction. MISm handles this edge case separately by computing the weighted Specificity as $P = 0$. Therefore, it can be concluded that MISm is always defined.

Analyzing the scoring gradient, regarding $P = 0$ but let $FP > 0$: In this case, $DSC = 0$ and any prediction will yield the same score, despite a strong possible variation of FP. The fixed scoring outcome is not capable of reflecting the prediction quality, properly. In contrast to that, the Spec grades the predictions but underweights FP as seen in the example above. As MISm utilizes a weighted Spec in the case $P = 0$, a more appropriate scoring gradient is sustained.

7.3.3 Weighting Coefficient Analysis

Different weighting coefficients were investigated for MISm and their impact on scoring capabilities on the edge cases, in which no predictions are present in the mask. To visualize the

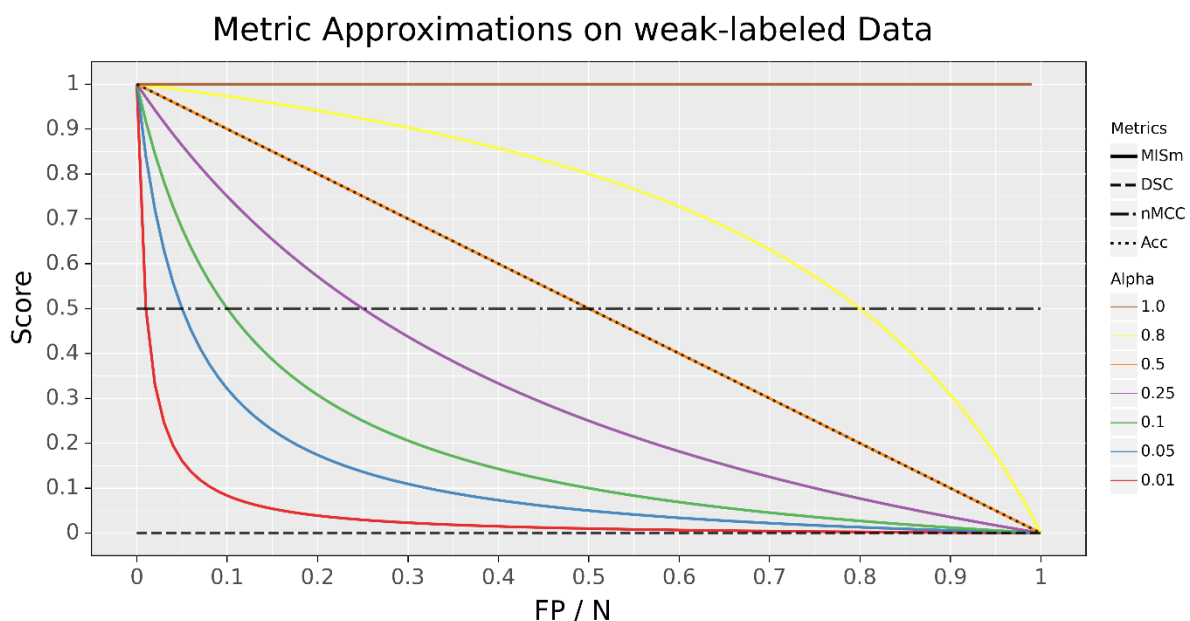


Figure 7.4: Comparison of the performance metrics considered with the presented MISm in terms of the ratio of false positives to actual negatives if the class observed is not present in the image.

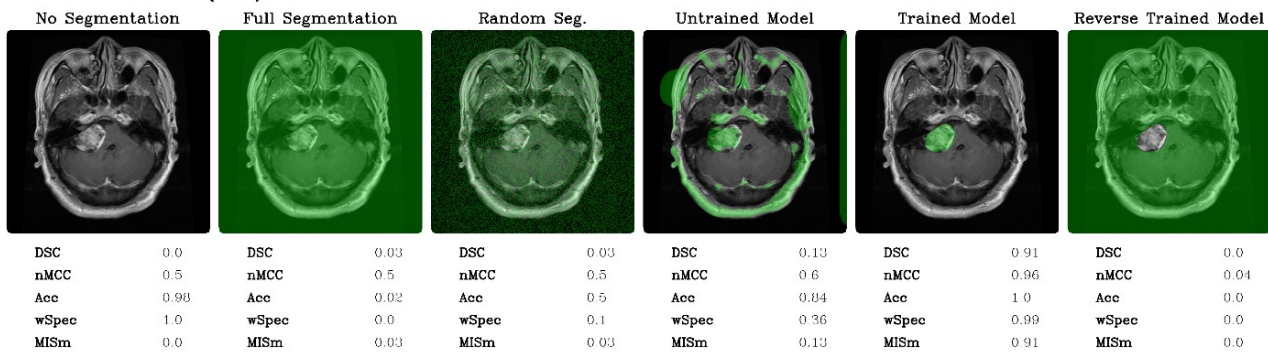
score with different weighting coefficients and to compare it with popular metrics of the research field, the different scores in comparison to the ratio of pixels falsely classified as positive to pixels classified as negative for the edge cases ($P = 0$) were plotted in Figure 7.4.

Even though MISm provides a foundation for the evaluation of datasets with control samples, the weighting factor α is a dynamic variable that results in inconsistent performance assessments by varying weighting factors. The selection of the weighting factor α is still a subjective definition of the assessor, which causes the usage of MISm for quantitative evaluation as ineffective due to the incomparability as consequence. In order to utilize MISm as a generic evaluation method for objective performance measurement, it is mandatory to use a fixed and community-accepted weighting factor. The Author proposes the weighting factor as $\alpha = 0.1$, which is implemented in the software MISeval as the default weight for MISm.

7.3.4 Experimental Application

For experimental application, MISm was compared with popular metrics, such as Accuracy (Acc), Dice Similarity Coefficient (DSC), normalized Matthews Correlation Coefficient (nMCC), and weighted Specificity (wSpec), for MRI brain tumor segmentation, which can be seen in Figure 7.5. Part A of the figure shows an MRI scan of the brain with an annotated tumor [60, 157]. Based on annotation, various predicted segmentation cases were tested for evaluation ($P > 0$). In part B, the edge cases in which no tumor is present in the image ($P = 0$), were illustrated [454]. Popular MIS evaluation metrics and the proposed MISm were calculated for the respective cases to allow comparability between them. For wSpec and MISm, $\alpha = 0.1$ was selected.

A: Normal Cases ($P > 0$)



B: Edge Cases ($P=0$)

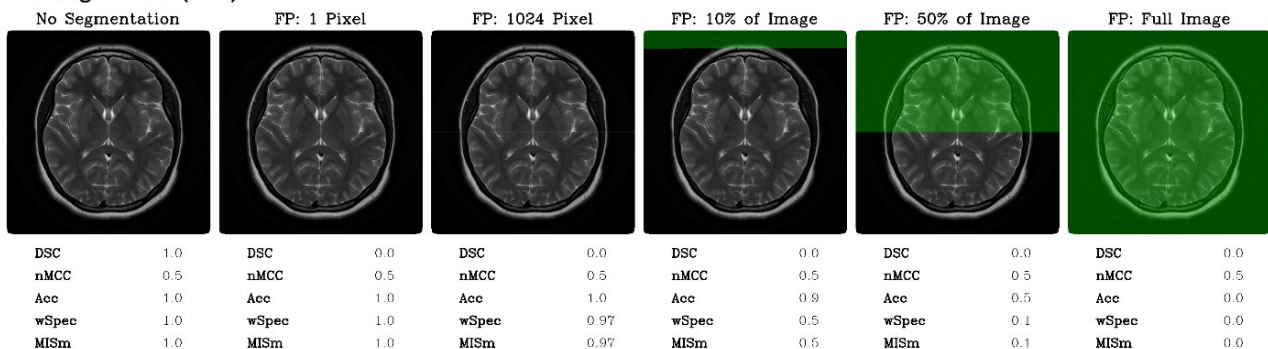


Figure 7.5: Scoring comparison between MISm and multiple common MIS metrics by application on normal as well as edge cases.

7.3.5 Discussion

MISm proposes a solution to the limitations identified within current gold-standard metrics popular in the field of MIS. By utilizing wSpec, MISm allows evaluating datasets with weak label annotations. To guarantee flexibility in usage, the weighting α was designed flexibly. However, for performance assessment in the context of evaluation, it is strongly recommended to use the proposed fixed weighting coefficient $\alpha = 0.1$. In summarizing, it was proved that MISm is an always applicable metric that is suitable for appropriate prediction scoring.

The MISm equals the DSC if the mask contains an actual positive annotation. As regards, there is no ground truth annotation in the mask, the metrics differ significantly as shown in the theoretical analysis and experimental application. The DSC provides a constant value of zero, whereas the MISm has an adequate scoring gradient by decreasing appropriately for each error, starting at a value of one. In comparison to the nMCC, which is also widely used in several MIS studies [186, 455–457], similar limitations as the DSC were identified, as shown in Figure 7.4 and Figure 7.5. Furthermore, interpretation of nMCC is not always intuitive, because a score equal to zero corresponds to an inversed ground truth annotation and 0.5 is equivalent to randomness. All other cases of prediction inaccuracy converge to or take a value of 0.5. This is why the nMCC insufficiently evaluates the quality of predictions.

The weighting coefficient analysis and experimental application revealed that the Accuracy is capable of scaling in absence of actual positives. Still, the score is massively influenced by the inclusion of true negatives due to their significantly higher number in MIS. Even so, the Accuracy is capable of handling the identified edge cases, the true negative inclusion constrains the application in practical settings.

Utilization as Loss Function

As future work, the capabilities of the metric MISm as a loss function for model training are quite promising. Currently, the inclusion of control samples in the training process based on an MIS dataset is rare due to the difficulties in performance assessment for the loss function as well as the minimal information gain of control samples for the model. However, the significant difficulties in utilizing MIS models in clinical routine [32, 45, 52, 75, 442] indicate that current state-of-the-art models from research are often overfitted on their task. Passing images to these models with different medical conditions, from healthy patients or with non-relevant imaging abnormalities like artifacts drastically reduces performance or leads to complete malfunctioning of the prediction. Training models with loss functions capable of also scoring control samples could help reduce the overfitting bias and overcome this challenge. Thus, the integration of the MISm as a loss function into popular neural network frameworks like PyTorch and TensorFlow is planned.

8

Discussion

In this chapter, the Author discusses the advancements of medical image segmentation and classification focusing on the general developments in the field, the proposed frameworks, the contributions by the Author, and limitations as well as further challenges. The main objective of this chapter is to contextualize the impact of the presented contributions toward the field and discuss the gained insights in the ‘big picture’. Moreover, the advancements through the presented contributions to further research fields are also briefly discussed. In addition, the translational software integration process of imaging-based AI models from the university lab into clinical application is addressed. Finally, the Author presents future work and an outlook on deep neural network based medical image analysis.

8.1 Advancements in Medical Image Segmentation

The automatic segmentation of ROIs in medical images presents a significant challenge. Whereas the field of medical image processing itself consists of multiple obstacles like biases from technical noise or inconsistent signal assessment and a strong variance in modalities, the automatic processing of medical images is characterized by performing complex distinguishing tasks for which a human would require years of experience and knowledge [13, 182]. Starting from 2015 with the breakthrough architecture U-Net by Ronneberger et al. [108], researchers were capable of developing MIS pipelines with exceptional segmentation accuracy [13, 105, 125]. In the last years, state-of-the-art MIS pipelines like those by Isensee et al. [109, 118] demonstrated outstanding performance across various imaging modalities as well as medical conditions. With excellent performing convolutional neural network and deep learning models, the urge to move automatic MIS from the research labs into practical application in clinics is uprising [24, 52, 62, 442]. Still, the landscape of standalone pipelines of top-performing models, designed only for a single specific public dataset, has shown drastic issues in terms of reproducibility and reusability [32, 45]. Solving these challenges often requires extensive knowledge in the field of deep learning based MIS which significantly handicaps the progress on translational application of automated MIS in clinical research.

Over the course of the last four years, the awareness of limitations to reproducibility and clinical usability in MIS studies has rapidly increased. The research field experienced a shift from focusing on achieving the highest possible segmentation performance to studies about reproducibility, standardization, as well as reliability [44]. To counter the previously described issues, multiple frameworks for MIS also have been developed, recently [49, 109, 175]. One of these is the proposed framework MIScnn which is discussed in the next subchapter 8.1.1 in detail. However, the focus shift of the research field can be grouped into the following two topics: Research based on clinical application and deep learning centered research.

Due to the increased capabilities of modern MIS pipelines, clinicians strive to integrate segmentation models into clinical workflows for supporting diagnosis or treatment decisions as well as for automation of time-consuming processes [24, 52, 62, 442]. Research based on clinical application aims to setup as well as use MIS pipelines from the literature in clinical environments for processing local patient data. The focus of this research centers on the usability and generalizability aspects of MIS pipelines without the usage requirement of extensive experience in deep learning. Additionally, method reproducibility has become an important key element of novel frameworks, challenges, and meta-analysis studies [39, 44, 118, 226]. This can be also observed in the development and deployment quality of frameworks in which open-source strategies and software documentation are becoming more prevalent features in research studies, as well. The reason for this can be assessed to be the desired reduced complexity in the application of high-quality software and the high legal requirements of integrating artificial intelligence models into medical workflows (which is further discussed in Chapter 8.4).

Alongside the clinical application, deep learning centered research consists of studies that attempt to further improve the capabilities of MIS models. The field of MIS has heavily centralized on the U-Net architecture which is why other methods of improving the performance are focused on research instead of developing novel architectures [100, 103, 111, 125]. Most notably, studies tediously try to enhance the U-Net architecture with advanced mechanisms like semi-supervised learning, attention strategies, or transformers [100, 111, 125]. One of the largest fields of deep learning centered research is deep ensemble learning in which multiple model predictions are combined [118, 200]. Top-performing pipelines like nnU-Net [109] heavily utilize ensemble learning strategies like Stacking, Bagging, as well as Augmenting. The advancements in the field of ensemble learning are individually discussed in Chapter 8.3.1.

Next to research on the application and further performance increase, prediction robustness is an increasing field in MIS. Even powerful AutoML pipelines like nnU-Net revealed inferior performance in clinical applications [109, 458]. This can be explained through the overall trend of acceptable overfitting on published training data but which reduces the generalizability on realistic data in clinical settings resulting from unseen features in images, different device configurations, and unaccounted biases. Another research field gaining popularity is federated learning in which neural network models are trained decentralized by multiple sites holding local data without sharing [459, 460]. Through the increased interest of hospitals to contribute to AI research but not being able to permit data publication due to patient data privacy regulations, the concept of training models locally provides a perfect solution and allows a significant growth of available ‘real-world’ imaging data directly from hospitals.

8.1.1 The proposed Framework: MIScnn

The goal of the implemented framework MIScnn is to provide a high-level API for building MIS pipelines with preprocessing, data augmentation, model architecture selection, and model utilization. MIScnn offers a highly configurable and open-source pipeline with several interfaces for custom deep learning models, image formats, or fitting metrics. The modular structure of MIScnn allows an MIS novice to rapidly setup a functional pipeline for a custom dataset by using just a few lines of code. Additionally, switchable models and an automatic evaluation functionality allow robust and unbiased comparisons between deep learning models. The proposed universal framework for MIS, following the Python philosophy of simple and intuitive modules, is an important step in contributing to practical application development.

To ensure usability and generalizability, one of the key aspects of the proposed framework is open-source development and deployment. The philosophy behind MIScnn is that clinical researchers are able to setup MIS pipelines without being experts in the field of deep learning. Through the extensive documentation of the API usage, multiple example applications, and the provided stability of the package ensured by continuous integration processes based on unit testing, the proposed framework is able to offer an intuitive approach for building standardized and reusable pipelines for MIS. This concept of MIScnn has successfully enabled clinical

applications and is capable of supporting the integration of MIS models into already established workflows [461–466].

Through the conducted studies presented in this thesis and the popularity of MIScnn, estimated by the evaluation of GitHub repository metadata and the number of external studies which utilized the proposed framework, it was possible to prove the functionality, high performance, and framework usability of MIScnn. Utilizing an extensive library of state-of-the-art methods, MIScnn is capable of building modern MIS pipelines and allows generating robust as well as reliable models with even complex as well as class-imbalanced medical imaging data. Furthermore, the proposed framework demonstrated high modularity by enabling simple options for customization and supporting the development of specifically designed pipelines for custom workflows or datasets. Nevertheless, MIScnn offers intuitive usability based on a well-documented Python API. It was possible to conclude that MIScnn is a powerful toolkit for state-of-the-art medical image segmentation which has contributed to the standardization in the field of MIS. The Author hopes that MIScnn will further help migrate medical image segmentation models from the research labs into clinical applications.

Next to the application of MIScnn in deep learning and clinical research projects, the Author heavily utilized the proposed framework in education for medical information science students. Due to its design as a toolkit, MIScnn is simple to use but still close to the structure of an MIS pipeline implemented with a base framework. In the student projects, the framework acts as a foundation that allows a straightforward application but also options for customization and contribution of own features. This allows a more detailed understanding of the applied methods as well as concepts of MIS compared to using a black box framework which is called entirely through a command line interface.

Related Work and Limitations

Prior to the development of MIScnn, toolkits or frameworks for standardized building of MIS pipelines were rare. Nevertheless, one of the first established frameworks was NiftyNet by Gibson et al. [43]. The open-source platform NiftyNet provided building blocks for creating MIA pipelines including MIS in order to simplify the development process [43]. Similar to MIScnn, NiftyNet was also based on the TensorFlow base framework [43]. However, NiftyNet was designed as a configurable application instead of an API which complicated novel method integration and hampered the workflows of native deep learning experts. Subsequently, the framework significantly lacked capabilities to create state-of-the-art MIS pipelines which are competitive in performance with custom-developed pipelines. This is why MIScnn was developed which combined the capabilities for building state-of-the-art MIS pipelines with an intuitive and straightforward Python API.

After the first release of MIScnn in the year 2019, further novel frameworks for MIA were published for which the nnU-Net [109] and MONAI [175] have been the most prominent according to the popularity evaluated in Chapter 3.4.3. The framework nnU-Net (short for ‘no-new-U-Net’) is a self-adapting AutoML framework for deep learning based biomedical image

segmentation which was also released as an open-source project in 2019 by Isensee et al. [109] from the German Cancer Research Center (DKFZ). The framework provides powerful capabilities for fully automatic MIS pipeline building including hyperparameter configuration, preprocessing, neural network architecture, data processing for training as well as prediction, and ensemble learning strategies [109]. nnU-Net was able to prove superior performance in more than 23 international biomedical segmentation competitions and defined the state-of-the-art of MIS in terms of high performance as well as AutoML application [109]. In contrast to the AutoML framework nnU-Net, the framework MONAI (short for ‘Medical Open Network for AI’) is a toolkit for deep learning based biomedical image analysis which was released in the year 2020 [175]. MONAI has been developed by a collaboration of more than 16 institutes including the DKFZ, Stanford University, the Technical University of Munich, the National Cancer Institute of the U.S.A., and NVIDIA [175]. Similar to MIScnn, MONAI is a toolkit that can be utilized for building MIS pipelines by providing a Python API for data loading, pre/postprocessing, as well as neural network models.

Comparing the features of MIScnn with MONAI and nnU-Net reveals that the three frameworks share the majority of key methods and philosophies. All frameworks heavily utilize advanced preprocessing techniques like patching, training strategies like on-the-fly-augmentation, and centralization on the U-Net as neural network architecture. Furthermore, the projects advocate as well as employ the open-source mentality and have the same goal to simplify deep learning based MIS application. Furthermore, all three projects were successful in building an open-source community around the packages resulting in wide framework usage in research studies [467, 468]. However, there are also multiple differences between the frameworks. The most fundamental difference is that nnU-Net as well as MONAI are implemented in PyTorch [71] whereas MIScnn is based on TensorFlow [72]. Experiments by Chirodea et al. [469] comparing the two base frameworks with each other showed that there are only marginal differences and concluded that PyTorch is more oriented towards beginners while TensorFlow offers more options at the cost of user-friendliness. However, in the last years, the scientific community in computer vision tends to actively utilize PyTorch for research, whereas TensorFlow has become more popular in industry applications [470]. This trend was observed by analyzing the ratio of published studies utilizing Pytorch compared to TensorFlow and was performed by Horace He in the year 2019 [470]. Even though both frameworks provide the same functionality and offer a large community, the implementation availability of novel research methods in TensorFlow could decrease in the future which would also impact the possibilities of integrating new methods into MIScnn.

In terms of functionality, nnU-Net differs drastically as AutoML framework from toolkits like MIScnn or MONAI. Whereas MIScnn and MONAI require basic but still sufficient knowledge in deep learning based MIS in order to build a pipeline with the toolkit API, nnU-Net only requires data engineering expertise for application. The automation of nnU-Net is designed based on a set of fixed parameters, interdependent rules through design choices as well as heuristics, and decisions learned empirically from data [109]. This strategy allows obtaining proper configurations for any dataset and excellent performance in ‘out-of-the-box’

applications. However, an AutoML framework does not allow customization of utilized methodology and aggravates the ‘black box’ conception of neural network models. Still, the performance of MIScnn has the potential to be further improved by incorporating the automatic parameter findings of nnU-Net to enhance the default settings of the general MIScnn pipeline. Overall, the current field of MIS reveals distinct tendencies between methodology research, which focuses on toolkits based on the feasibility for customization as well as integration of novel methods, and clinical application research, which focuses on AutoML frameworks based on simplicity in usage to obtain strong segmentation performance.

The most adequate comparison is between MIScnn and MONAI as both frameworks are toolkits for building MIS pipelines by a Python API. Both frameworks aim to provide pipeline building blocks for allowing researchers to focus on experimentation in MIS without reimplementing of a complete pipeline from scratch as well as to offer a framework for standardized applications. Analyzing the building blocks of both frameworks reveal high similarity in terms of functionality and library size containing state-of-the-art methods. Furthermore, the two frameworks share similar philosophies highlighting reproducibility, standardization, and avoiding reimplementing of pipelines from scratch. However, the MONAI project provides overall more additional features in terms of recently published methods from the literature, the number of alternative training strategies as well as architectures, and deployment solutions. Also, MIScnn only supports class weighting but not sample weighting for loss functions. The lack of features in MIScnn can be explained through MONAI's considerably larger contributor community from multiple universities and labs. This allows the large-scale implementation of additional methods, maintaining extensive documentation as well as code quality, rapid integration of novel techniques, and providing comprehensive support for users. Due to the higher financial support and manpower, MIScnn could be overtaken by MONAI in the future in terms of cutting-edge feature comparability whereby MONAI would be a worthy successor based on the shared philosophies and similar implementation structure. Nevertheless, MIScnn is a powerful framework for high-performing MIS, being a worthy equivalent to current large-scale frameworks, and still contributing to the state-of-the-art in the field.

Further Development

During the past years after the release and growth of MIScnn, the Author could gain experience in how to successfully maintain a larger open-source project. As evaluated in Chapter 4.2.4, the proposed framework obtained a strong community which also resulted in 4 up to 8 open issues per month. Whereas the topic of some issues were feature requests and bug reports as expected, a large number of issues revolved around complex MIS related questions and specific methods for medical imaging like resampling of CT or MRI scans. Direct experimental analysis of the background profiles of the issue creators in GitHub revealed that MIScnn is mainly utilized by deep learning experts in general computer vision and clinical experts with only limited experience but notable interest in computer science. This resulted in lessons learned on how to efficiently provide continuous support of a framework due to the significant time consumption

requirement in bug fixes as well as in detailed explanations for complex MIS concepts. Furthermore, because of the required expertise in deep learning based MIS for providing reliable support and further software updates, it was noted that the potential to outsource the framework maintenance to undergraduates (student assistants) was only limited. Therefore, the Author started structuring the issue system by keeping informative discussions providing frequently asked questions open for anyone to read as well as highlighting these issues as informative for beginners.

The active MIScnn development is currently focused on multiple key features: Extend preprocessing and data augmentation methods, implement more efficient patch skipping techniques instead of excluding every blank patch (e.g. denoising patch skipping), and implementation of an open interface for custom preprocessing techniques for specific image types like MRIs. Next to the planned feature implementations, the MIScnn roadmap includes the model library extension with more state-of-the-art deep learning models for MIS. Additionally, an objective comparison of the U-Net model version variety is outlined to get more insights into different model performances with the same pipeline. Community contributions in terms of implementations or critique are welcomed and can be included after reviewing. Currently, MIScnn already offers a robust pipeline for medical image segmentation, nonetheless, the Author still plans to regularly update and support the framework in the future.

8.1.2 Scientific Impact and Contributions

One of the key goals of the proposed framework MIScnn is the standardized building of MIS pipelines instead of continuous pipeline reimplementations from scratch in order to strengthen reproducibility and simplify experimentation. MIScnn has been not only utilized in in-house research but also widely in various studies around the globe. As concluded in Chapter 4.2.4, MIScnn has gained substantial popularity with more than 150,000 downloads as well as more than 300 GitHub Stars. This makes it one of the most popular frameworks in the field of MIS as well as representing the state-of-the-art of MIS pipelines [467, 468, 471]. Therefore, the Author concluded that MIScnn provided major contributions to the field in terms of the current state-of-the-art as well as standardized application of MIS in research. The framework considerably contributed to the progress of clinical decision support by paving the way for standardized as well as maintainable MIS pipelines which can now be reliably integrated into clinical workflows.

Analyzing in which way and for what research MIScnn was utilized is complex due to there are only limited options to efficiently track projects using the framework. However, through publications that cited the MIScnn manuscript [49] as well as being indexed in the literature databases Google Scholar [472] and PubMed (MEDLINE) [473], a selection of 25 studies was obtained for further evaluation and were categorized in the following groups: Deep learning research, application research, clinical research, research using MIScnn as comparison, and in-house research. The following subchapters provide a brief overview of these projects.

Deep Learning Research

Studies labeled as deep learning research consist of research aiming to develop novel methods like loss functions or architectures. In this context, MIScnn is utilized as an experimentation framework for quick testing of novel methods, which can be integrated into the framework without efforts through the open interfaces, and to compare outcomes of different methods but with an identical setup.

The team of Yeung et al. [290, 291, 474–476] from the Department of Radiology at the University of Cambridge heavily utilized MIScnn for their research. In their first study based on MIScnn, the authors developed the novel loss function Unified Focal loss which is a generalization of the Dice Similarity Coefficient and cross-entropy based losses [290]. The novel loss function focuses on handling class imbalance and outperformed common loss functions in five experiments discussed in this study [290]. They continued their loss function research by calibrating the DSC [475] and studying boundary uncertainty [476]. In the DSC calibration study, they developed an effective extension of the DSC loss called the DSC++ loss which selectively modulates the penalty associated with overconfident as well as incorrect predictions [475]. In the boundary uncertainty study, Yeung et al. [476] incorporated boundary uncertainty into the DSC loss and achieved consistently improved performance through the more accurate reflection of the segmentation process, being robust to segmentation errors, as well as better generalization. Both loss functions were validated on multiple biomedical imaging datasets demonstrating significant performance improvement [475, 476]. In a follow-up study, the authors analyzed attention mechanisms by including dynamic boundary-based losses into their Unified Focal loss and integrated a Focal layer into the neural network architecture attention modules [474]. Furthermore, they developed a heuristic to assess the optimal attention strength (Focal parameter) specific to the model and dataset [474]. Their method was able to achieve the best results in terms of performance and efficiency in three experiments [474]. Based on their findings, they continued to work on developing a novel neural network architecture that incorporates advanced attention mechanisms. Yeung et al. [291] presented the Focus U-Net which combines efficient spatial and channel attention into a focus gate utilizing a tunable focal parameter for background suppression. The novel architecture was applied for automatic polyp segmentation during colonoscopy and outperformed state-of-the-art results across five public datasets [291]. Next to the Cambridge studies, Jumutc et al. [477] from the Riga Technical University developed a novel multi-path U-Net architecture by introducing individual receptive field pathways to the U-Net. In comparison with common U-Net variants, the authors achieved considerable improvements in segmentation as well as generalization for cell and colony-forming unit image segmentation [477].

Application Research

Studies labeled as application research consist of implementations using the MIScnn framework for training and prediction on a medical imaging dataset. In this context, the focus of the study

is on the application and achievable results on the specific dataset. Often exclusively the available features provided by MIScnn are utilized for the study.

Affane et al. [478] from Clermont Auvergne University performed extensive experimentation on segmenting liver vessels in CT scans with different architectures as well as input shapes (thick slices, patches, and full volume). Their experiments revealed that full volumes achieve in 2 out of 3 architectures the best performance, whereas the MultiRes U-Net achieved the best performance overall with the thick slice input by a DSC of 0.880 [478]. Based on the kidney tumor segmentation approach by the Author on the KiTS19 dataset (described in Chapter 4.3), Jimin Heo from South Korea without an affiliation participated in the KiTS21 challenge hosted by the MICCAI conference 2021 [224, 479]. He setup a 3D U-Net model trained with the sum of the focal and DSC loss which was able to achieve a DSC of 93.69 for kidney, 78.83 for mass (tumor and cyst), and 75.03 for tumor [479]. In the external evaluation through the KiTS21 challenge, Jimin Heo achieved rank #14 [479]. Ogreen et al. [480] from the University of Sibiu aimed to demonstrate the strong performance of low-resource architectures, nowadays, in order to open the way for more researchers with limited hardware to employ and use deep neural networks. The authors setup a 3D U-Net with an input size of 144x144x64 pixels for segmenting the esophagus, heart, aorta, and trachea in thorax CT scans from the SegTHOR challenge [480, 481]. The pipeline demonstrated a strong performance with DSCs of 0.861, 0.945, 0.909, and 0.935 for esophagus, heart, trachea, and aorta, respectively, which resulted in rank #8 in the challenge [480]. Besides deep learning research, Yeung from the University of Cambridge [482] also published an application study in which he built a pipeline for gastrointestinal polyp and instrument segmentation. His pipeline utilized an attention U-Net combined with transfer learning through a ResNet152 encoder pre-trained on ImageNet [482]. The developed pipeline was able to achieve a DSC of 0.872 for polyp and 0.944 for instrument segmentation [482]. Ruijter et al. [464] from the Eindhoven University of Technology developed a model for segmenting the lumen-wall boundary of healthy central and peripheral vessels in ultrasound imaging in order to improve the diagnosis of arterial disease and intervention planning. In cooperation with the Catharina Hospital in Eindhoven, the authors collected 36,153 ultrasound images and deployed multiple U-Net variants [464]. The pipeline showed strong segmentation capabilities in which the MultiRes U-Net achieved the highest performance with a DSC of 0.93 [464].

Clinical Research

Studies labeled as clinical research consist of implementations that aim to use the MIScnn framework in a clinical environment or workflow instead of an exclusive application in an external research lab. In this context, the focus of a study is on utilizing MIScnn to obtain a state-of-the-art MIS model as a method in a multi-step process of a clinical predictor or workflow.

Bjornsson et al. [466] from the University of Iceland developed an automated proximal femur segmentation pipeline in order to later integrate it into their screening method for hip fracture

in the context of osteoporosis. In a collaboration with the Icelandic Heart Association, they were able to collect CT scans of the proximal femur from a cohort and achieved a DSC of 0.990 as well as a Hounsfield Distance (95) of 0.981 mm [466]. A follow-up study by the group of Bjornsson et al. [465] refined the previous model and continued working on postprocessing in order to be fully automated without the need for any manual intervention. They validated their pipeline on 1,147 proximal femur CT scans collected by the Icelandic Heart Association, again, which resulted in a DSC of 0.990 ± 0.008 and a Hounsfield Distance (95) of $.999 \pm 0.331$ mm demonstrating strong accuracy as well as robustness [465]. Bjornsson et al. [465] were able to prove that their method is equivalent to the current best method in the field but faster and without requiring human interaction. Furthermore, the authors test-wise integrated the developed segmentation model into their clinical finite element analysis workflow for hip fracture risk prediction and achieved highly correlating predicted femoral strength compared to using manual segmentation (coefficient of determination by 0.986) [465]. Bjornsson et al. [465] concluded that the pipeline has strong potential for becoming a clinically viable option in at-risk patients screening for hip fracture susceptibility. Stacke et al. [463] from the Linköping University and Stanford University developed the novel automated method Correlated Feature Aggregation by Region (CorrFABR) for aggressiveness classification of renal cell carcinoma (kidney tumor, short RCC) by analyzing features obtained from CT scans correlating with features identified in pathology images. The authors describe their method in three steps [463]: “(1) *Feature Aggregation where region-level features are extracted from radiology and pathology images, (2) Fusion where radiology features correlated with pathology features are learned on a region level, and (3) Prediction where the learned correlated features are used to distinguish aggressive from indolent clear cell RCC using CT alone as input.*” For feature extraction, the method required segmented regions of kidneys and tumors in the CT scans for which the kidney tumor segmentation model of MIScnn (described in Chapter 4.3) was applied [463]. Stacke et al. [463] concluded the CorrFABR pipeline achieved a classification increase from an F1-score of 0.68 to 0.73 compared to radiology features alone demonstrating the potential of incorporating pathology disease characteristics in RCC aggressiveness classification. Müller et al. [461, 462] from the University Medical Center of the Johannes Gutenberg University Mainz utilized MIScnn to assess the splenic volume (SV) in CT scans for analyzing SV as a potential prognostic factor for patients with hepatocellular carcinoma during immunotherapy like transarterial chemoembolization. The authors analyzed the prognostic factor potential of SV by measuring differences between immunotherapy treatment initiation and follow-up [461]. The segmentation of the SV by MIScnn revealed discrepancies in 4% (2 cases) of samples which were corrected by consensus reading [461]. The authors concluded that high SV prior to and during treatment was a significant prognostic factor for impaired outcome [461]. In a follow-up study, the authors were able to retrospectively collect 327 cases which were divided into 100 cases for training and 227 for validation [462]. The model achieved a DSC of 0.96, and 223 segmentation masks were visually approved whereas four cases needed manual segmentation correction [462]. Afterward, Müller et al. [462] observed significant correlations between SV and survival as well as risk of hepatic decompensation concluding that automated SV assessment showed superior survival prediction

compared to current two-dimensional surrogates for splenic size and is a promising imaging biomarker for integrating into the radiological routine.

Research using MIScnn as Comparison

Due to the standardized and simple application of MIScnn, multiple studies used the framework as comparison defining MIScnn as the state-of-the-art in the field in order to prove the functionality of their novel method. In this context, the focus of a study is on their method in which MIScnn only acts as a representation of the state-of-the-art for comparison purposes.

Singh et al. [483] from the University of Rovira i Virgili developed a pipeline for COVID-19 infected region segmentation on CT scans called LungINFseg similar to the COVID-19 study conducted by the Author in Chapter 4.5. Additionally, Singh et al. [483] proposed a receptive-field-aware module that increases the learning ability of the model through receptive field enlargement and added the novel module into their utilized U-Net-like architecture. The authors compared their method with 13 state-of-the-art segmentation models including MIScnn with comparable hyperparameters [483]. In this comprehensive evaluation, MIScnn and Inf-Net [316] performed equivalently as the second-best method after the proposed LungINFseg [483]. Integration of the LungINFseg architecture into the MIScnn framework would extend the architecture library and offers potential for further pneumonia segmentation studies. Wu et al. [484] from the Shandong University of Science and Technology studied patching techniques in the context of lacking feature information in patch edges resulting in grid-like artifacts in neural network model predictions. The authors developed an exponential-distance-weighted method to decrease uncertainty predictions and remove artifacts [484]. They compared their method with three state-of-the-art patch fusion methods including MIScnn and its patching functionality [484]. Whereas MIScnn utilizes simple averaging of overlapping patches resulting in blocking artifacts, the proposed method demonstrated superior performance measured by DSC and effective blocking artifact elimination [484]. In a follow-up study by Xu et al. [485] (co-author of the initial study) from the Shandong University of Science and Technology, a further advanced weighting function based on the truncated Gaussian function was developed resulting in smoother bias fields. In the corresponding comprehensive evaluation with five state-of-the-art fusion methods including MIScnn, the authors demonstrated further increased performance and effectiveness [485]. MIScnn has the potential to integrate this improved and novel technique in its patch overlap processing strategy. Zhang et al. [471] from Xidian University proposed a novel toolkit for MIA called MEDAS, which is short for MEDical open-source platform As Service. MEDAS provides a full MIA pipeline including pre-/postprocessing, image augmentation, neural network, visualization, and debugging modules which are accessible in a user-friendly interface [471]. The authors demonstrated the functionality of the platform in five case studies ranging from pulmonary nodule detection to multi-organ segmentation [471]. In their publication, Zhang et al. [471] introduced MIScnn as a state-of-the-art deep learning-based medical image toolkit in order to provide comparisons with related work. However, the Author was not able to evaluate or compare MEDAS with MIScnn due to the framework is currently closed-source and the provided online platform is only available in

Chinese. Li et al. [468] from the Helmholtz-Zentrum Dresden-Rossendorf performed a meta-analysis on open-source biomedical image analysis models in order to understand new requirements of publishing neural network models to remain the open-source mentality. They analyzed 50 MIA models including MIScnn and identified various new features for publishing neural network based studies like trained model weights, utilized transfer learning weights, and storage solutions of model weights but also hyperparameters as well as code availability [468]. The authors concluded that there are positive trends in openness but only a fraction of publications make necessary elements available to the research community [468]. Even though MIScnn as well as the Author heavily utilize reproducibility-ensuring actions like publishing model weights to Zenodo [447], the study introduced further methods which could be integrated into future publishing procedures of the Author.

In-House Research

Studies labeled as in-house research consist of internal projects which were or are currently conducted in the Author's affiliated lab. In this context, the focus of a study can range from deep learning research to clinical application in collaboration with the University Hospital Augsburg. The majority of in-house studies were presented individually in detail in Chapter 4.

The focus of the Author was research on standardized and reliable MIS application. This included the continuous enhancement of the proposed framework and providing support for researchers in their studies using MIScnn. In order to demonstrate the adaptability and reliability of MIScnn to provide strategies for other researchers on how to handle complex datasets, the kidney tumor segmentation study was performed [49]. As the study is based on 3D abdominal CT scans which are considerably more complex to process compared to regular 2D imaging based computer vision, the developed MIScnn pipeline was able to successfully prove state-of-the-art functionality. Additionally, MIScnn also showed performance comparability with custom-created solutions even though the proposed pipeline required only a few lines of code. Nevertheless, the Author conducted also studies in the context of application and deep learning research. The general aim of these studies was to further increase the predictive capabilities of MIScnn by analyzing robustness and reliability of MIS models. In the nucleus segmentation study based on pathology imaging, the Author analyzed MIS models which were trained in the context of noise-introduced annotations [486]. The combination of adequate masks and rough bounding boxes including artifacts created a challenging dataset for standardized solutions. Still, it was demonstrated that MIScnn provides a mature and effective API to handle such complex segmentation tasks. Furthermore, the robustness of MIS models were analyzed which were trained on a limited sized dataset. The COVID-19 lung infection segmentation study revealed what MIScnn is capable of training accurate and robust models without overfitting on limited data [110]. Both studies contributed to the field of reliable MIS and advanced the MIScnn hyperparameter configuration as well as pipeline setup for robust model building in future applications. This advancement in standardized application with MIScnn contributes to reliable MIS in clinical research by further decreasing the impact of possible biases and increasing application simplicity.

Besides the already conducted studies, further two promising in-house studies based on MIScnn are currently running but intermediate results were not included in this thesis for overview reasons. XAI is currently an active research topic in MIA to increase trust and transparency of neural network models in clinical applications [126, 343, 398]. However, the majority of XAI methods are developed for MIC [343, 398]. The ‘XAIseg’ project analyzes the usability of popular MIC explainable artificial intelligence methods for MIS. In the context of this study, gradient-based XAI methods of MIC were translated for MIS architectures and implemented in a new MIScnn XAI module which currently consists of Grad-CAM [344], Occlusion Sensitivity [343], and certainty maps generated through the softmax probabilities of each pixel. The project aims to investigate the ability of the translated gradient-based XAI methods for neural network decision explanation as well as to compare these methods to image segmentation outcome visualization like certainty maps. The insights of this project can contribute to the integration capabilities of XAI methods in standardized MIS pipelines as well as their usability in the context of MIS. Another currently running as well as funded project is ‘OCTOLAB’ (OCT-optimised laser therapy for basal cell carcinomas) which is a collaboration between the Author’s affiliated lab, the dermatology department of the University Hospital Augsburg, the Medical Laser Centrum Lübeck, and industry partners for dermatologic laser-based devices. This project aims to develop and integrate an MIS model based on MIScnn to support diagnosis and therapy of basal cell carcinoma. The pipeline, which is currently in development, segments the basal cell carcinoma in optical coherence tomography to determine tumor thickness, pulse length, and repetition rate for later optimal long-pulsed infrared laser therapy outcome. After the project end, OCTOLAB could represent a successful integration of MIS in a clinical workflow as well as medical device to support diagnosis and therapy procedures.

8.1.3 Limitations and further Challenges

Even so, the field of MIS heavily progressed in the last years by focusing on toolkit development for counteracting the lack of reproducibility and for improving standardization [49, 109], the methodology of accurate MIS as well as the integration in clinical workflows for decision support still presents major challenges [32, 38, 39].

Through the utilization of toolkits, needless reimplementations of essential pipeline steps can be avoided. Nevertheless, toolkits like MIScnn still require knowledge of deep learning based MIS for appropriate hyperparameter configuration. To ensure high performance as well as intuitive usability, AutoML frameworks are crucial for clinical application. Therefore, a new type of framework is needed which combines the advantages of toolkits for enabling customization and experimentation for research with the advantages of AutoML approaches for straightforward application. A such novel framework could have the potential to unify the MIS community and entirely standardize deep learning based MIS.

Model robustness based on exposure to other non-annotated conditions in an image is still a huge issue in the field [38, 45, 487, 488]. Often binary segmentation models are developed outside of clinical environments and specifically designed for a single disease type. Thus, these

models lose reliable functionality if confronted with other conditions or features that were not present in the training set. An example of this scenario would be a missing limb in a lung infection segmentation model based on X-ray images. Another example is the presence of multiple conditions in a single image which can be common in elderly patients. To the Author's knowledge, datasets with such rare conditions are currently not available in the literature. Through standardized frameworks like MIScnn, MONAI, and nnU-Net the reproducibility was drastically increased but a substantial lack of usability of MIS pipeline in clinical workflows still defines the research field [32]. To reliably apply MIS models in a clinical workflow and be efficient for clinical researchers, the community has to enhance the current approaches designed for disease-specific segmentation trained only on ordinary cases. Models must be capable to predict reliably and comprehend the unexpected.

The decision process in U-Net based segmentation models can be determined by not only the associated features in the ROI but also by disjunct features in the image. The application of XAI methods to clarify opaque decision-making processes is still limited in the field of MIS [489]. Development of segmentation-based as well as translation of classification-based XAI methods is needed to establish transparent MIS models.

8.2 Advancements in Medical Image Classification

Equivalent to MIS, the automated classification of medical images presents a significant challenge due to the processing complexity of medical images induced by technical noise as well as strong variance, and due to the complexity of the distinguishing task which can only be done accurately by well-trained experts [10, 13, 194]. Despite these challenges, numerous application studies of modern deep learning based MIC pipelines proved excellent performance capabilities which is why the integration into clinical workflows is an active research topic and aspired by clinicians to be utilized as clinical decision support systems [9, 24, 35, 39, 52, 62, 442]. Even so, there is a large number of applications conducted in the last years, the majority of MIC implementations revealed to be ‘island solutions’ lacking reproducibility, generalizability, and reusability [32, 35, 39, 45, 49]. In the last three years, these limitations of translational application were noticed and pointed out by clinicians which asserted the need for robustness and standardization in the field of MIC [32, 38, 39, 458]. This is why the research focus of the field shifted from developing the best-performing model on lab data to clinical usability and integration into clinical environments.

Analyzing the literature of recently published implementations, application pipelines utilized more frequently open-source strategies like a Git repository and software documentation apart from the publication manuscript resulting in a significant improvement in reproducibility and transparency [352, 384, 468]. Furthermore, application studies focused more often on optimizing the processes apart from the neural network model like annotation, efficient training strategies, and hardware [31, 189, 263, 490–494]. Apart from research publications, an important indicator for the shifted research focus is the submission requirements for competitive MIA challenges which represent the current state-of-the-art and are commonly hosted by clinicians. In the last years, participation in these challenges has become notably stricter through the requirement of Docker submission [389, 495–497] in which the implemented MIC pipeline has to be fully reproducible and reusable in a virtual container [42]. These requirements on submissions improved reproducibility by increasing implementation transparency as well as reliable application. Still, this advancement does not solve the general lack of reproducibility in the field or the issue of missing standardization. These organized challenges revealed that such strategies claim time-consuming support by experts in order to successfully setup the submitted containers [498]. Adaptations of such implementations require invariably deep learning experts. To reduce the required expert knowledge and allow simple integration as well as usability for IT personnel or clinical researchers, standardized frameworks and AutoML solutions are highly needed in the field. For these reasons, multiple toolkits for MIC have been developed, recently [98, 175]. One of these is the proposed framework AUCMEDI which is discussed in the next subchapter in detail.

Analogous to the field of MIS, the current MIC research can be categorized into the three major subfields of application studies, robustness enhancement research, and deep learning research. In application studies, researchers apply MIC pipelines from the literature on either local patient data or more commonly on public datasets in order to demonstrate strong performance

capabilities [263, 491, 496]. For robustness enhancement research, studies in MIC focus on various aspects around the deep learning architecture like preprocessing, loss functions, training strategies, transfer learning, and ensemble learning, but also include adversarial attacks to identify model weaknesses like hidden biases or overfitting [189, 263, 407, 493, 499]. However, comparing the deep learning research with the field of MIS shows that the field is not centralized around a single architecture and is determined significantly by architectures developed for general image classification in computer vision [329]. The selection of an architecture is commonly conducted by utilizing one of the current top-performing architectures measured on the ImageNet dataset [76, 83]. Traditionally, these architectures have been based on a CNN design which is still the most used principle for architectures, today [82, 95, 500]. However, ViT demonstrated superior performance in 2020 resulting in a new era of MIC research using ViT-based architectures [94, 500]. Nevertheless, the CNN-based ConvNeXt [95] released in 2022 demonstrated even higher performance than the ViT which is why the optimal architecture selection is currently highly debated in the field. Due to the common usage of computer vision architectures in MIC, hyperparameters like the input size of an image into the model play an essential role in pipeline design and are often one of the few differences between top-performing pipelines in MIC challenges [159, 173, 389]. This also strengthens the current hardware requirement for building top-performing MIC pipelines. In contrast to directly using computer vision architectures, specific MIC architecture research is often focused on further improving popular architectures by handling limited dataset sizes or integration of semi-supervised strategies which allow training with less annotation effort by clinicians [492, 494, 501].

Another up-rising topic in the field of MIC is XAI. Neural network models provide powerful prediction capabilities but are computed, in particular for image classification, by opaque processes without providing insights into which region or content of the image determined the prediction of the model [126, 398]. For the integration of MIC models into clinical workflows, understanding the decision-making process of a novel CDS tool is crucial for establishing trust in its capabilities [126, 343, 398, 502]. This is why XAI methods like Grad-CAM [344] are frequently used in recent application studies [206, 384, 402, 489, 503]. However, the generated insights are variable depending on the used XAI methods. Popular XAI methods are gradient-based methods [344], layer-wise relevance propagation (LRP) [504], and GAN based methods like counterfactual explanations [505]. Even so, the insights generated by XAI methods are strongly promoted in the general computer vision community, the level of real usefulness of these methods for clinicians is still highly discussed and an active research topic [343, 502].

8.2.1 The proposed Framework: AUCMEDI

The goal of the implemented framework AUCMEDI is to provide automated classification of medical images with state-of-the-art preprocessing, image augmentation, neural network model architectures, training strategies, ensemble learning, and explainable artificial intelligence. The proposed framework offers a high-level API as a toolkit for deep learning experts as well as standardized clinical research, and AutoML for straightforward application as well as

integration into sensitive IT infrastructure. The modular but intuitive structure of AUCMEDI allows rapid setup of a high-performing MIC pipeline for a custom dataset with just a few lines of code. AUCMEDI is highly adaptable and configurable due to its interfaces to any core module as well as direct compatibility with TensorFlow architectures.

One of the key aspects of AUCMEDI is the open-source development and continuous deployment procedures to ensure usability and high code quality. The framework incorporates the philosophies of user-friendliness, modularity, extensibility, and applicability which allows researchers to setup complex MIC pipelines without being experts in the field of deep learning. AUCMEDI offers high-quality resources through the central website of the software acting as a project hub, the extensive documentation of the API and AutoML module through an automatically created wiki based on docstrings, multiple examples as well as tutorials, package stability ensured by continuous integration, availability ensured by continuous delivery, and utilization of contributor-friendly coding conventions. As documentation is one of the key drivers for the current lack of reproducibility in machine learning [40–42, 44, 351], AUCMEDI is able to considerably contribute to facilitating reproducibility in MIC. The package availability as Python API, CLI, and Docker container guarantees application in varying environments and use cases ranging from experimentation to maintained integration into workflows. This multiplicity of open-source resources represents an outstanding feature compared to other solutions in the field and validates the high quality as well as stability of the framework. The proposed universal framework is an important step in contributing to standardized MIC and practical application which was already possible to demonstrate in multiple in-house studies.

Even though toolkits for standardized applications are a huge contribution to the field and allow deep learning experts to build state-of-the-art pipelines without the need for the time-consuming reimplementations of established methods, clinical researchers are still not being able to utilize these pipelines due to the required expertise in the field of deep neural networks [32]. Such knowledge is often essential for simple adaptations to individual environments, integration into workflows, or maintenance over the years. To solve this issue, AUCMEDI offers a powerful AutoML module that allows clinical researchers direct application and integration into workflows. Designing such AutoML approach is difficult due to the complexity of medical imaging, high variance in patients, data representation in 2D as well as 3D, and the diverse modalities with different intensity value scalings [19, 124, 182]. Nevertheless, based on the dynamic API of AUCMEDI, it was possible to implement such automated classification. Furthermore, the virtualization of the AutoML module through Docker allows simple integration into sensitive IT infrastructures like clinical workflows. To the Author's knowledge, there is no AutoML framework for MIC apart from AUCMEDI which is capable of providing state-of-the-art as well as competitive performance. Thus, the proposed framework considerably contributes to standardized application research and paves the way for the integration of MIC pipelines into the clinical routine.

For enabling efficient research, deep learning experts are dependent on dynamic toolkits with open interfaces to develop novel methods and clinical researchers are dependent on AutoML for application as well as integration into workflows. Due to the majority of MIA pipelines focusing on the functionality as either a toolkit or an AutoML tool, users of the MIA community are often divided between multiple frameworks which correspond to their application type. For an attempt to unify these application types in order to further increase standardization as well as interdisciplinary collaboration between deep learning and medical experts, a framework needs to provide both functionalities. AUCMEDI is able to combine both types of usage for building MIC pipelines. This results in a powerful tool providing the freedom for customization as well as expandability with a toolkit to support methodology research and simple usability with an AutoML framework to support clinical application research.

One of the major features of the proposed framework is the simplicity in using it as a toolkit which does not require extensive deep learning knowledge but still represents the structure of an MIC pipeline without the need of implementing fundamental methods. Equivalent to MIScnn, it was possible to utilize AUCMEDI in education for medical information science students. The broad usage by students not only demonstrated its simplicity yet effectiveness in building functional MIC pipelines but also provided essential insights for students into the concepts of modern MIC.

Through the proven effectiveness of AUCMEDI based on outstanding predictive capabilities demonstrated in competitive challenge participation, wide adaptability to diverse imaging modalities, AutoML functionality, comprehensive resources, and simplicity in application, the Author concluded that AUCMEDI is a powerful tool for state-of-the-art medical image classification which contributes to the standardization in the field. The Author hopes that AUCMEDI will further help migrate MIC models from the research labs into clinical applications.

Related Work and Limitations

Image classification is a popular research field in computer vision which is why multiple approaches for standardized toolkits in general image classification have been made [506–508]. However, base frameworks like TensorFlow [72] and PyTorch [71] are still by far the most widely used tools for building general image classification pipelines in the context of research [30, 263, 469]. Similarly to general image classification, the majority of pipelines for MIC research or application are currently implemented with base frameworks. Nevertheless, through the AutoML mentality and the increasingly required expertise in deep learning, more toolkits as well as AutoML frameworks for general image classification have been published in the last three years. Popular examples are Google Cloud AutoML Vision by Google LLC [506, 509] (2017), Ludwig by Molino et al. [507] (2019), and AutoKeras by Jin et al. [508] (2019). These frameworks established large communities and contributed to the more straightforward application of deep learning based image classification models. In the context of MIC, it is possible to utilize such general image classification frameworks for medical images in regular

imaging formats like JPG or PNG. Especially medical images, which are generated by visible light imaging and are similarly encoded like regular digital photography, can be processed by general image classification frameworks. However, the field of medical imaging analysis contains various imaging modalities which require specific processing procedures. Medical images often contain more information than a regular image in computer vision which is either encoded in the image representation itself or as metadata [19]. Examples of such additional information are HU in CT scans or 3D images. In computer vision, 3D image classification or class imbalanced datasets are niche research fields and not sufficiently enough represented or even possible to handle in general image classification frameworks. Often it is technically possible to convert medical images into the required formats for general image classification frameworks, but not advisable due to the drastic information loss and low performance which is the result of lacking adequate preprocessing methods, architectures, and training strategies for medical images [510]. On account of these issues, MIC pipelines have been still developed with base frameworks.

For medical image classification, toolkits or frameworks for standardized pipeline building are rare. One of the first established frameworks was NiftyNet by Gibson et al. [43] in the year 2018 which is introduced in Chapter 8.1.1. The framework is also based on TensorFlow similar to AUCMEDI but is implemented as a configurable application instead of an API [43]. Furthermore, NiftyNet significantly lacked the capabilities to create state-of-the-art MIC pipelines which are compatible in performance with custom-developed pipelines. Subsequently, NiftyNet was less utilized in deep learning research or clinical application resulting in the deprecation of the project in 2020 [511]. This is why the Author started to develop AUCMEDI in 2020 which combined the capabilities for building state-of-the-art MIC pipelines through its intuitive Python API but also allows rapid application with its AutoML module. Since 2020, a handful of frameworks for standardized MIC were developed [43, 98, 175, 218]. MONAI [175] is the most prominent framework according to its popularity evaluated in Chapter 3.4.3 and is introduced in Chapter 8.1.1. Comparing the features of MONAI and AUCMEDI reveals that the two frameworks share the majority of key methods as well as fundamental philosophies. Both frameworks provide an extensive library of preprocessing methods, on-the-fly image augmentation, training strategies, neural network architectures, and evaluation procedures. In terms of open-source mentality, MONAI and AUCMEDI both advocate as well as employ similar concepts and have the same goal to simplify deep learning based MIC application. MONAI was able to successfully build a large open-source community around its package, whereas the community around AUCMEDI is significantly smaller but continues growing considering its recent publication on GitHub. Although a comparison of community sizes or scientific impact will be interesting after a manuscript publication of AUCMEDI, it is most likely that MONAI will be a more prominent and larger project in the future due to the massive funding and manpower difference compared to AUCMEDI. This can be also observed in the latest efforts of MONAI for providing software solutions to assist in other relevant processes around the MIC model implementation like the annotation as well as deployment process [175]. Still, for the development of state-of-the-art MIC pipelines,

AUCMEDI provides advanced techniques which are not featured in MONAI. The most crucial feature in performance-boosting missing in MONAI is ensemble learning. Whereas MONAI only supports a limited usage of Bagging via cross-validation and Augmenting, AUCMEDI offers simple one-line functions for creating complex but highly configurable ensemble learning pipelines including Stacking, Bagging, as well as Augmenting. In summary, MONAI centers around the toolkit concept as providing only additional methods for native PyTorch implementations and still requiring extensive knowledge in the field of deep learning. In contrast to this concept, AUCMEDI focuses on simplification of the development process to reduce the required knowledge in deep learning but still enables research possibilities as well as competitive high performance. This benefit opens MIC to a broader interdisciplinary community and offers clinical researchers the possibility to conduct state-of-the-art research by themselves without the need for years of experience in the field of deep learning.

There are also multiple differences between MONAI and AUCMEDI. The most fundamental difference is that MONAI is implemented in PyTorch whereas AUCMEDI is based on TensorFlow. The increasing popularity of PyTorch in deep learning research was already discussed in relation to MIScnn in Chapter 8.1.1 [470]. In summary, even though both frameworks currently provide the same functionality, novel research methods could be not available in TensorFlow in the future and impact the competitiveness of AUCMEDI. However, the field of MIS can be defined as a more complex or niche subfield. By the way of comparison, it is smaller than other subfields and shares less similarities to general image segmentation resulting in more independent research. In contrast, the field of MIC is highly determined by general image classification in terms of neural network architecture. As the community of TensorFlow is by numbers still larger than the PyTorch community and remarkably active in general computer vision, recent advancements in general computer vision are rapidly available in TensorFlow, as well. A good example of this could be observed during the release of the groundbreaking ConvNeXt by Liu et al. [95]. The image classification architecture was published as PyTorch implementation in July 2022 and was available in TensorFlow less than two months later (September 2022) [72]. This demonstrates that important contributions to the field, even though the methods are originally implemented in PyTorch, are promptly available in TensorFlow, as well, and will likely not impact the method competitiveness of AUCMEDI in the future.

Apart from the comparison with MONAI, AUCMEDI offers a unique and innovative feature that is exceptional in the field of MIC: Automated machine learning [31]. To the Author's knowledge, there is no other AutoML framework for MIC which is capable of achieving state-of-the-art and competitive performance. Therefore, the proposed framework shapes the field of automated MIC for deep learning research and clinical applications through the avoidance of previously required custom pipeline implementations as well as deep learning expertise. Despite that, two AutoML frameworks were released in the last two years which provide currently not sufficient state-of-the-art methods for high-performing MIC but built an active and growing community [218, 223]. The Nobrainer framework by Kaczmarzyk et al. [218] (2022) focuses on 3D processing of medical images through deep learning based MIA. Even

though the framework lacks state-of-the-art architectures like DenseNet [90] or ViT [94] as well as training strategies like transfer learning or ensemble learning, it provides a large variety of other methods for MIA like segmentation, generative modeling, knowledge representation, and Siamese representation learning [218]. Another recently developed AutoML framework is ivadomed by Gros et al. [223] (2021). This framework is a promising tool for automated MIA capable of modern segmentation, classification, as well as object detection [223]. Furthermore, it supports complex pipeline designs based on ensemble learning like cascading networks [223]. However, it still lacks methods for competitive high-performing image classification like the ViT [94] and supportive techniques like XAI. Still, ivadomed is a promising tool with a rapidly growing community. Comparing the architecture libraries of the two frameworks with AUCMEDI reveals that the proposed framework offers a more extensive library with recently published cutting-edge models in image classification. Though, AUCMEDI still lacks support for the novel ConvNeXt [95] (2022) and 3D ViT [512, 513] (2021). In addition, the two frameworks are capable of handling multiple MIA tasks, whereas AUCMEDI is specialized in only MIC. Nevertheless, AUCMEDI is currently the only framework offering state-of-the-art as well as competitive MIC methods which had been demonstrated in the previously presented challenge participation studies.

Further Development

Through the continuous development of MIScnn during the past years, the Author was able to gain experience in how to successfully maintain an open-source project. This experience contributed to the development of AUCMEDI resulting in a high-quality implementation based on an elaborated module structure supported by continuous integration as well as delivery processes and code conventions. The development process was conducted with an open-source mentality through proper usage of the issue system as agile development resulting in transparency as well as lower burdens for contributors. The usage of automated documentation deployment based on docstrings allows the maintenance of the extensive wiki without an increase in time cost. The Author plans to facilitate methods for community growth like hosting a workshop at an IEEE or EFMI conference on how to use AUCMEDI for personal research and publishing the manuscript introducing the framework in a peer-reviewed journal.

The active development of AUCMEDI currently focuses on implementing the ConvNeXt [95] architecture as well as further recently developed architectures into the library. Next to the current development, the AUCMEDI roadmap includes the following advancements. It is planned to further extend the AutoML module with an ensemble learning cascading pipeline as well as introduce provenance strategies through logging and advanced model management. Moreover, the integration of layer-wise relevance propagation [504] as a method in the XAI module would provide another method type for explaining the opaque decision-making of neural network models. Community contributions in terms of implementations or critique are welcomed and can be included after reviewing. Currently, AUCMEDI already offers a powerful solution for medical image classification, nonetheless, the Author plans to continue the development, support, and maintenance of AUCMEDI.

8.2.2 Scientific Impact and Contributions

One of the key goals of the proposed framework AUCMEDI is the standardized building of MIC pipelines instead of continuous pipeline reimplementations from scratch. Furthermore, AUCMEDI aims to combine the concept of an extensive but intuitive API for deep learning experts with straightforward usability for interested clinicians and the capabilities of simple integration into sensitive IT infrastructures. Currently, no manuscript about AUCMEDI has been drafted or published in a journal, yet, resulting in only limited community awareness of the package. Also, the framework was only technically referenced in the implementations for the MIC challenges participation but not in the corresponding published articles. AUCMEDI has gained a small reputation with more than 10,000 downloads on PyPI and 10 GitHub Stars. Compared to other frameworks in MIA, these numbers currently imply a rather small community. However, the package was introduced at the annual conference of the German Association for Medical Informatics, Biometry and Epidemiology (GMDS) [514] at the end of August 2022 explaining the growth in interest in the unpublished project. Through the planned publication of an article for AUCMEDI as well as organizing workshops for application tutorials in the future, the Author hopes to make the framework more acquainted within the field of MIC and grow its community. Nevertheless, the proposed framework was already successfully utilized and is still currently utilized in multiple in-house deep learning research, applications, and clinical studies.

AUCMEDI contributes to further progress the research field but also demonstrates unsurpassed performance by an MIC framework determined through MIC challenges participation. Furthermore, the AutoML module represents the current state-of-the-art for automated MIC in the field. Through intuitive and rapid application capabilities, the proposed framework simplifies experimentation and strengthens reproducibility through its robust implementation. AUCMEDI contributes to the progress of clinical decision support by paving the way for standardized as well as maintainable MIC pipelines allowing reliable integration into clinical workflows. The Author concluded that AUCMEDI has the potential to be a prominent solution for standardized and high-performance medical image classification.

The AUCMEDI framework was utilized in multiple in-house studies which are currently being conducted or already published in peer-reviewed journals or conferences [159, 448, 498]. These studies are briefly described in the following section and were categorized into deep learning research, application research, and clinical research. The definitions of these categories are equivalent to the categorization of MIScnn studies in Chapter 8.1.2.

Deep Learning Research

For analyzing the impact of ensemble learning techniques on MIC performance, an extensive study was conducted evaluating Augmenting, Stacking, and Bagging [448]. Furthermore, multiple pooling functions were compared, as well. The study used AUCMEDI to build a universal image classification pipeline for four medical imaging datasets and conducted experiments by measuring the performance variance using different ensemble learning methods

[448]. The outcomes showed that the integration of ensemble learning techniques is a powerful method for MIC pipeline improvement in which Stacking achieved the strongest performance boost [448]. In such study designs, AUCMEDI can be utilized as an experimentation framework for integrating novel methods and comparing outcomes of different methods but with an identical setup.

Application Research

In order to analyze the performance capabilities of AUCMEDI, the Author participated in two large MIC challenges in which predictions are evaluated by external researchers, and the corresponding models from the research labs around the world are ranked based on their achieved performances. In the RIADD challenge (Chapter 5.4) [158, 173], a multi-disease detection pipeline for a complex multi-label as well as highly class-imbalanced retinal imaging dataset was implemented utilizing AUCMEDI [159]. The pipeline incorporated modern deep ensemble learning techniques by combining multiple architectures in a 5-fold cross-validation setup to improve its predictive capabilities. The approach demonstrated highly accurate as well as reliable retinal condition predictions and was ranked as one of the top-performing submissions in the challenge [159, 173]. The pipeline was also included in the challenge organizer's evaluation manuscript which defines the current state-of-the-art for retinal image classification. In the STOIC challenge (Chapter 5.5) [137, 389], a pipeline for COVID-19 infection and severe outcome prediction was implemented utilizing AUCMEDI. The pipeline incorporated state-of-the-art ensemble, transfer, and deep learning techniques for reliable inference. The approach demonstrated excellent infection as well as severity prediction capabilities and was ranked as one of the top-performing submissions in the challenge [389, 498]. In contrast to the RIADD challenge, the STOIC challenge focused more on model reproducibility and reusability which is why the pipelines were required to be implemented in a Docker container. In the recently hosted workshop of the STOIC challenge [498], the winners for the final test set were awarded in which AUCMEDI was able to achieve rank 4. Comparing the top-performing pipelines revealed that the major difference between AUCMEDI and higher-ranking pipelines was the architecture selection. Whereas the AUCMEDI-based approach implemented a 3D DenseNet, the three higher-ranking approaches utilized 3D ViT or a custom-implemented 3D ConvNeXt architecture [498]. Likewise as in the RIADD challenge, the pipeline will be also included in the STOIC organizer's evaluation manuscript which defines the current state-of-the-art for COVID-19 classification. The challenges participation with AUCMEDI demonstrated the excellent performance capabilities and proved that the proposed framework is unmatched in terms of performance competitiveness as a toolkit in which only purely specialized pipelines built with base frameworks can compete.

Next to only image classification challenges, an in-house study is currently conducted with an AUCMEDI-based implementation detecting instance of prostate cancer in MRI scans. The PI-CAI challenge estimates the performance of radiologists around the world and compares its findings with AI solutions [515, 516]. The task is to detect prostate tumors and their likelihood of being malignant prostate cancer [515, 516]. To achieve such instance-based classification, a

combination of MIS through MIScnn and MIC through AUCMEDI is implemented for evaluating the potential of combing the two proposed frameworks.

Clinical Research

The AUCMEDI framework is currently also utilized in two clinical studies. The successful application of the proposed framework for clinical research demonstrates its reliable performance, adaptability for diverse medical disciplines, and effectiveness in terms of integration capabilities into clinical workflows.

In the EKIPRO project, which is described in detail in Chapter 5.6, the collaborated study with the pathology department of the University Hospital Augsburg aims to setup an MIC pipeline in order to compare XAI methods in terms of the quality and information gain for clinicians at the use case of Gleason scoring in prostate carcinoma tissue. In such a clinical study, AUCMEDI is utilized as a tool for the application of state-of-the-art MIC as well as to rapidly generate multiple XAI methods. The generated XAI approaches are visualized and manually assessed by pathologists for evaluating their informative value. Here, AUCMEDI contributes to the clinical study by enabling rapid application and integration of an MIC pipeline into a clinical workflow for research.

In the GrandAID project, the collaborated study with an industrial partner and the dermatology department of the University Hospital Augsburg aims to implement a skin condition classification to provide personalized cosmetics for customers. The research aspect behind this study is to develop a reliable pipeline that is capable of assessing dermatologic features in smartphone-generated photos of human faces. Such a tool would have the potential to be used for rapid automated assessment of dermatology patients which can be utilized as clinical decision support or documentation, but also as triage to identify severe cases prior to an appointment. Here, AUCMEDI is utilized as a state-of-the-art MIC classifier and contributes to the clinical study by enabling reliable application and integration of an MIC pipeline into a practical workflow.

8.2.3 Limitations and further Challenges

The field of MIC has considerably progressed in the last years by focusing on toolkits as well as AutoML development for counteracting the lack of reproducibility and for improving standardization. However, the field presents still multiple critical challenges for standardization and reliable integration into clinical workflows.

ViT started a new era of architectures for image classification by offering an alternative to CNNs. Nevertheless, implementations of 3D ViT [513] are still only described in custom-developed pipelines for general image classification but are not available in toolkits for MIC as well as not widely utilized in applications for MIC [175, 218, 512]. Furthermore, standardized solutions currently neglect the possibility of integrating metadata like patient age and instead focus on pure image-based prediction [175, 218, 498]. However, such metadata contains

relevant information which can significantly improve accuracy as well as model reliability for hard cases.

Equivalent to the field of MIS, model robustness based on exposure to non-annotated conditions in an image is still an issue in MIC [38, 45, 487]. Commonly, classifiers are trained on distinguishing between healthy control samples and patients with a specific condition like pneumonia in CT scans. Despite that, studies revealed such models have the risk of performing catastrophically in clinical applications due to elderly patients often having other active conditions or even multiple conditions [32, 45, 458]. This type of issue critically hampers the progress of integrating deep learning based MIC models into clinical workflows. Although, public datasets which recently been published started offering more noise-introduced control samples. Challenges like ISIC [58] or RIADD [158, 173] included an ‘other’ class containing edge cases, multi-condition presence, and atypical images like blurred as well as noisy image capturing. Such intended introduction of biases in testing data is an effective and endorsed approach for assessing model robustness. Nevertheless, variance in classification datasets to estimate model reliability as well as robustness is a critical part of validating AI solutions. This is why strategies for assessing robustness have to be established as a standard evaluation procedure next to performance assessment in the field of MIC. To enable MIC-based clinical decision support, further research has to be done on how to handle the inevitable prediction uncertainty in deep learning models.

8.3 Advancements in further Research Fields

Building a state-of-the-art pipeline in the field of medical image analysis with a focus on deep learning based MIC and MIS requires knowledge in various other domains. Especially in recent years, the adjacent research fields ensemble learning and evaluation reproducibility became particularly important in high-performing as well as reliable pipelines [32, 44, 109, 200, 338]. In order to offer frameworks for the standardized application of deep learning based MIA pipelines with leading-edge capabilities, it is essential to understand as well as incorporate the concepts and recent methods in these adjacent fields. This is why the Author conducted extensive research in the field of ensemble learning and evaluation reproducibility, which not only contributed to the corresponding research fields but also to the enhancement of the proposed frameworks MIScnn and AUCMEDI.

8.3.1 Ensemble Learning

Analyzing high-performing pipelines for MIS as well as MIC revealed that the most accurate MIA pipelines are heavily based on ensemble learning strategies [187, 189, 400–404]. This is why the majority of novel MIS pipelines are commonly utilizing ensemble learning techniques and representing the state-of-the-art for more efficient training data usage due to the limited dataset sizes in medical imaging or for further performance increase [338–341]. In contrast to the field of MIS, novel MIC pipelines often are single-model approaches solving issues through advanced architectures instead of the application of ensemble learning. This can be explained through the extensive development of novel architectures in general image classification, whereas the field of image segmentation heavily relies on the U-Net for the last 6 years. The centralization of a single architecture in the field of MIS lead to the focus on integrating other techniques like ensemble learning to allow further progress.

The most common strategies for deep ensemble learning are Stacking by combining different model types, Bagging by repeated training dataset sampling, and Augmenting by image augmentation prior to inference. However, the most widely implemented one in general MIA is Bagging through cross-validation sampling [338]. Still, the state-of-the-art of MIS, represented through the nnU-Net framework [109], is a combination of Stacking and Bagging in which models with multiple different input sizes for the U-Net are trained in a cross-validation manner. However, the extent and which ensemble learning strategies are beneficial in deep learning based MIA pipelines has been still an open question. Therefore, the Author conducted two studies analyzing the impact of ensemble learning strategies on classification as well as segmentation performance in MIA (Chapter 6) [448, 449]. Whereas the initial segmentation study was limited to a single dataset and did not analyze Stacking strategies or different pooling functions, the follow-up classification study included comprehensive experiments on multiple datasets, various pooling functions, and all major deep ensemble learning strategies. Both studies revealed significant performance increases by integrating ensemble learning strategies ranging from 0-4% for Augmenting, 1-11% for Bagging, and 1-13% for Stacking (only classification). Based on the classification experiments, the Author also

concluded that the complexity of a dataset determines the possible performance gain through ensemble learning, in which more models based on datasets with more challenging features are able to demonstrate a stronger performance improvement. Nevertheless, through the conducted experiments, it was possible to formulate the following three recommendations: Augmenting can be always applied through its small but effective performance increase without any additional training time. On the other hand, Stacking and Bagging can provide strong performance gains at the cost of major increases in training time. Still, Stacking combined with a metalearner like Logistic Regression is capable to provide robust performance increase without the risk of a possible predictive capability loss. The experiments allowed comprehensive insights into deep ensemble learning impact on performance and contributed to the field of deep learning based MIA pipeline building.

The gained insights in the efficient utilization of ensemble learning were integrated into pipeline development with the proposed frameworks AUCMEDI and MIScnn to further improve robustness as well as performance. In particular, AUCMEDI heavily utilizes automated ensemble learning strategies and demonstrated powerful performance in the challenges RIADD (Chapter 5.4) and STOIC (Chapter 5.5). MIA pipelines based on ensemble learning are commonly manually implemented and island solutions [35, 39, 49]. Thus, the capabilities of integrating ensemble learning methods in standardized frameworks or toolkits are still limited. The Author concluded that even though ensemble learning is a prominent strategy in MIA, frameworks often lack support for it which handicaps standardized integration of the powerful multi-model strategies. To the Author's knowledge, fully automated ensemble learning including Augmenting, Bagging, as well as Stacking with different pooling functions is only supported in nnU-Net [109] and AUCMEDI [98] for standardized application, whereas MIScnn [49] and MONAI [175] provide only limited support for Augmenting and Bagging.

Nevertheless, the field of deep ensemble learning offers still a wide range of open questions and interesting research topics. In the context of opaque decision-making systems, pipelines based on ensemble learning methods represent another challenge for XAI through the increased complexity of using multiple models. However, the combination of distinctive models trained on different subsets of data could also represent a solution for ensuring strong data privacy in hospitals but the capabilities of training AI solutions for research [459]. Utilizing ensemble learning would allow combining the predictive capabilities of anonymized models trained individually at hospitals without the need to centralize patient data [459, 460]. This active research field is called federated learning [459, 460].

8.3.2 Reproducibility of Performance Assessment

In 2016, Monya Baker published a large-scale survey in Nature revealing the presence of a large reproducibility crisis in all major research fields [517]. 6 years after this article, numerous enhancements have been made by researchers as well as journals to assure the reproducibility of studies [518–520]. Nonetheless, multiple meta-analyses concluded that there is a massive drop in reproducibility in the field of machine learning, in particular MIA [41, 44, 63, 487].

Especially the corona pandemic proved that the high availability of AI models in research for potential clinical decision support does not keep its promise for performance neither functionality to assist clinicians [32]. The ultimate goal of AI-based research and the premise of the associated studies is the application in the real world to automate processes or provide information for users [10]. Still, the majority of AI implementations are neither reproducible nor applicable in any real-world scenario [32, 45, 52, 75, 442–444]. This current situation in the field of MIA is critical as clinicians are interested in the application of automated models but can not utilize the implementations from the literature.

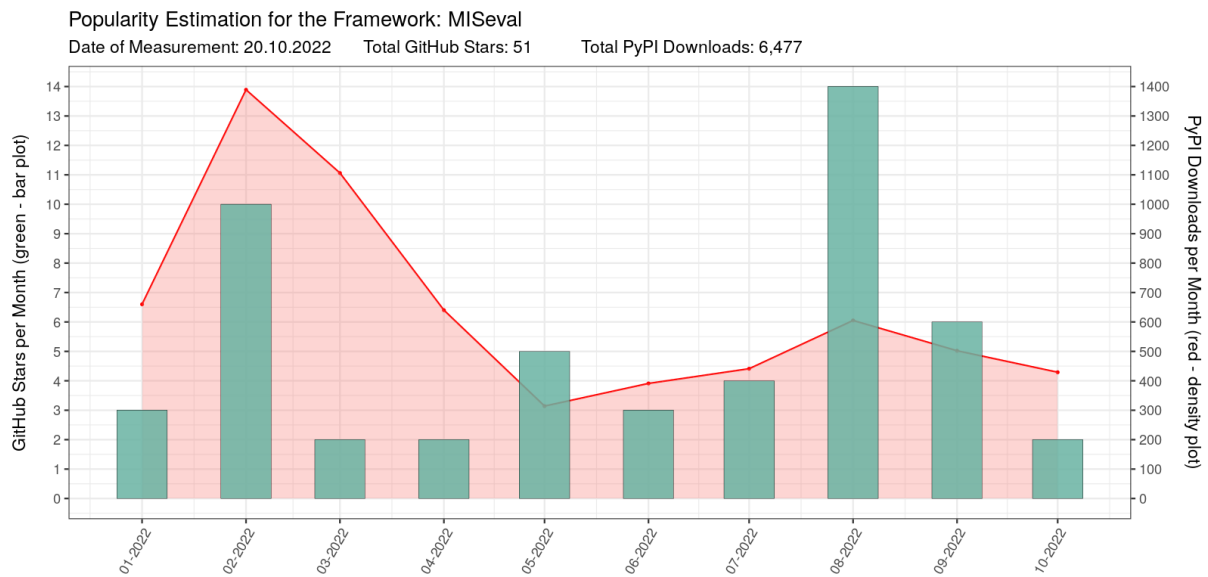


Figure 8.1: Popularity estimation for the MISeval framework.

Next to a lack of standardization of MIA pipelines, studies revealed that statistical bias is one of the key drivers for insufficient reproducibility which is caused by improper metric usage and implementation [32, 44–48]. This is why the Author tried contributing to solutions for establishing evaluation standards to increase reproducibility. Noticing the lack of an open-source framework in Python for standardized evaluation in MIS, the Author proposed the framework MISeval [293]. The proposed metric library for MIS evaluation allows straightforward usage of a large number of metrics without the need for error-prone reimplementations. As illustrated in Figure 8.1, the framework gained notable popularity in the scientific community with over 6,000 downloads from PyPI [364]. Furthermore to standardized evaluation metrics, the Author identified and discussed common pitfalls by proposing a community guideline for robust evaluation in MIS [226]. The recommended usage and understanding of interpretation biases in evaluation metrics is crucial for correct as well as robust model evaluation and improving community standards in the field. The Author hopes that such guidelines will help improve evaluation quality, reproducibility, and comparability in future studies in the field of MIS.

The lack of control samples for evaluating the specificity of MIS models is another key driver for community-accepted overfitting and reusability issues in clinical environments [29, 37, 40, 521–523]. The development of the novel metric MISm [524] for performance assessment in the presence of control samples allows further evaluation insights in proper medical datasets. MISm also lays out the foundation for a confusion matrix based loss function for adequate scoring of predictions on control samples without the need for multi-label encoding. Even so, the proposed metric contributes to robust performance assessment and reliable model building through the potential usage capabilities as loss function, standardized application of MISm for scoring comparison between methods in the literature is currently limited due to its flexible parameter definition.

Incorrect or improper evaluation of MIS is critical due to its potential impact as clinical decision support system or in terms of reproducibility for clinical research [46, 47, 225, 226, 453]. The conducted studies contribute to robust and standardized evaluation procedures to avoid statistical bias and to further increase the reproducibility of MIS models in healthcare. These contributions help in building reliable MIS models, increase the trust of clinicians in AI solutions, and, therefore, accelerate the progress of AI integration into medicine. Nevertheless, more community focus on this issue is needed as reproducibility directly determines the success of AI models in real-world applications.

8.4 Software Integration Process: From University Research to Clinical Application

Because of the outstanding predictive capabilities of AI models for analyzing medical imaging data in recent studies [8, 13, 34–37], AI tools are highly desired by clinicians to be integrated as CDS systems into clinical routines for providing new information, increasing diagnostic as well as treatment decision confidence, and reducing time-consuming processes [24, 25]. However, meta-analyses and application studies revealed that deep neural network based MIA methods lack reproducibility resulting from poor performance evaluation procedures [32, 44, 45, 48]. A major contributor to this issue is the current standard of assessing performance based on public datasets which are already preprocessed, biased, and not reflecting the reality of clinical data [40, 41]. Through pipeline standardization via frameworks, MIA pipelines which are capable of using different data origins have drastically advanced the applicability in clinical environments. This progress allows improved performance assessment through the clinical application on practical medical data in realistic use cases and additional qualitative evaluation through physicians. Still, the integration of AI models from the research lab into a clinical environment is a complex and challenging task [9]. In this chapter, the Author briefly outlines the gained experience in clinical applications at the University Hospital Augsburg. The Author hopes that these insights contribute to performance assessment reliability, help fellow researchers in the study design for AI-based clinical applications, and, consequently, facilitate the progress of migrating MIA models into clinical routine.

8.4.1 Challenges of Integration

The integration process of AI models into clinical environments reveals various challenges which have to be regarded in study design and are summarized in this subchapter.

The coronavirus pandemic demonstrated that attempts by numerous clinicians to utilize AI models as CDS systems failed due to low reproducibility, reusability, as well as generalizability in the field of MIA [32, 35, 38, 39]. The lack of standardization in implementations, as one of the major contributors to the reproducibility issue, has been considerably improved in the last years by the development of standardized frameworks and toolkits whose usage should be strongly considered to ensure reproducibility in other clinical sites. Nevertheless, custom-implemented pipelines are still prevalent in the literature increasing drastically the integration complexity [42, 43]. Even though toolkits are utilized for pipeline building, comprehensive knowledge in deep learning and system administration is often required to integrate as well as maintain an MIA pipeline due to the high complexity and sensitivity of IT infrastructure in hospitals. The complexity of integration into clinical workflows is further increased by extensive usage of commercial software in hospitals without any open interfaces for 3rd party plugins to connect an AI model with the established systems that in-house staff already know to operate. This issue results in the need of implementing custom tools aggravating the enigmatic heterogeneity of software tools in the IT infrastructure of a hospital. An additional requirement for modern MIA which has to be considered for application is the hardware

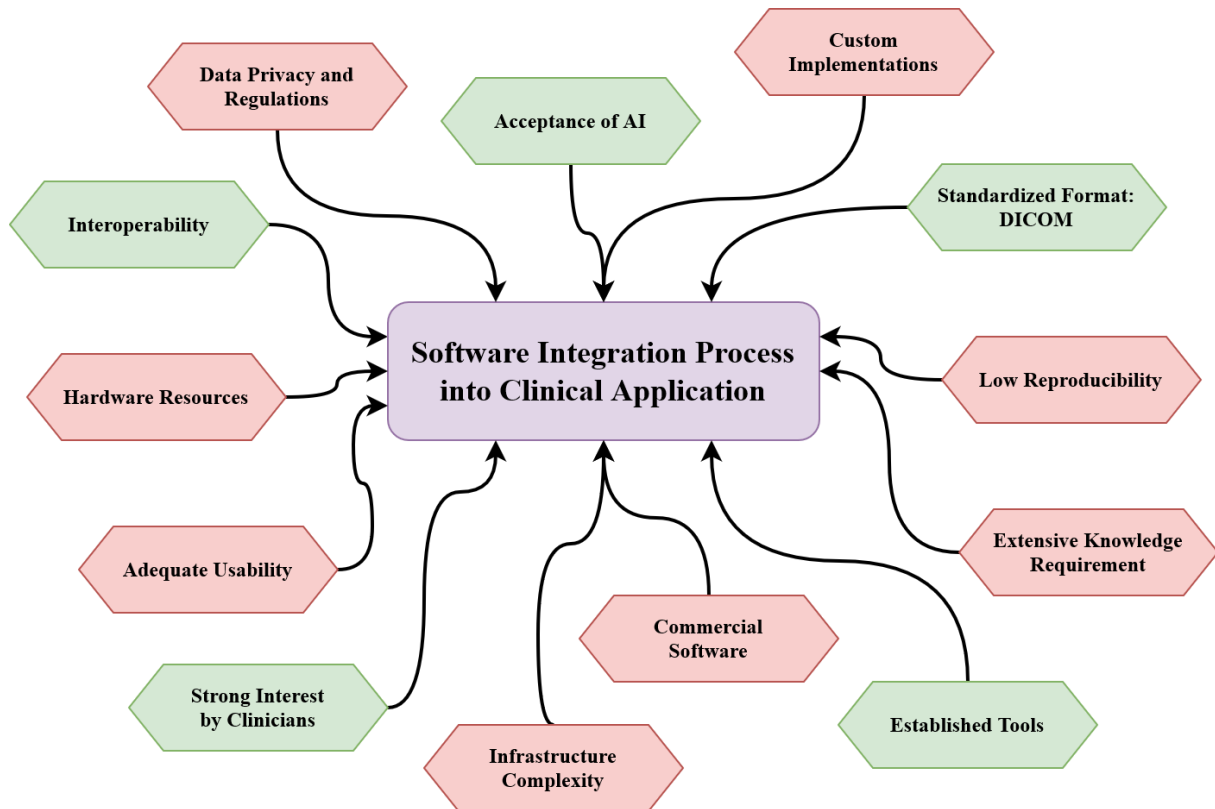


Figure 8.2: Chances (green) and challenges (red) of the software integration process for AI in medical imaging.

resources. As already described in Chapter 2.3.3, state-of-the-art deep neural network architectures require expensive GPUs, in particular for training new models. Therefore, large-scale usage of AI models has to be done in professional data centers or clouds, whereas a GPU workstation can be adequate for research experimentation by small teams. Accordingly, the regional, national, as well as transnational (like European Union) data privacy laws and regulations for medical data have to be guaranteed which can prohibit the usage of external data centers from large vendors like Google, Microsoft, or Amazon. Furthermore, collaborated projects often need contractual agreements for sharing data in which data privacy has to be validated by a data protection officer or ethics committee. Apart from challenges originating from the AI model and infrastructure, the organizational effort of a clinical application study must not be underestimated. The resources needed, especially in terms of personnel, are often substantial due to necessary expertise in medicine, data privacy as well as proper study design, deep learning, and system administration. Furthermore, the required time to pass the mandatory instances for conducting a clinical study is also often underestimated. Lastly, the adequate representation of the predicted features and the trust of clinicians in the AI model are essential for the success of the integration.

8.4.2 Chances by Interoperability in Medical Imaging

The integration of AI models for MIA also unveils chances based on the existing interoperability in medical imaging.

As one of the only disciplines in medical information science, medical imaging data is encoded in the standardized format DICOM which is established all around the globe [54, 127]. In comparison with other data exchange standards like HL7, DICOM is utilized exclusively for sharing medical imaging data. The majority of data input as well as output in medical imaging tools can be covered through a single DICOM interface reducing complicated implementation of multiple custom connectors which drastically reduces integration complexity. Furthermore, the DICOM format supports the encoding of computer-generated predictions [127] which is why the majority of existing software for medical imaging can integrate predictions from MIA models. Such a standardized and established format considerably decreases the complexity of integration in medical workflows as well as infrastructure and allows advanced implementation of clinical applications building upon already established tools. Another advantage of MIA is the prevalence of computer-assisted tools in clinical routines for disciplines associated with medical imaging. Radiologists as well as pathologists push forward the integration process of modern tools to automate time-consuming processes or incorporate decision support systems [8, 9, 368, 525]. The ambition of these two disciplines to use novel methods is an essential driver in the integration process which is also notably simplified through the already established procedures of integrating new methods into the clinical workflows compared to other steadier disciplines in medicine.

8.4.3 Design of a Clinical Application Study

The following subchapter describes the major steps in a clinical application study in order to provide a brief overview of a typical clinical study setup for MIA.

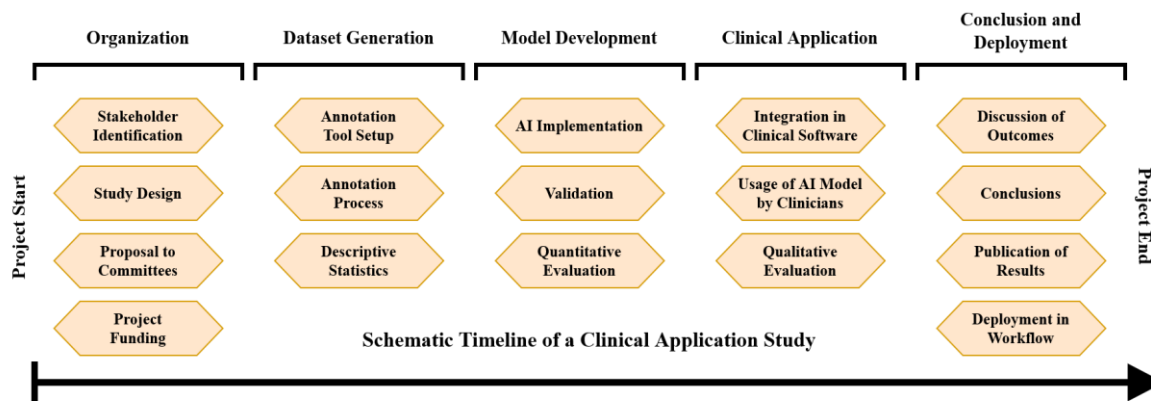


Figure 8.3: Major steps in a clinical application study illustrated with a schematic timeline.

Organization

The first step towards a clinical application is always stakeholder identification. As already mentioned, a clinical study for an AI application requires multiple experts in different fields including medical, deep learning, system administration, and organizational expertise, which is why such experts willing to participate have to be identified and brought together for interdisciplinary collaboration. Afterward, the participating stakeholders design the planned methods and tasks for each participant to solve the shared research question. The result of this

process is a study proposal describing the procedure and milestones of the next months up to years. Consequently, this proposal is submitted to the corresponding committees in the hospital like the ethics committee, data protection officer, or administration for data processing contracts between institutes. As the conduction of clinical application studies requires high hardware and personnel resources, project funding is often pursued, whereby sufficient time must be planned for the processing of an application by funding organizations prior to the start of the project.

Dataset Generation

After the start of the clinical study, the next step is to create a sufficiently large dataset for training and evaluation of the AI model. To achieve this, an annotation tool must be provided to the clinicians to generate annotated images. As medical imaging viewers, which already are available at the hospital and clinicians are trained to operate, often are capable of creating labels and marking ROIs in images, established tools should be utilized if possible. If it is required to deploy a new tool for annotation, it is highly recommended to properly setup the new software by consulting the IT department of the hospital in order to allow reusability for other colleagues and future projects. The integration of tools in the hospital infrastructure which are only used once should be strongly avoided. With a functional annotation framework, one or multiple clinicians can start generating an annotated dataset for MIA for which sufficient time has to be planned. It must be noted that annotation is one of the most important steps in a clinical application and impacts the outcome of the whole study. The last step of the dataset generation is an extensive analysis of the data through descriptive statistics for obtaining first insights as well as quickly detecting a need for rectification.

Model Development

Depending on the imaging modality, the gathered dataset needs proper preprocessing to be usable for training an AI model for MIA. Through the recent availability of standardized frameworks for pipeline building, the development process takes less time compared to the other major steps of a clinical application study. Nevertheless, to obtain a strong AI model, multiple iterations of hyperparameter adjustments, model training, and performance assessment on a validation set can be required for which a deep neural network expert with experience in computer vision is needed. After the successful training of a powerful deep neural network model for MIA, the last step of the model development is the quantitative evaluation via performance assessment on a hold-out testing set. This evaluation is conducted by the deep learning expert and is mostly based on common metrics like F1-score.

Clinical Application

With a trained and validated AI model for accurate prediction, the next step of a clinical application study is the integration of the model into already established software or in a custom user interface in order to be usable by clinicians. The specific requirements of a graphical user interface for proper usability are highly dependent on the corresponding MIA task. It must be considered that the integration into the IT infrastructure of a hospital, even only for research

purposes, can take up long-term efforts and resources which is why this step represents one of the most critical parts of a clinical application study. After the deployment of a user interface hosting the AI model, clinicians can start using the model on prospective or retrospective samples to assess its accuracy, reliability, robustness, and usability. Commonly, this qualitative evaluation by clinicians is conducted retrospectively on a hold-out testing set from the generated dataset.

Conclusion and Deployment

Finally, the results of the quantitative evaluation by deep learning experts as well as the qualitative evaluation by clinicians are discussed in the context of the research question. The clinical application study ends with completing all milestones and writing a report about the outcomes as well as corresponding insights or contributions to the field. Commonly, the results are published at a conference or in a journal. After a successful application study, the integration of the implemented AI model into clinical workflows for further research or routine is conceivable for which medical product certification may be the next step.

8.5 Future Work and Outlook

The Author plans to further contribute to standardization in the field facilitating application and research of deep neural network based MIA. Through the proposed frameworks MIScnn and AUCMEDI, it was possible to create efficient as well as popular toolkits for standardized MIA which enables long-term improvement in the research field. Nevertheless, as already mentioned in the corresponding subchapters, the continuous support and further development of the proposed frameworks is assured for the future by the Author. Thus, the frameworks will still provide a foundation for reliable and maintainable application in clinical research as well as possible integration into clinical workflows as CDS in the future. As the agenda for the further development of the frameworks, it is planned to integrate novel architectures like transformer-based U-Net [526] for MIScnn or 3D variants of ViT [94] as well as ConvNeXt [95] for AUCMEDI. Moreover, the integration of automated ensemble learning techniques is already in development for MIScnn. To publicize AUCMEDI, further MIA challenge participation is planned for demonstrating its outstanding predictive capabilities in the literature. One of the major aims of the Author as future work is also the clinical application of the proposed frameworks in collaboration with the University Hospital Augsburg or other university hospitals. This would allow utilizing the strong predictive capabilities of pipelines based on MIScnn or AUCMEDI to support clinicians in challenging decision-making processes.

In the general research field of MIA, the increasing focus on standardization has grown as clinical applications and CDS systems are becoming more prevalent [29]. In particular, AutoML solutions are already highly desired but will become increasingly popular as the standardization of MIA pipelines further continues and the solutions become more adaptable to different medical imaging modalities [31]. In this context, the Author plans to advance the AutoML capabilities of AUCMEDI by integrating more complex ensemble learning strategies like cascading based on different input sizes, hyperparameters, as well as architectures. Likewise, the AutoML module will continue to offer extensive Docker [350] functionality to assure application in sensitive IT infrastructure as a key feature for future clinical applications.

Another key aspect of future MIA is the inclusion of XAI methods. As clinical application will become more prevalent, user understanding of the decision-making processes by AI models will also become more essential for integration in practical workflows. Recently published MIC studies often include Grad-CAM heatmaps in terms of a new gold-standard in application reporting and performance evaluation [343]. However, the availability of XAI methods for image segmentation, 3D imaging models, or other subfields of MIA is still limited as the literature mainly focuses on methods for 2D image classification, which is tackled by the ongoing in-house study XAIseg. This is why further research is necessary for standardized usage as well as increasing prevalence of XAI methods in MIA fields. Moreover, the real benefit of XAI methods for clinicians is still heavily discussed in the field which is why the presented in-house study EKIPRO is currently being conducted [343, 398].

For the improvement in reproducibility of MIA pipelines, the Author plans to continue his contributions to the field for the identification and strengthening of current weak spots responsible for insufficiency in reproducibility. An extensive meta-analysis of the literature concerning recent neural network applications is pursued by the Author in order to identify the major drivers for lacking reproducibility even though standardized frameworks were utilized like MIScnn or AUCMEDI. These insights would allow further improvement of current toolkits to establish strong reproducibility in the field. As statistical bias is a continuous threat to reproducibility, efforts like workshops for correct performance assessment and evaluation at conferences are endeavored by the Author in the future.

The future of deep neural network based MIA is promising but also presents similar as well as new challenges. Standardization in other subfields of MIA, which were not focused on in this dissertation like object detection [217], also needs to be established for utilizing the full spectrum of MIA techniques in clinical applications. Up-coming research fields like federated learning [459, 460, 527] and privacy-preserving deep learning [527–529] will likely become major research fields in the future, due to the availability of high-performing deep learning methods with the possibility of integration into clinical workflows but the lack of regulation-conform usage of the large amount of data scattered across hospitals. Overcoming these challenges in the future would allow the medical field to utilize powerful AI assistance based on the comprehensive knowledge from multiple medical facilities.

9

Conclusions

In this chapter, the Author summarizes the achieved contributions, contextualizes the results within the defined objectives, and evaluates the accomplishments of this dissertation.

➤ **Main Objective: Standardization**

The main objective of this dissertation was the development of frameworks for standardized pipeline building in deep neural network based medical image classification and segmentation. In order to define the current state-of-the-art of modern medical image analysis pipelines, a comprehensive meta-analysis was conducted which provided an overview of mandatory methods to implement in a framework for standardized pipeline building. Consequently, the Author developed the Python frameworks MIScnn for medical image segmentation and AUCMEDI for medical image classification. The proposed frameworks follow the philosophy of providing simple and intuitive building blocks to setup pipelines utilizing state-of-the-art methodology for straightforward application as well as enabling complex research without the need to continuously reimplement common components. To achieve standardization and ensure usability, the proposed frameworks provide outstanding open-source features like software stability and reproducibility through continuous integration as well as delivery, utilization of contributor-friendly coding conventions, and extensive documentation including wikis as well as usage examples. The frameworks, in particular MIScnn, were able to establish a considerably large as well as active open-source community represented through more than 160,000 downloads in total. Moreover, AUCMEDI combined the concept of an extensive but simple toolkit for deep learning experts with straightforward usability for interested clinicians through automated machine learning techniques. The Author proved through more than a dozen in-house studies as well as various external studies conducted around the globe, that the proposed frameworks provide excellent segmentation as well as classification capabilities, simplification of experimentation for research, and reliable as well as rapid application. It was possible to conclude that the proposed frameworks made a major contribution to standardized medical image analysis by enabling diverse clinical applications, supporting research, and increasing reproducibility.

➤ **Subsidiary Objectives: Application, Research, and Reproducibility**

As the standardization of pipeline building in medical image segmentation as well as classification was the main objective of this dissertation, the subsidiary three objectives focused on the utilization of the proposed frameworks and the further advancement of the research field.

The second objective was to demonstrate the capabilities of the proposed frameworks through application studies. The Author presented a strong kidney tumor segmentation in 3D CT scans, standardized image classification across various medical disciplines, state-of-the-art defining multi-disease detection in retinal imaging capable of recognizing even rare diseases, state-of-the-art defining COVID-19 infection and severity prediction in 3D CT scans implemented as highly reproducible as well as easily applicable Docker container by a collaboration of multiple labs at the University Augsburg, and a clinical study analyzing the informative value of explainable AI for pathologists on the use case of prostate carcinoma by a collaboration of the University Augsburg and University Hospital Augsburg. These studies successfully proved that the proposed frameworks are adaptable to different medical imaging data, provide state-of-the-art methods, demonstrate reliability as well as robustness, and achieve high performance.

The third objective was to utilize the proposed frameworks to further advance the state-of-the-art of medical image segmentation as well as classification. One of the core advantages of standardized pipeline building is the potential to substantially support research through the rapid integration of novel methods for experimentation and comparison. This dissertation presented a study on effective nucleus segmentation based on noise-introduced annotations in microscopy imaging, a study on effective COVID-19 lung infection segmentation based on limited training data of 3D CT scans, and two studies analyzing the performance impact of ensemble learning in image segmentation as well as classification. Furthermore, various external studies reported effective utilization of the MIScnn framework for deep learning, application, and clinical research. These studies directly contributed to progressing the field and proved that the proposed frameworks facilitated further research.

The last objective of this dissertation was to improve reproducibility in medical image segmentation. Through the development of frameworks for standardized pipeline building, it was possible to achieve a substantial contribution toward the general reproducibility of implementations for medical image segmentation. However, performance assessment revealed significant statistical biases in the field. Thus, the Author introduced the framework MISeval for standardized and reliable performance assessment in medical image segmentation which revealed increasing popularity. In addition, the Author presented a guideline for evaluation metrics discussing pitfalls as well as metric behavior in order to improve evaluation quality in the community, and a novel metric for performance assessment in the presence of control samples. This work contributed to strengthening performance assessment and increasing reproducibility as well as reusability of medical image segmentation models in the field.

➤ **Concluding Words**

The proposed frameworks in this work led to an improved knowledge representation of the field, enabled rapid as well as high-performing applications, facilitated further research, and strengthen the reproducibility of future studies. Consequently, these solutions contributed considerably to the progress of clinical decision support by paving the way for standardized as well as maintainable medical image analysis pipelines and allowing reliable application as well as integration into clinical workflows. The Author hopes that these contributions will help migrate medical image analysis pipelines from the research labs into clinical applications and thereby further improve healthcare.

“The science of today is the technology of tomorrow.”

Edward Teller, Hungarian-American physicist, 1908-2003, [530].

References

1. Alan L. Mackay. *A Dictionary of Scientific Quotations*. 1st Editio. CRC Press; 1991.
2. Bohr A, Memarzadeh K. The rise of artificial intelligence in healthcare applications. In: *Artificial Intelligence in Healthcare*. Elsevier; 2020. p. 25–60.
3. Stone P, Brooks R, Brynjolfsson E, Calo R, Etzioni O, Hager G, et al. *Artificial Intelligence and life in 2030: the one hundred year study on artificial intelligence*. 2016. <https://ai100.stanford.edu>. Accessed 4 Nov 2022.
4. Dignum V. Ethics in artificial intelligence: introduction to the special issue. *Ethics and Information Technology*. 2018;20:1–3. doi:10.1007/s10676-018-9450-z.
5. Popenici SAD, Kerr S. Exploring the impact of artificial intelligence on teaching and learning in higher education. *Res Pract Technol Enhanc Learn*. 2017;12:1–13. doi:10.1186/s41039-017-0062-8.
6. *The Oxford Dictionary of Phrase and Fable*. Oxford University Press; 2005.
7. Hutter F, Kotthoff L, Vanschoren J. *Automated Machine Learning*. Cham: Springer International Publishing; 2019. doi:10.1007/978-3-030-05318-5.
8. Hosny A, Parmar C, Quackenbush J, Schwartz LH, Aerts HJWL. Artificial intelligence in radiology. *Nat Rev Cancer*. 2018;18:500–10. doi:10.1038/s41568-018-0016-5.
9. Esteva A, Chou K, Yeung S, Naik N, Madani A, Mottaghi A, et al. Deep learning-enabled medical computer vision. *npj Digit Med*. 2021;4:5. doi:10.1038/s41746-020-00376-2.
10. Handels H. *Medizinische Bildverarbeitung*. 2009.
11. Chan H-P, Samala RK, Hadjiiski LM, Zhou C. Deep Learning in Medical Image Analysis. In: *Advances in Experimental Medicine and Biology*. Springer; 2020. p. 3–21. doi:10.1007/978-3-030-33128-3_1.
12. Ravi D, Wong C, Deligianni F, Berthelot M, Andreu-Perez J, Lo B, et al. Deep Learning for Health Informatics. *IEEE J Biomed Heal Informatics*. 2017;21:4–21. doi:10.1109/JBHI.2016.2636665.
13. Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, et al. A survey on deep learning in medical image analysis. *Med Image Anal*. 2017;42 December 2012:60–88. doi:10.1016/j.media.2017.07.005.
14. Feng X, Jiang Y, Yang X, Du M, Li X. Computer vision algorithms and hardware implementations: A survey. *Integration*. 2019;69:309–20. doi:10.1016/j.vlsi.2019.07.005.
15. Lee K, Zung J, Li P, Jain V, Seung HS. Superhuman Accuracy on the SNEMI3D Connectomics Challenge. 2017; Nips:1–11. <http://arxiv.org/abs/1706.00120>.
16. Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press; 2018. <https://mitpress.mit.edu/9780262039406/foundations-of-machine-learning/>. Accessed 26 Aug 2022.
17. Szeliski R. *Computer Vision: Algorithms and Applications*. Cham: Springer International Publishing; 2022. doi:10.1007/978-3-030-34372-9.
18. Scatliff JH, Morris PJ. From Roentgen to magnetic resonance imaging: the history of medical imaging. *N C Med J*. 2014;75:111–3. doi:10.18043/ncm.75.2.111.
19. Bourne R. *Fundamentals of Digital Imaging in Medicine*. 1st edition. London: Springer London; 2010. doi:10.1007/978-1-84882-087-6.
20. Darby MJ, Barron DA, Hyland RE. *Oxford Handbook of Medical Imaging*. 2012.
21. Suetens P. *Fundamentals of Medical Imaging*. Second Edi. Cambridge University Press; 2009.
22. Manning D, Ethell S, Donovan T, Crawford T. How do radiologists do it? The influence of experience and training on searching for chest nodules. *Radiography*. 2006;12:134–42. doi:10.1016/j.radi.2005.02.003.
23. Ma J, Wang Y, An X, Ge C, Yu Z, Chen J, et al. Towards Efficient COVID-19 CT Annotation: A Benchmark for Lung and Infection Segmentation. 2020;:1–7. <http://arxiv.org/abs/2004.12537>.
24. Sutton RT, Pincock D, Baumgart DC, Sadowski DC, Fedorak RN, Kroeker KI. An overview of clinical decision support systems: benefits, risks, and strategies for success. *npj Digital Medicine*. 2020;3:1–10. doi:10.1038/s41746-020-0221-y.
25. Berner ES, editor. *Clinical Decision Support Systems*. Cham: Springer International Publishing; 2016. doi:10.1007/978-3-319-31913-1.
26. Wang G. A Perspective on Deep Imaging. *IEEE Access*. 2016;4:8914–24. doi:10.1109/ACCESS.2016.2624938.
27. Shen D, Wu G, Suk H-I. Deep Learning in Medical Image Analysis. *Annu Rev Biomed Eng*. 2017;19:221–48. doi:10.1146/annurev-bioeng-071516-044442.
28. Bankman I. *Handbook of Medical Image Processing and Analysis*. Elsevier LTD.; 2009.
29. Limkin EJ, Sun R, Dercle L, Zacharakis EI, Robert C, Reuzé S, et al. Promises and challenges for the implementation of computational medical imaging (radiomics) in oncology. *Ann Oncol*. 2017;28:1191–206. doi:10.1093/annonc/mdx034.
30. Puttagunta M, Ravi S. Medical image analysis based on deep learning approach. *Multimed Tools Appl*. 2021;80:24365–98. doi:10.1007/s11042-021-10707-4.
31. Faes L, Wagner SK, Fu DJ, Liu X, Korot E, Ledsam JR, et al. Automated deep learning design for medical image classification by health-care professionals with no coding experience: a feasibility study. *Lancet Digit Heal*. 2019;1:e232–42. doi:10.1016/S2589-7500(19)30108-6.
32. El Naqa IM, Hu Q, Chen W, Li H, Fuhrman JD, Gorre N, et al. Lessons learned in transitioning to AI in the medical imaging of COVID-19. *J Med Imaging*. 2021;8:010902. doi:10.1117/1.jmi.8.s1.010902.
33. Waring J, Lindvall C, Umeton R. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artif Intell Med*. 2020;104:101822. doi:10.1016/j.artmed.2020.101822.
34. Ker J, Wang L, Rao J, Lim T. Deep Learning Applications in Medical Image Analysis. *IEEE Access*. 2018;6:9375–89. doi:10.1109/ACCESS.2017.2788044.
35. Fourcade A, Khonsari RH. Deep learning in medical image analysis: A third eye for doctors. *J Stomatol Oral Maxillofac Surg*. 2019;120:279–88. doi:10.1016/j.jormas.2019.06.002.
36. Muhammad S, Muhammad A, Adnan M, Muhammad Q, Majidi A, Khan MK, et al. Medical Image Analysis using Convolutional Neural Networks A Review. *J Med Syst*. 2018;42:1–13. doi:10.1007/s10916-018-1088-1.
37. Savadjiev P, Chong J, Dohan A, Vakalopoulou M, Reinhold C, Paragios N, et al. Demystification of AI-driven medical image interpretation: past, present and future. *European Radiology*. 2019;29:1616–24. doi:10.1007/s00330-018-5674-x.
38. Swarna SR, Boyapati S, Dutt V, Bajaj K. Deep Learning in Dynamic Modeling of Medical Imaging: A Review Study. In: *2020 3rd International Conference on Intelligent Sustainable*

- Systems (ICISS). IEEE; 2020. p. 745–9. doi:10.1109/ICISS49785.2020.9315990.
39. Bizzego A, Bussola N, Chierici M, Maggio V, Francescato M, Cima L, et al. Evaluating reproducibility of AI algorithms in digital pathology with DAPPER. *PLoS Comput Biol*. 2019;15:e1006269. doi:10.1371/journal.pcbi.1006269.
40. Park JE, Park SY, Kim HJ, Kim HS. Reproducibility and Generalizability in Radiomics Modeling: Possible Strategies in Radiologic and Statistical Perspectives. *Korean J Radiol*. 2019;20:1124. doi:10.3348/kjr.2018.0070.
41. Hutson M. Artificial intelligence faces reproducibility crisis. *Science* (80-). 2018;359:725–6. doi:10.1126/science.359.6377.725.
42. Bahaidarah L, Hung E, Oliveira AFDM, Penumaka J, Rosario L, Trisovic A. Toward Reusable Science with Readable Code and Reproducibility. 2021. <http://arxiv.org/abs/2109.10387>. Accessed 23 Sep 2022.
43. Gibson E, Li W, Sudre C, Fidon L, Shakeri DI, Wang G, et al. NiftyNet: a deep-learning platform for medical imaging. *Comput Methods Programs Biomed*. 2018;158:113–22. doi:10.1016/j.cmpb.2018.01.025.
44. Renard F, Guedria S, Palma N De, Vuillerme N. Variability and reproducibility in deep learning for medical image segmentation. *Sci Rep*. 2020;10:1–16. doi:10.1038/s41598-020-69920-0.
45. Parikh RB, Teeple S, Navathe AS. Addressing Bias in Artificial Intelligence in Health Care. *JAMA - Journal of the American Medical Association*. 2019;322:2377–8. doi:10.1001/jama.2019.18058.
46. Zhang Y, Mehta S, Caspi A. Rethinking Semantic Segmentation Evaluation for Explainability and Model Selection. 2021. <http://arxiv.org/abs/2101.08418>. Accessed 8 Jan 2022.
47. Powers DMW. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2020. <http://arxiv.org/abs/2010.16061>. Accessed 8 Jan 2022.
48. Reinke A, Tizabi MD, Sudre CH, Eisenmann M, Rädtsch T, Baumgartner M, et al. Common Limitations of Image Processing Metrics: A Picture Story. 2021. <http://arxiv.org/abs/2104.05642>. Accessed 30 Oct 2022.
49. Müller D, Kramer F. MISenn: a framework for medical image segmentation with convolutional neural networks and deep learning. *BMC Med Imaging*. 2021;21:12. doi:10.1186/s12880-020-00543-7.
50. Heaven WD. Hundreds of AI tools have been built to catch covid. None of them helped. | MIT Technology Review. <https://www.technologyreview.com/2021/07/30/1030329/machine-learning-ai-failed-covid-hospital-diagnosis-pandemic/>. Accessed 20 Sep 2022.
51. Drepper J, Semler SC. IT-Infrastrukturen in der patientenorientierten Forschung. 2014. <https://www.toolpool-gesundheitsforschung.de/produkte/it-report>.
52. Sandeep Kumar E, Satya Jayadev P. Deep Learning for Clinical Decision Support Systems: A Review from the Panorama of Smart Healthcare. Springer, Cham; 2020. p. 79–99. doi:10.1007/978-3-030-33966-1_5.
53. Scimago Journal & Country Rank. <https://www.scimagojr.com/>. Accessed 11 Aug 2022.
54. Martin Dugas. *Medizininformatik*. Springer Vieweg; 2018.
55. Staal J, Abramoff MD, Niemeijer M, Viergever MA, Van Ginneken B. Ridge-based vessel segmentation in color images of the retina. *IEEE Trans Med Imaging*. 2004;23:501–9. doi:10.1109/TMI.2004.825627.
56. Bernal J, Sánchez FJ, Fernández-Esparrach G, Gil D, Rodríguez C, Vilariño F. WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. *Comput Med Imaging Graph*. 2015;43:99–111. doi:10.1016/j.compmedimag.2015.02.007.
57. Combalia M, Codella NCF, Rotemberg V, Helba B, Vilaplana V, Reiter O, et al. BC20000: Dermoscopic Lesions in the Wild. 2019. <http://arxiv.org/abs/1908.02288>. Accessed 28 Oct 2021.
58. Codella NCF, Gutman D, Celebi ME, Helba B, Marchetti MA, Dusza SW, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 International symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). In: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). IEEE; 2018. p. 168–72. doi:10.1109/ISBI.2018.8363547.
59. Tschandl P, Rosendahl C, Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Sci Data*. 2018;5:1–9. doi:10.1038/sdata.2018.161.
60. Cheng J, Yang W, Huang M, Huang W, Jiang J, Zhou Y, et al. Retrieval of Brain Tumors by Adaptive Spatial Pooling and Fisher Vector Representation. *PLoS One*. 2016;11:e0157112. doi:10.1371/journal.pone.0157112.
61. Al-Dhabyani W, Goma M, Khaled H, Fahmy A. Dataset of breast ultrasound images. *Data Br*. 2020;28. doi:10.1016/j.dib.2019.104863.
62. Kermany DS, Goldbaum M, Cai W, Valentim CCS, Liang H, Baxter SL, et al. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*. 2018;172:1122–1131.e9. doi:10.1016/j.cell.2018.02.010.
63. Bulten W, Kartasalo K, Chen PHC, Ström P, Pinckaers H, Nagpal K, et al. Artificial intelligence for diagnosis and Gleason grading of prostate cancer: the PANDA challenge. *Nat Med*. 2022;28:154–63. doi:10.1038/s41591-021-01620-2.
64. National Institute of Biomedical Imaging and Bioengineering (NIBIB). *Nuclear Medicine*. <https://www.nibib.nih.gov/science-education/science-topics/nuclear-medicine>. Accessed 7 Oct 2022.
65. Petersilge CA, McDonald J, Bishop M, Yudkovitch L, Treuting C, Towbin AJ. Visible Light Imaging: Clinical Aspects with an Emphasis on Medical Photography—a HIMSS-SIIM Enterprise Imaging Community Whitepaper. *J Digit Imaging*. 2022;35:385–95.
66. Mitchell TM. *Machine Learning*. McGraw-Hill Science/Engineering/Math; 1997.
67. Kevin Zhou, Hayit Greenspan, Dinggang Shen. *Deep Learning for Medical Image Analysis*. Academic Press; 2017.
68. Kröse B, Smagt P van der. *An Introduction to Neural Networks*. 1993.
69. Goodfellow I, Bengio Y, Courville · Aaron. *Deep Learning*. MIT Press; 2016. <https://www.deeplearningbook.org/>.
70. O’Shea K, Nash R. *An Introduction to Convolutional Neural Networks*. 2015. <http://arxiv.org/abs/1511.08458>. Accessed 4 Oct 2022.
71. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS’19). 2019. doi:10.5555/3454287.3455008.
72. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. <https://www.tensorflow.org/>.
73. Agarap AF. Deep Learning using Rectified Linear Units (ReLU). 2018. <http://arxiv.org/abs/1803.08375>. Accessed 1 Sep 2022.
74. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, et al. Review of deep learning: concepts, CNN

- architectures, challenges, applications, future directions. *J Big Data*. 2021;8:53. doi:10.1186/s40537-021-00444-8.
75. Altaf F, Islam SMS, Akhtar N, Janjua NK. Going deep in medical image analysis: Concepts, methods, challenges, and future directions. *IEEE Access*. 2019;7:99540–72.
76. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis*. 2015;115:211–52. doi:10.1007/s11263-015-0816-y.
77. Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E. Deep Learning for Computer Vision: A Brief Review. *Comput Intell Neurosci*. 2018;2018:1–13. doi:10.1155/2018/7068349.
78. Ridnik T, Sharir G, Ben-Cohen A, Ben-Baruch E, Noy A. ML-Decoder: Scalable and Versatile Classification Head. 2021. <http://arxiv.org/abs/2111.12933>. Accessed 23 Oct 2022.
79. Tan M, Le Q V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. 36th Int Conf Mach Learn ICML 2019. 2019;2019-June:10691–700. <http://arxiv.org/abs/1905.11946>. Accessed 27 Feb 2021.
80. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society; 2016. p. 770–8. doi:10.1109/CVPR.2016.90.
81. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems*. 2012. doi:10.1145/3065386.
82. Wang W, Yang Y. Development of convolutional neural network and its application in image classification: a survey. *Opt Eng*. 2019;58:1. doi:10.1117/1.OE.58.4.040901.
83. Deng J, Dong W, Socher R, Li L-J, Kai Li, Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *Institute of Electrical and Electronics Engineers (IEEE)*; 2010. p. 248–55.
84. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, International Conference on Learning Representations, ICLR; 2015*. <http://www.robots.ox.ac.uk/>. Accessed 17 Dec 2021.
85. Sultana F, Sufian A, Dutta P. Advancements in image classification using convolutional neural network. In: *Proceedings - 2018 4th IEEE International Conference on Research in Computational Intelligence and Communication Networks, ICRCICN 2018*. Institute of Electrical and Electronics Engineers Inc.; 2018. p. 122–9.
86. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going Deeper with Convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015. p. 1–9.
87. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the Inception Architecture for Computer Vision. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society; 2016. p. 2818–26. doi:10.1109/CVPR.2016.308.
88. Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Institute of Electrical and Electronics Engineers Inc.; 2017. p. 5987–95. doi:10.1109/CVPR.2017.634.
89. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, inception-ResNet and the impact of residual connections on learning. In: *31st AAAI Conference on Artificial Intelligence, AAAI 2017*. AAAI press; 2017. p. 4278–84.
90. Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely Connected Convolutional Networks. In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Institute of Electrical and Electronics Engineers Inc.; 2017. p. 2261–9. doi:10.1109/CVPR.2017.243.
91. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. <http://arxiv.org/abs/1704.04861>. Accessed 1 Sep 2022.
92. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*. 2018;:4510–20. <http://arxiv.org/abs/1801.04381>. Accessed 17 Dec 2021.
93. Howard A, Sandler M, Chu G, Chen L-C, Chen B, Tan M, et al. Searching for MobileNetV3. *Proc IEEE Int Conf Comput Vis*. 2019;2019-October:1314–24. <http://arxiv.org/abs/1905.02244>. Accessed 23 Oct 2022.
94. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020. <http://arxiv.org/abs/2010.11929>. Accessed 26 Sep 2022.
95. Liu Z, Mao H, Wu C-Y, Feichtenhofer C, Darrell T, Xie S. A ConvNet for the 2020s. 2022;:11966–76. <http://arxiv.org/abs/2201.03545>. Accessed 23 Oct 2022.
96. Zoph B, Vasudevan V, Shlens J, Le Q V. Learning Transferable Architectures for Scalable Image Recognition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society; 2018. p. 8697–710.
97. Chollet F. Xception: Deep learning with depthwise separable convolutions. *Institute of Electrical and Electronics Engineers Inc.*; 2017.
98. Müller D, Mayer S, Hartmann D, Soto-Rey I, Kramer F. AUCMEDI - A Framework for Automated Classification of Medical Images. 2022. doi:10.5281/zenodo.6633540.
99. Ioannidou A, Chatzilari E, Nikolopoulos S, Kompatsiaris I. Deep learning advances in computer vision with 3D data: A survey. *ACM Comput Surv*. 2017;50. doi:10.1145/3042064.
100. Hesamian MH, Jia W, He X, Kennedy P. Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges. *J Digit Imaging*. 2019;32:582–96. doi:10.1007/s10278-019-00227-x.
101. Singh SP, Wang L, Gupta S, Goli H, Padmanabhan P, Gulyás B. 3D Deep Learning on Medical Images: A Review. 2020;:1–13. <http://arxiv.org/abs/2004.00218>.
102. Ma J, Ge C, Wang Y, An X, Gao J, Yu Z, et al. COVID-19 CT Lung and Infection Segmentation Dataset. 2020. doi:10.5281/zenodo.3757476.
103. Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D. Image Segmentation Using Deep Learning: A Survey. 2020. <http://arxiv.org/abs/2001.05566>. Accessed 9 Nov 2020.
104. Ulku I, Akagunduz E. A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images. 2019. <http://arxiv.org/abs/1912.10230>. Accessed 21 Oct 2020.
105. Asgari Taghanaki S, Abhishek K, Cohen JP, Cohen-Adad J, Hamarneh G. Deep semantic segmentation of natural and medical images: a review. *Artif Intell Rev*. 2020;:1–42. doi:10.1007/s10462-020-09854-1.
106. Badrinarayanan V, Kendall A, Cipolla R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans Pattern Anal Mach Intell*. 2017;39:2481–95. doi:10.1109/TPAMI.2016.2644615.
107. Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans Pattern Anal Mach Intell*. 2018;40:834–48.

108. Ronneberger O, Philipp Fischer, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015.
109. Isensee F, Jaeger PF, Kohl SAA, Petersen J, Maier-Hein KH. nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation. *Nat Methods*. 2021;18:203–11. doi:10.1038/s41592-020-01008-z.
110. Müller D, Soto-Rey I, Kramer F. Robust chest CT image segmentation of COVID-19 lung infection based on limited data. *Informatics Med Unlocked*. 2021;25:100681. doi:10.1016/j.imu.2021.100681.
111. Wang R, Lei T, Cui R, Zhang B, Meng H, Nandi AK. Medical image segmentation using deep learning: A survey. *IET Image Process*. 2022;16:1243–67. doi:10.1049/ipr2.12419.
112. Zhang Z, Liu Q, Wang Y. Road Extraction by Deep Residual U-Net. *IEEE Geosci Remote Sens Lett*. 2018;15:749–53. doi:10.1109/LGRS.2018.2802944.
113. Kolařík M, Burget R, Uher V, Říha K, Dutta M. Optimized High Resolution 3D Dense-U-Net Network for Brain and Spine Segmentation. *Appl Sci*. 2019;9:404.
114. Ibtehaz N, Rahman MS. MultiResUNet: Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation. 2019. <http://arxiv.org/abs/1902.04049>. Accessed 19 Jul 2019.
115. Oktay O, Schlemper J, Le Folgoc L, Lee M, Heinrich M, Misawa K, et al. Attention U-Net: Learning Where to Look for the Pancreas. In: *Medical Imaging with Deep Learning 2018*. 2018. <https://openreview.net/forum?id=Skft7cijM>. Accessed 24 Oct 2022.
116. Milletari F, Navab N, Ahmadi SA. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In: *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*. Institute of Electrical and Electronics Engineers Inc.; 2016. p. 565–71.
117. Çiçek Ö, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2016. p. 424–32. doi:10.1007/978-3-319-46723-8_49.
118. Isensee F, Jaeger PF, Kohl SAA, Petersen J, Klaus H. Automated Design of Deep Learning Methods for Biomedical Image Segmentation. 2020;:1–55. <https://arxiv.org/abs/1904.08128>.
119. Smistad E, Falch TL, Bozorgi M, Elster AC, Lindseth F. Medical image segmentation on GPUs – A comprehensive review. *Med Image Anal*. 2015;20:1–18. doi:10.1016/j.media.2014.10.012.
120. Wolfgang Kuhlen T, Scholl I, Aach T, Deserno TM, Kuhlen T, Scholl I, et al. Challenges of medical image processing. *Comput Sci Res Dev*. 2011;26:5–13. doi:10.1007/s00450-010-0146-9.
121. Toennies KD. The Analysis of Medical Images. In: *Guide to Medical Image Analysis*. London: Springer London; 2012. p. 1–19. doi:10.1007/978-1-4471-2751-2_1.
122. Duncan JS, Ayache N. Medical image analysis: Progress over two decades and the challenges ahead. *IEEE Trans Pattern Anal Mach Intell*. 2000;22:85–106.
123. Viergever MA, Maintz JBA, Klein S, Murphy K, Staring M, Pluim JPW. A survey of medical image registration – under review. *Medical Image Analysis*. 2016;33:140–4.
124. Pham DL, Xu C, Prince JL. Current Methods in Medical Image Segmentation. *Annu Rev Biomed Eng*. 2000;2:315–37. doi:10.1146/annurev.bioeng.2.1.315.
125. Liu X, Song L, Liu S, Zhang Y. A review of deep-learning-based medical image segmentation methods. *Sustain*. 2021;13:1–29. doi:10.3390/su13031224.
126. Minh D, Wang HX, Li YF, Nguyen TN. Explainable artificial intelligence: a comprehensive review. *Artif Intell Rev*. 2022;55:3503–68. doi:10.1007/s10462-021-10088-y.
127. Pianykh OS. *Digital Imaging and Communications in Medicine (DICOM)*. Springer Berlin Heidelberg; 2012.
128. Neuroimaging Informatics Technology Initiative. <https://nifti.nih.gov/background>. Accessed 19 Jul 2019.
129. Heller N, Sathianathan N, Kalapara A, Walczak E, Moore K, Kaluzniak H, et al. The KiTS19 Challenge Data: 300 Kidney Tumor Cases with Clinical Context, CT Semantic Segmentations, and Surgical Outcomes. 2019. <http://arxiv.org/abs/1904.00445>. Accessed 19 Jul 2019.
130. Prior FW, Clark K, Commean P, Freymann J, Jaffe C, Kirby J, et al. TCIA: An information resource to enable open science. In: *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. 2013. p. 1282–5.
131. Menze BH, Jakab A, Bauer S, Kalpathy-Cramer J, Farahani K, Kirby J, et al. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). *IEEE Trans Med Imaging*. 2015;34:1993–2024.
132. Herrmann MD, Clunie DA, Fedorov A, Doyle SW, Pieper S, Klepeis V, et al. Implementing the DICOM standard for digital pathology. *J Pathol Inform*. 2018;9:37.
133. Pennefather PS, Suhanic W. BioTIFF: A New BigTIFF File Structure For Organizing Large Image Datasets And Their Associated Metadata. *Biophys J*. 2009;96:30a. doi:10.1016/j.bpj.2008.12.045.
134. Lowekamp BC, Chen DT, Ibáñez L, Blezek D. The Design of SimpleITK. *Front Neuroinform*. 2013;7 DEC:45. doi:10.3389/fninf.2013.00045.
135. Yaniv Z, Lowekamp BC, Johnson HJ, Beare R. SimpleITK Image-Analysis Notebooks: a Collaborative Environment for Education and Reproducible Research. *J Digit Imaging*. 2018;31:290–303. doi:10.1007/s10278-017-0037-8.
136. Reading and Writing for Images and Transforms — SimpleITK 1.2.0.dev documentation. <https://simpleitk.readthedocs.io/en/v1.2.4/Documentation/docs/source/IO.html>. Accessed 27 Aug 2022.
137. Revel M-P, Boussouar S, de Margerie-Mellon C, Saab I, Lapotre T, Mompoin D, et al. Study of Thoracic CT in COVID-19: The STOIC Project. *Radiology*. 2021;301:E361–70. doi:10.1148/radiol.2021210384.
138. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. *Nature*. 2020;585:357–62. doi:10.1038/s41586-020-2649-2.
139. Bradski G. *The OpenCV Library*. Dr Dobb's J Softw Tools. 2000.
140. Buslaev A, Iglovikov VI, Khvedchenya E, Parinov A, Druzhinin M, Kalinin AA. Alumentations: Fast and Flexible Image Augmentations. *Information*. 2020;11:125. doi:10.3390/info11020125.
141. Pérez-García F, Sparks R, Ourselin S. TorchIO: A Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *Comput Methods Programs Biomed*. 2021;208:106236.
142. Isensee F, Jäger P, Wasserthal J, Zimmerer D, Petersen J, Kohl S, et al. batchgenerators - a python framework for data augmentation. 2020. doi:doi:10.5281/zenodo.3632567.
143. Buzug T. *Computed tomography: From photon statistics to modern cone-beam CT*. Springer Berlin Heidelberg; 2008.
144. Herbert Lepor. *Prostatic Diseases*. W.B. Saunders Company;

- 2000.
145. Buzug TM. *Computed Tomography*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. doi:10.1007/978-3-540-39408-2.
146. Masoudi S, Harmon SA, Mehralivand S, Walker SM, Raviprakash H, Bagci U, et al. Quick guide on radiology image pre-processing for deep learning applications in prostate cancer research. *J Med Imaging*. 2021;8:010901. doi:10.1117/1.JMI.8.1.010901.
147. Zhou T, Ruan S, Canu S. A review: Deep learning for medical image segmentation using multi-modality fusion. 3-4:100004. doi:10.1016/j.array.2019.100004.
148. Wang G, Li W, Ourselin S, Vercauteren T. Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks. In: *International MICCAI Brainlesion Workshop*. 2018. p. 178-90. doi:10.1007/978-3-319-75238-9_16.
149. García-Lorenzo D, Francis S, Narayanan S, Arnold DL, Collins DL. Review of automatic segmentation methods of multiple sclerosis white matter lesions on conventional magnetic resonance imaging. *Med Image Anal*. 2013;17:1-18.
150. Isensee F, Jäger PF, Full PM, Vollmuth P, Maier-Hein KH. nnU-Net for Brain Tumor Segmentation. In: *International MICCAI Brainlesion Workshop*. Springer Science and Business Media Deutschland GmbH; 2021. p. 118-32. doi:10.1007/978-3-030-72087-2_11.
151. Zhang ML, Li YK, Liu XY, Geng X. Binary relevance for multi-label learning: an overview. *Front Comput Sci*. 2018;12:191-202.
152. Sorower MS. A literature survey on algorithms for multi-label learning. 2010. doi:10.1.1.364.5612.
153. Le KH, Tran T V., Pham HH, Nguyen HT, Le TT, Nguyen HQ. Learning from Multiple Expert Annotators for Enhancing Anomaly Detection in Medical Image Analysis. 2022. <http://arxiv.org/abs/2203.10611>. Accessed 28 Aug 2022.
154. Rodriguez-Ruiz A, Lång K, Gubern-Merida A, Broeders M, Gennaro G, Clauser P, et al. Stand-Alone Artificial Intelligence for Breast Cancer Detection in Mammography: Comparison With 101 Radiologists. *J Natl Cancer Inst*. 2019;111:916-22.
155. Xiong Z, Wang R, Bai HX, Halsey K, Mei J, Li YH, et al. Artificial Intelligence Augmentation of Radiologist Performance in Distinguishing COVID-19 from Pneumonia of Other Origin at Chest CT. *Radiology*. 2020;296:E156-65.
156. Aljabri M, AlAmir M, AlGhamdi M, Abdel-Mottaleb M, Collado-Mesa F. Towards a better understanding of annotation tools for medical imaging: a survey. *Multimed Tools Appl*. 2022;81:25877-911.
157. Cheng J, Huang W, Cao S, Yang R, Yang W, Yun Z, et al. Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition. *PLoS One*. 2015;10:e0140381. doi:10.1371/journal.pone.0140381.
158. Pachade S, Porwal P, Thulkar D, Kokare M, Deshmukh G, Sahasrabudhe V, et al. Retinal Fundus Multi-Disease Image Dataset (RFMiD): A Dataset for Multi-Disease Detection Research. *Data*. 2021;6:14. doi:10.3390/data6020014.
159. Müller D, Soto-Rey I, Kramer F. Multi-disease detection in retinal imaging based on ensembling heterogeneous deep learning models. In: *Studies in Health Technology and Informatics*. IOS Press BV; 2021. doi:10.3233/SHTI210537.
160. Gessert N, Nielsen M, Shaikh M, Werner R, Schlaefer A. Skin Lesion Classification Using Ensembles of Multi-Resolution EfficientNets with Meta Data. *arXiv*. 2019. <http://arxiv.org/abs/1910.03910>. Accessed 27 Feb 2021.
161. De Raad KB, Van Garderen KA, Smits M, Van Der Voort SR, Incekar F, Oei EH, et al. The effect of preprocessing on convolutional neural networks for medical image segmentation. In: *Proceedings - International Symposium on Biomedical Imaging*. IEEE Computer Society; 2021. p. 655-8.
162. Hill DLG, Batchelor PG, Holden M, Hawkes DJ. Medical image registration. *Phys Med Biol*. 2001;46:R1-45. doi:10.1088/0031-9155/46/3/201.
163. Shorten C, Khoshgoftaar TM. A survey on Image Data Augmentation for Deep Learning. *J Big Data*. 2019;6. doi:10.1186/s40537-019-0197-0.
164. Eaton-rosen Z, Bragman F. Improving Data Augmentation for Medical Image Segmentation. In: *1st Conference on Medical Imaging with Deep Learning (MIDL 2018)*. Amsterdam, The Netherlands; 2018.
165. Perez L, Wang J. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. 2017. <http://arxiv.org/abs/1712.04621>. Accessed 23 Jul 2019.
166. Taylor L, Nitschke G. Improving Deep Learning using Generic Data Augmentation. 2017. <http://arxiv.org/abs/1708.06020>. Accessed 23 Jul 2019.
167. Sandfort V, Yan K, Pickhardt PJ, Summers RM. Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks. *Sci Rep*. 2019;9:1-9. doi:10.1038/s41598-019-52737-x.
168. Boyat AK, Joshi BK. A Review Paper : Noise Models in Digital Image Processing. *Signal Image Process An Int J*. 2015;6:63-75. doi:10.5121/sipij.2015.6206.
169. Gudbjartsson H, Patz S. The rician distribution of noisy mri data. *Magn Reson Med*. 1995;34:910-4. doi:10.1002/mrm.1910340618.
170. Kazemina S, Baur C, Kuijper A, van Ginneken B, Navab N, Albarqouni S, et al. GANs for Medical Image Analysis. *arXiv*. 2018. <http://arxiv.org/abs/1809.06222>. Accessed 25 Dec 2020.
171. Chen Y, Yang XH, Wei Z, Heidari AA, Zheng N, Li Z, et al. Generative Adversarial Networks in Medical Image augmentation: A review. *Comput Biol Med*. 2022;144:105382.
172. COVID-19 LUNG CT LESION SEGMENTATION CHALLENGE - 2020 - Grand Challenge. <https://covid-segmentation.grand-challenge.org/COVID-19-20/>. Accessed 29 May 2021.
173. Home - RIADD (ISBI-2021) - Grand Challenge. <https://riadd.grand-challenge.org/Home/>. Accessed 27 Feb 2021.
174. Nalepa J, Myller M, Kawulok M. Training- And Test-Time Data Augmentation for Hyperspectral Image Segmentation. *IEEE Geosci Remote Sens Lett*. 2020;17:292-6.
175. MONAI Consortium. MONAI: Medical Open Network for AI. 2022. <https://github.com/Project-MONAI/MONAI>. Accessed 25 Aug 2022.
176. Yang S, Xiao W, Zhang M, Guo S, Zhao J, Shen F. Image Data Augmentation for Deep Learning: A Survey. 2022. <http://arxiv.org/abs/2204.08610>. Accessed 22 Aug 2022.
177. Chaki J, Dey N. *A Beginner's Guide to Image Preprocessing Techniques*. CRC Press; 2018. doi:10.1201/9780429441134.
178. Jaeger PF, Kohl SAA, Bickelhaupt S, Isensee F, Kuder TA, Schlemmer H-P, et al. Retina U-Net: Embarrassingly Simple Exploitation of Segmentation Supervision for Medical Object Detection. 2018. <http://arxiv.org/abs/1811.08661>.
179. Nyúl LG, Udupa JK. On standardizing the MR image intensity scale. *Magn Reson Med*. 1999;42:1072-81. doi:10.1002/(SICI)1522-2594(199912)42:6<1072::AID-MRM11>3.0.CO;2-M.
180. Yang J, Yuan J, Shen X. Neuronal edge detection with median filtering and gradient sharpening. *Proc 2012 4th Int Symp Inf Sci Eng ISISE 2012*. 2012;:259-62.
181. Hussain Z, Gimenez F, Yi D, Rubin D. Differential Data

- Augmentation Techniques for Medical Imaging Classification Tasks. *AMIA . Annu Symp proceedings AMIA Symp.* 2017;2017:979–84.
182. Song S, Zheng Y, He Y. A review of Methods for Bias Correction in Medical Images. *Biomed Eng Rev.* 2017;3:1–10.
183. Tustison NJ, Avants BB, Cook PA, Zheng Y, Egan A, Yushkevich PA, et al. N4ITK: Improved N3 bias correction. *IEEE Trans Med Imaging.* 2010;29:1310–20. doi:10.1109/TMI.2010.2046908.
184. Gaál G, Maga B, Lukács A. Attention U-Net Based Adversarial Architectures for Chest X-ray Lung Segmentation. 2020;:1–7. <http://arxiv.org/abs/2003.10304>.
185. Esposito M, De Pietro G. An ontology-based fuzzy decision support system for multiple sclerosis. *Eng Appl Artif Intell.* 2011;24:1340–54. doi:10.1016/j.engappai.2011.02.002.
186. Park K-B, Choi SH, Lee JY. M-GAN: Retinal Blood Vessel Segmentation by Balancing Losses Through Stacked Deep Fully Convolutional Networks. *IEEE Access.* 2020;8:146308–22. doi:10.1109/ACCESS.2020.3015108.
187. Hameed Z, Zahia S, Garcia-Zapirain B, Javier Aguirre J, María Vanegas A. Breast Cancer Histopathology Image Classification Using an Ensemble of Deep Learning Models. *Sensors.* 2020;20:4373. doi:10.3390/s20164373.
188. Pei HY, Yang D, Liu GR, Lu T. MPS-net: Multi-point supervised network for ct image segmentation of covid-19. *IEEE Access.* 2021;9:47144–53.
189. Xue D, Zhou X, Li C, Yao Y, Rahaman MM, Zhang JJ, et al. An Application of Transfer Learning and Ensemble Learning Techniques for Cervical Histopathology Image Classification. *IEEE Access.* 2020;8:104603–18. doi:10.1109/ACCESS.2020.2999816.
190. Wang G, Liu X, Li C, Xu Z, Ruan J, Zhu H, et al. A Noise-Robust Framework for Automatic Segmentation of COVID-19 Pneumonia Lesions from CT Images. *IEEE Trans Med Imaging.* 2020;39:2653–63.
191. Horry M, Chakraborty S, Pradhan B, Paul M, Zhu J, Loh HW, et al. Debiasing pipeline improves deep learning model generalization for X-ray based lung nodule detection. 2022. <http://arxiv.org/abs/2201.09563>. Accessed 23 Aug 2022.
192. Feng X, Qing K, Tustison NJ, Meyer CH, Chen Q. Deep convolutional neural network for segmentation of thoracic organs-at-risk using cropped 3D images. *Med Phys.* 2019;46:2169–80. doi:10.1002/mp.13466.
193. Jason Brownlee. *Data Preparation for Machine Learning. Machine Learning Mastery*; 2020; 2020.
194. Willemink MJ, Koszek WA, Hardell C, Wu J, Fleischmann D, Harvey H, et al. Preparing medical imaging data for machine learning. *Radiology.* 2020;295:4–15. doi:10.1148/radiol.2020192224.
195. Chen Q, Xie W, Zhou P, Zheng C, Wu D. Multi-Crop Convolutional Neural Networks for Fast Lung Nodule Segmentation. *IEEE Trans Emerg Top Comput Intell.* 2022;6:1190–200. doi:10.1109/TETCI.2021.3051910.
196. Park J, Yun J, Kim N, Park B, Cho Y, Park HJ, et al. Fully Automated Lung Lobe Segmentation in Volumetric Chest CT with 3D U-Net: Validation with Intra- and Extra-Datasets. *J Digit Imaging.* 2020;33:221–30.
197. Talebi H, Milanfar P. Learning to Resize Images for Computer Vision Tasks. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE; 2021. p. 487–96. doi:10.1109/ICCV48922.2021.00055.
198. Frangi AF, Tsafaris SA, Prince JL. *Simulation and Synthesis in Medical Imaging.* Springer International Publishing; 2018. doi:10.1109/TMI.2018.2800298.
199. Zhi-Hua Zhou. *Machine Learning.* Springer Nature; 2021.
200. Heller N, Isensee F, Maier-Hein KH, Hou X, Xie C, Li F, et al. The state of the art in kidney and kidney tumor segmentation in contrast-enhanced CT imaging: Results of the KiTS19 Challenge. 2020. <https://kits19.grand-challenge.org>. Accessed 2 Sep 2022.
201. Kingma DP, Lei Ba J. Adam: A Method for Stochastic Optimization. 2014. <https://arxiv.org/abs/1412.6980>.
202. Liu L, Jiang H, He P, Chen W, Liu X, Gao J, et al. On the Variance of the Adaptive Learning Rate and Beyond. 2019. <http://arxiv.org/abs/1908.03265>. Accessed 26 Aug 2022.
203. Kandel I, Castelli M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express.* 2020;6:312–5.
204. Radiuk PM. Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Inf Technol Manag Sci.* 2018;20.
205. Golmant N, Vemuri N, Yao Z, Feinberg V, Gholami A, Rothauge K, et al. On the Computational Inefficiency of Large Batch Sizes for Stochastic Gradient Descent. 2018. <http://arxiv.org/abs/1811.12941>. Accessed 29 Aug 2022.
206. Harmon SA, Sanford TH, Xu S, Turkbey EB, Roth H, Xu Z, et al. Artificial intelligence for the detection of COVID-19 pneumonia on chest CT using multinational datasets. *Nat Commun.* 2020;11:1–7. doi:10.1038/s41467-020-17971-2.
207. Shin HC, Roth HR, Gao M, Lu L, Xu Z, Nogues I, et al. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Trans Med Imaging.* 2016;35:1285–98. doi:10.1109/TMI.2016.2528162.
208. Karimi D, Warfield SK, Gholipour A. Transfer learning in medical image segmentation: New insights from analysis of the dynamics of model parameters and learned representations. *Artif Intell Med.* 2021;116:102078. doi:10.1016/j.artmed.2021.102078.
209. Kassem MA, Hosny KM, Fouad MM. Skin Lesions Classification into Eight Classes for ISIC 2019 Using Deep Convolutional Neural Network and Transfer Learning. *IEEE Access.* 2020;8 July:114822–32.
210. Ravishankar H, Sudhakar P, Venkataramani R, Thiruvankadam S, Annangi P, Babu N, et al. Understanding the mechanisms of deep transfer learning for medical images. In: *Deep Learning and Data Labeling for Medical Applications*. Springer Verlag; 2016. p. 188–96.
211. Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, et al. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Trans Med Imaging.* 2016;35:1299–312.
212. Anzanello MJ, Fogliatto FS. Learning curve models and applications: Literature review and research directions. *Int J Ind Ergon.* 2011;41:573–83.
213. Lin T-Y, Goyal P, Girshick R, He K, Dollár P. Focal Loss for Dense Object Detection. *IEEE Trans Pattern Anal Mach Intell.* 2017;42:318–27. <http://arxiv.org/abs/1708.02002>. Accessed 27 Feb 2021.
214. Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, et al. JAX: composable transformations of Python+NumPy programs. 2018. <http://github.com/google/jax>.
215. Perone CS, cclauss, Saravia E, Ballester PL, MohitTare. *MedicalTorch: A medical imaging framework for Pytorch.* 2018. doi:10.5281/ZENODO.1495335.
216. Mancolo F. Eisen: a python package for solid deep learning. 2020. <http://arxiv.org/abs/2004.02747>. Accessed 1 Sep 2022.
217. Baumgartner M, Jäger PF, Isensee F, Maier-Hein KH.

- nnDetection: A Self-configuring Method for Medical Object Detection. In: 24th international conference on medical image computing & computer assisted intervention (MICCA). Springer Science and Business Media Deutschland GmbH; 2021. p. 530–9.
218. Kaczmarzyk J, McClure P, Zulfikar W, Rana A, Rajaei H, Richie-Halford A, et al. Nobrainer: A framework for developing neural network models for 3D image processing. 2022. doi:10.5281/ZENODO.5838606.
219. GitHub. <https://github.com/>. Accessed 9 Jan 2022.
220. NVIDIA, Vingelmann P, Fitzek FHP. CUDA. 2007. <https://developer.nvidia.com/cuda-toolkit>.
221. The Khronos Group Inc. OpenCL The Khronos Group Inc. <https://www.khronos.org/opencl/>. Accessed 2 Sep 2022.
222. Beers A, Brown J, Chang K, Hoebel K, Patel J, Ly KI, et al. DeepNeuro: an open-source deep learning toolbox for neuroimaging. *Neuroinformatics*. 2021;19:127–40. doi:10.1007/s12021-020-09477-5.
223. Gros C, Lemay A, Vincent O, Rouhier L, Bourget M-H, Bucquet A, et al. ivadomed: A Medical Imaging Deep Learning Toolbox. *J Open Source Softw*. 2021;6:2868. doi:10.21105/joss.02868.
224. Results | KiTS21. <https://kits21.kits-challenge.org/results>. Accessed 2 Sep 2022.
225. Taha AA, Hanbury A. Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool. *BMC Med Imaging*. 2015;15:29. doi:10.1186/s12880-015-0068-x.
226. Müller D, Soto-Rey I, Kramer F. Towards a guideline for evaluation metrics in medical image segmentation. *BMC Res Notes*. 2022;15:210. doi:10.1186/s13104-022-06096-y.
227. Popovic A, de la Fuente M, Engelhardt M, Radermacher K. Statistical validation metric for accuracy assessment in medical image segmentation. *Int J Comput Assist Radiol Surg*. 2007;2:169–81. doi:10.1007/s11548-007-0125-1.
228. Lever J, Krzywinski M, Altman N. Points of Significance: Classification evaluation. Nature Publishing Group; 2016.
229. Eng J. Receiver operating characteristic analysis: A primer. *Acad Radiol*. 2005;12:909–16.
230. Kumar R V, Antony GM. A Review of Methods and Applications of the ROC Curve in Clinical Trials. *Drug Inf J*. 2010;44:659–71. doi:10.1177/009286151004400602.
231. Hanley JA, McNeil BJ. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*. 1982;143:29–36. doi:10.1148/radiology.143.1.7063747.
232. Cohen J. A Coefficient of Agreement for Nominal Scales. *Educ Psychol Meas*. 1960;20:37–46. doi:10.1177/001316446002000104.
233. Cohen's Kappa: what it is, when to use it, how to avoid pitfalls | KNIME. <https://www.knime.com/blog/cohens-kappa-an-overview>. Accessed 8 Jan 2022.
234. Delgado R, Tibau XA. Why Cohen's Kappa should be avoided as performance measure in classification. *PLoS One*. 2019;14:e0222916. doi:10.1371/journal.pone.0222916.
235. Aydin OU, Taha AA, Hilbert A, Khalil AA, Galinovic I, Fiebach JB, et al. On the usage of average Hausdorff distance for segmentation performance assessment: hidden error when used for ranking. *Eur Radiol Exp*. 2021;5:4. doi:10.1186/s41747-020-00200-2.
236. Karimi D, Salcudean SE. Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks. *IEEE Trans Med Imaging*. 2019;39:499–513. <http://arxiv.org/abs/1904.10030>. Accessed 8 Jan 2022.
237. Nai YH, Teo BW, Tan NL, O'Doherty S, Stephenson MC, Thian YL, et al. Comparison of metrics for the evaluation of medical segmentations using prostate MRI dataset. *Comput Biol Med*. 2021;134:104497.
238. Zhao H, Gallo O, Frosio I, Kautz J. Loss Functions for Neural Networks for Image Processing. 2015. <http://arxiv.org/abs/1511.08861>. Accessed 25 Aug 2022.
239. Jadon S. A survey of loss functions for semantic segmentation. In: 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2020. Institute of Electrical and Electronics Engineers Inc.; 2020.
240. Ma J, Chen J, Ng M, Huang R, Li Y, Li C, et al. Loss odyssey in medical image segmentation. *Med Image Anal*. 2021;71:102035.
241. Seyed SSM, Erdogmus D, Gholipour A, Salehi SSM, Erdogmus D, Gholipour A. Tversky loss function for image segmentation using 3D fully convolutional deep networks. In: *Lecture Notes in Computer Science*. Springer Verlag; 2017. p. 379–87. doi:10.1007/978-3-319-67389-9_44.
242. Janocha K, Czarnecki WM. On loss functions for deep neural networks in classification. *Schedae Informaticae*. 2016;25 Volume 25:49–59. doi:10.4467/20838476SI.16.004.6185.
243. Cooper Y. The loss landscape of overparameterized neural networks . In: *ICLR 2019 Conference* . 2019.
244. Rezaei-Dastjerdehei MR, Mijani A, Fatemizadeh E. Addressing Imbalance in Multi-Label Classification Using Weighted Cross Entropy Loss Function. In: 27th National and 5th International Iranian Conference of Biomedical Engineering, ICBME 2020. Institute of Electrical and Electronics Engineers Inc.; 2020. p. 333–8.
245. Taylor BJ. *Methods and Procedures for the Verification and Validation of Artificial Neural Networks*. Boston: Kluwer Academic Publishers; 2006. doi:10.1007/0-387-29485-6.
246. Raschka S. Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. 2018. <http://arxiv.org/abs/1811.12808>. Accessed 26 Aug 2022.
247. Qiao Z, Bae A, Glass LM, Xiao C, Sun J. FLANNEL: Focal Loss Based Neural Network Ensemble for COVID-19 Detection. Oxford University Press; 2021. doi:10.1093/jamia/ocaa280.
248. Hou X, Xie C, Li F, Nan Y. Cascaded Semantic Segmentation for Kidney and Tumor. 2019.
249. Khan R, Hanbury A, Stoeftinger J. Skin detection: A random forest approach. *Proc - Int Conf Image Process ICIP*. 2010;:4613–6.
250. Prechelt L. Early stopping - But when? *Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics)*. 2012;7700 LECTURE NO:53–67. doi:10.1007/978-3-642-35289-8_5.
251. Rosenfeld A, Kak A. *Digital Picture Processing*. Elsevier; 1982.
252. Sobel I. An Isotropic 3x3 Image Gradient Operator. In: *Presentation at Stanford A.I. Project* 1968. 2014.
253. Maintz JBA, Viergever MA. A survey of medical image registration. *Med Image Anal*. 1998;2:1–36. doi:10.1016/S1361-8415(01)80026-8.
254. Wells WM, Crimson WEL, Kikinis R, Jolesz FA. Adaptive segmentation of mri data. *IEEE Trans Med Imaging*. 1996;15:429–42. doi:10.1109/42.511747.
255. MacQueen JB. Some Methods for Classification and Analysis of MultiVariate Observations. In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press; 1967. p. 281–97.
256. Lloyd SP. Least Squares Quantization in PCM. 1982.

257. Ahn E, Kumar A, Feng D, Fulham M, Kim J. Unsupervised Feature Learning with K-means and An Ensemble of Deep Convolutional Neural Networks for Medical Image Classification. 2019. <http://arxiv.org/abs/1906.03359>. Accessed 9 Dec 2021.
258. Moriya T, Roth HR, Nakamura S, Oda H, Nagara K, Oda M, et al. Unsupervised segmentation of 3D medical images based on clustering and deep representation learning. In: Gimi B, Krol A, editors. *Medical Imaging 2018: Biomedical Applications in Molecular, Structural, and Functional Imaging*. SPIE; 2018. p. 71. doi:10.1117/12.2293414.
259. E. A. Arnya Arnya R, Mohsin Abdulazeez A. Medical Images Segmentation Based on Unsupervised Algorithms: A Review. *Qubahan Acad J*. 2021;1:71–80. doi:10.48161/qaj.v1n2a51.
260. McInerney T, Terzopoulos D. Deformable models in medical image analysis: a survey. *Med Image Anal*. 1996;1:91–108. doi:10.1016/S1361-8415(96)80007-7.
261. Guo Y, Liu Y, Georgiou T, Lew MS. A review of semantic segmentation using deep neural networks. *Int J Multimed Inf Retr*. 2018;7:87–93. doi:10.1007/s13735-017-0141-z.
262. Zhou Z, Siddiquee MMR, Tajbakhsh N, Liang J. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. 2018. <http://arxiv.org/abs/1807.10165>. Accessed 19 Jul 2019.
263. Cai L, Gao J, Zhao D. A review of the application of deep learning in medical image classification and segmentation. *Ann Transl Med*. 2020;8:713–713. doi:10.21037/atm.2020.02.44.
264. O'Reilly-Shah VN, Gentry KR, Walters AM, Zivot J, Anderson CT, Tighe PJ. Bias and ethical considerations in machine learning and the automation of perioperative risk assessment. *British Journal of Anaesthesia*. 2020;125:843–6. doi:10.1016/j.bja.2020.07.040.
265. Bhattacharya S, Reddy Maddikunta PK, Pham QV, Gadekallu TR, Krishnan S SR, Chowdhary CL, et al. Deep learning and medical image processing for coronavirus (COVID-19) pandemic: A survey. *Sustain Cities Soc*. 2021;65:102589.
266. Clark A. Pillow: The friendly PIL fork (Python Imaging Library). <https://github.com/python-pillow/Pillow>. Accessed 9 Sep 2022.
267. Brett M, Markiewicz CJ, Hanke M, Côté M-A, Cipollini B, McCarthy P, et al. nibabel: Python package to access a cacophony of neuro-imaging file formats. 2022. doi:10.5281/ZENODO.6658382.
268. Mason D. SU-E-T-33: Pydicom: An Open Source DICOM Library. *Med Phys*. 2011;38:3493–3493. doi:10.1118/1.3611983.
269. Lin G, Shen C, Hengel A Van Den, Reid I. Efficient Piecewise Training of Deep Structured Models for Semantic Segmentation. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*. 2016;2016-December:3194–203.
270. Isensee F, Maier-Hein KH. An attempt at beating the 3D U-Net. 2019;:1–7. <http://arxiv.org/abs/1908.02182>.
271. Coupé P, Manjón J V., Fonov V, Pruessner J, Robles M, Collins DL. Patch-based segmentation using expert priors: Application to hippocampus and ventricle segmentation. *Neuroimage*. 2011;54:940–54. doi:10.1016/j.neuroimage.2010.09.018.
272. Chollet, François, others. Keras. 2015. <https://keras.io>.
273. Chollet F. *Deep Learning with Python*. Manning Publications; 2017.
274. Fogel K. *Producing Open Source Software How to Run a Successful Free Software Project*. 2005. <http://producingoss.com/>. Accessed 12 Sep 2022.
275. The Git Project. Git. <https://git-scm.com/>. Accessed 12 Sep 2022.
276. The One DevOps Platform | GitLab. <https://about.gitlab.com/>. Accessed 12 Sep 2022.
277. Bitbucket | Git solution for teams using Jira. <https://bitbucket.org/>. Accessed 12 Sep 2022.
278. Hashemi Y, Nayebi M, Antoniol G. Documentation of Machine Learning Software. *SANER 2020 - Proc 2020 IEEE 27th Int Conf Softw Anal Evol Reengineering*. 2020;:666–7. <http://arxiv.org/abs/2001.11956>. Accessed 13 Sep 2022.
279. Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, et al. Jupyter Notebooks—a publishing format for reproducible computational workflows. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas - Proceedings of the 20th International Conference on Electronic Publishing, ELPUB 2016*. IOS Press BV; 2016. p. 87–90.
280. Travis CI – Start building today! <https://www.travis-ci.com/>. Accessed 13 Sep 2022.
281. Codecov - The Leading Code Coverage Solution. <https://about.codecov.io/>. Accessed 13 Sep 2022.
282. Python Software Foundation. Python: The Python programming language. <https://www.python.org/>. Accessed 12 Sep 2022.
283. Matthews J. A non-magical introduction to Pip and Virtualenv for Python beginners. 2013. <https://www.dabapps.com/blog/introduction-to-pip-and-virtualenv-python/>. Accessed 12 Sep 2022.
284. Linehaul Project, Python Packaging Authority. pypa/linehaul-cloud-function: Implementation of linehaul to feed the PyPI public BigQuery dataset via Google Cloud Functions. <https://github.com/pypa/linehaul-cloud-function/>. Accessed 14 Sep 2022.
285. Ferlay J, Colombet M, Soerjomataram I, Mathers C, Parkin DMM, Piñeros M, et al. Estimating the global cancer incidence and mortality in 2018: GLOBOCAN sources and methods. *Wiley-Liss Inc*. doi:10.1002/ijc.31937.
286. Khened M, Kori A, Rajkumar H, Krishnamurthi G, Srinivasan B. A generalized deep learning framework for whole-slide image segmentation and analysis. *Sci Rep*. 2021;11:1–14. doi:10.1038/s41598-021-90444-8.
287. Chan L, Hosseini MS, Rowsell C, Plataniotis KN, Damaskinos S, Edward Rogers Sr TS. HistoSegNet: Semantic Segmentation of Histological Tissue Type in Whole Slide Images. https://github.com/lyndonchan/hsn_v1. Accessed 12 May 2022.
288. Amgad M, Atteya LA, Hussein H, Mohammed KH, Hafiz E, Elsebaie MAT, et al. NuCLS: A scalable crowdsourcing approach and dataset for nucleus classification and segmentation in breast cancer. *Gigascience*. 2022;11:1–12. doi:10.1093/gigascience/giac037.
289. National Cancer Institute. The Cancer Genome Atlas Program - NCI. <https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga>. Accessed 6 Sep 2022.
290. Yeung M, Sala E, Schönlieb CB, Rundo L. Unified Focal loss: Generalising Dice and cross entropy-based losses to handle class imbalanced medical image segmentation. *Comput Med Imaging Graph*. 2022;95:102026.
291. Yeung M, Sala E, Schönlieb CB, Rundo L. Focus U-Net: A novel dual attention-gated CNN for polyp segmentation during colonoscopy. *Comput Biol Med*. 2021;137:104815.
292. King G, Zeng L. Logistic Regression in Rare Events Data. *Polit Anal*. 2001;9:137–63. doi:10.1093/oxfordjournals.pan.a004868.
293. Müller D, Hartmann D, Meyer P, Auer F, Soto-Rey I, Kramer F. MISeval: a Metric Library for Medical Image Segmentation Evaluation. In: *Studies in health technology and informatics*. Stud Health Technol Inform; 2022. doi:10.3233/shti220391.

294. He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. *IEEE Trans Pattern Anal Mach Intell.* 2020;42:386–97. doi:10.1109/TPAMI.2018.2844175.
295. Sohrai C, Alsafi Z, O'Neill N, Khan M, Kerwan A, Al-Jabir A, et al. World Health Organization declares global emergency: A review of the 2019 novel coronavirus (COVID-19). *Int J Surg.* 2020;76 February:71–6. doi:10.1016/j.ijsu.2020.02.034.
296. Shi F, Wang J, Shi J, Wu Z, Wang Q, Tang Z, et al. Review of Artificial Intelligence Techniques in Imaging Data Acquisition, Segmentation and Diagnosis for COVID-19. *IEEE Rev Biomed Eng.* 2020;:1–1.
297. Sasangohar F, Sasangohar F, Jones SL, Masud FN, Vahidy FS, Kash BA, et al. Provider Burnout and Fatigue during the COVID-19 Pandemic: Lessons Learned from a High-Volume Intensive Care Unit. *Anesth Analg.* 2020;:106–11. doi:10.1213/ANE.0000000000004866.
298. Bullock J, Luccioni A, Pham KH, Lam CSN, Luengo-Oroz M. Mapping the Landscape of Artificial Intelligence Applications against COVID-19. 2020;:1–32. <http://arxiv.org/abs/2003.11336>.
299. Dong E, Du H, Gardner L. An interactive web-based dashboard to track COVID-19 in real time. *Lancet Infect Dis.* 2020;20:533–4. doi:10.1016/S1473-3099(20)30120-1.
300. W. H. O. Coronavirus disease (COVID-19) pandemic. 2020. <https://www.who.int/emergencies/diseases/novel-coronavirus-2019>.
301. RKI - Coronavirus SARS-CoV-2 - SARS-CoV-2 Steckbrief zur Coronavirus-Krankheit-2019 (COVID-19). https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Steckbrief.html. Accessed 24 May 2020.
302. Amyar A, Modzelewski R, Ruan S. Multi-task Deep Learning Based CT Imaging Analysis For COVID-19: Classification and Segmentation. *medRxiv.* 2020;:2020.04.16.20064709. doi:<https://doi.org/10.1101/2020.04.16.20064709>.
303. Rodriguez-Morales AJ, Cardona-Ospina JA, Gutiérrez-Ocampo E, Villamizar-Peña R, Holguin-Rivera Y, Escalera-Antezana JP, et al. Clinical, laboratory and imaging features of COVID-19: A systematic review and meta-analysis. *Travel Med Infect Dis.* 2020;34 February. doi:10.1016/j.tmaid.2020.101623.
304. Salehi S, Abedi A, Balakrishnan S, Gholamrezaezhad A. Coronavirus Disease 2019 (COVID-19): A Systematic Review of Imaging Findings in 919 Patients. *Am J Roentgenol.* 2020;215:87–93. doi:10.2214/AJR.20.23034.
305. Rubin GD, Ryerson CJ, Haramati LB, Sverzellati N, Kanne JP, Raouf S, et al. The Role of Chest Imaging in Patient Management During the COVID-19 Pandemic. *Radiology.* 2020;158:106–16. doi:10.1148/radiol.2020201365.
306. Ai T, Yang Z, Hou H, Zhan C, Chen C, Lv W, et al. Correlation of Chest CT and RT-PCR Testing in Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases. *Radiology.* 2020;:23.
307. Ng M-Y, Lee EY, Yang J, Yang F, Li X, Wang H, et al. Imaging Profile of the COVID-19 Infection: Radiologic Findings and Literature Review Authors: Radiol Cardiothorac Imaging. 2020;2. doi:10.1148/ryct.2020200034.
308. Fang Y, Zhang H, Xie J, Lin M, Ying L, Pang P, et al. Sensitivity of Chest CT for COVID-19: Comparison to RT-PCR. *Radiology.* 2020;:8. doi:10.1148/radiol.2020200432.
309. An P, Xu S, Harmon S, Turkbey E, Sanford T, Amalou A, et al. CT Images in COVID-19 - The Cancer Imaging Archive (TCIA). 2020. doi:<https://doi.org/10.7937/tcia.2020.gqry-nc81>.
310. Gozes O, Frid-Adar M, Greenspan H, Browning PD, Bernheim A, Siegel E. Rapid AI Development Cycle for the Coronavirus (COVID-19) Pandemic: Initial Results for Automated Detection & Patient Monitoring using Deep Learning CT Image Analysis. 2020. <http://arxiv.org/abs/2003.05037>.
311. Chen X, Yao L, Zhang Y. Residual Attention U-Net for Automated Multi-Class Segmentation of COVID-19 Chest CT Images. 2020;14:1–7. <http://arxiv.org/abs/2004.05645>.
312. Gozes O, Frid-Adar M, Sagie N, Zhang H, Ji W, Greenspan H. Coronavirus Detection and Analysis on Chest CT with Deep Learning. 2020;:1–8. <http://arxiv.org/abs/2004.02640>.
313. Zhou T, Canu S, Ruan S. An automatic COVID-19 CT segmentation based on U-Net with attention mechanism. 2020;:1–14. <http://arxiv.org/abs/2004.06673>.
314. Saood A, Hatem I. COVID-19 lung CT image segmentation using deep learning methods: U-Net versus SegNet. *BMC Med Imaging.* 2021;21:19. doi:10.1186/s12880-020-00529-5.
315. Zheng B, Liu Y, Zhu Y, Yu F, Jiang T, Yang D, et al. Msdnet: Multi-scale discriminative network for covid-19 lung infection segmentation on CT. *IEEE Access.* 2020;8:185786–95.
316. Fan D-P, Zhou T, Ji G-P, Zhou Y, Chen G, Fu H, et al. InfNet: Automatic COVID-19 Lung Infection Segmentation from CT Scans. *IEEE Trans Med Imaging.* 2020;2019:1–11. doi:10.1109/tmi.2020.2996645.
317. He K, Zhao W, Xie X, Ji W, Liu M, Tang Z, et al. Synergistic Learning of Lung Lobe Segmentation and Hierarchical Multi-Instance Classification for Automated Severity Assessment of COVID-19 in CT Images. 2020. <http://arxiv.org/abs/2005.03832>. Accessed 2 Nov 2020.
318. Qiu Y, Liu Y, Xu J. MiniSeg: An Extremely Minimum Network for Efficient COVID-19 Segmentation. 2020;:1–10. <http://arxiv.org/abs/2004.09750>.
319. Wang Y, Zhang Y, Liu Y, Tian J, Zhong C, Shi Z, et al. Does non-COVID-19 lung lesion help? investigating transferability in COVID-19 CT image segmentation. *Comput Methods Programs Biomed.* 2021;202:106004.
320. Roy S, Carass A, Prince JL. Patch based intensity normalization of brain MR images. In: 2013 IEEE 10th International Symposium on Biomedical Imaging. *IEEE;* 2013. p. 342–5. doi:10.1109/ISBI.2013.6556482.
321. Yan Q, Wang B, Gong D, Luo C, Zhao W, Shen J, et al. COVID-19 Chest CT Image Segmentation -- A Deep Convolutional Neural Network Solution. 2020;:1–10. <http://arxiv.org/abs/2004.10987>.
322. Italian Society of Medical and Interventional Radiology. COVID-19 - Medical segmentation. 2020. <http://medicalsegmentation.com/covid19/>. Accessed 29 May 2021.
323. Busch C, Gross MH. Interactive Neural Network Texture Analysis and Visualization for Surface Reconstruction in Medical Imaging. *Comput Graph Forum.* 1993;12:49–60. doi:10.1111/1467-8659.1230049.
324. Van Der Geest RJ, Lelieveldt BPF, Reiber JHC. Quantification of global and regional ventricular function in cardiac magnetic resonance imaging. *Topics in Magnetic Resonance Imaging.* 2000;11:348–58. doi:10.1097/00002142-200012000-00004.
325. Doi K. Computer-Aided Diagnosis in Medical Imaging: Historical Review, Current Status and Future Potential. *Comput Med Imaging Graph.* 2007;31:198–211.
326. Russell SJ, Norvig P. Artificial intelligence: a modern approach. 3. Edition. Pearson Education, Inc.; 2010.
327. Tommasi T, Patricia N, Caputo B, Tuytelaars T. A deeper look at dataset bias. In: *Advances in Computer Vision and Pattern Recognition.* Springer London; 2017. p. 37–55. doi:10.1007/978-3-319-58347-1_2.
328. Li Z, Xu C. Discover the Unknown Biased Attribute of an Image Classifier. <https://git.io/J3kMh>. Accessed 20 Sep 2022.
329. Bressem KK, Adams LC, Erxleben C, Hamm B, Niehues

- SM, Vahldiek JL. Comparing different deep learning architectures for classification of chest radiographs. *Sci Rep.* 2020;10:13590. doi:10.1038/s41598-020-70479-z.
330. Fischer A, Klein P, Radulescu P, Gulsun M, Mohamed Ali A, R S V, et al. Deep Learning Based Automated Coronary Labeling For Structured Reporting Of Coronary CT Angiography In Accordance With SCCT Guidelines. *J Cardiovasc Comput Tomogr.* 2020;14:S21–2. doi:10.1016/j.jcct.2020.06.019.
331. Garcia E V., Klein JL, Moncayo V, Cooke CD, Del'Aune C, Folks R, et al. Diagnostic performance of an artificial intelligence-driven cardiac-structured reporting system for myocardial perfusion SPECT imaging. *J Nucl Cardiol.* 2020;27:1652–64. doi:10.1007/s12350-018-1432-3.
332. Fanni SC, Gabelloni M, Alberich-Bayarri A, Neri E. Structured Reporting and Artificial Intelligence. Springer, Cham; 2022. p. 169–83. doi:10.1007/978-3-030-91349-6_8.
333. Rajchl M, Koch LM, Ledig C, Passerat-Palmbach J, Misawa K, Mori K, et al. Employing Weak Annotations for Medical Image Analysis Problems. 2017. <http://arxiv.org/abs/1708.06297>. Accessed 28 Aug 2022.
334. Corbett-Davies S, Goel S. The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning. 2018. <http://arxiv.org/abs/1808.00023>. Accessed 20 Sep 2022.
335. Solovyev R, Ashawkey, Iakubovskii P, Müller D. Volumentations: Library for 3D augmentations. <https://github.com/ZFTurbo/volumentations>. Accessed 26 Sep 2022.
336. Solovyev R, Kalinin AA, Gabruseva T. 3D Convolutional Neural Networks for Stalled Brain Capillary Detection. 2021. <http://arxiv.org/abs/2104.01687>. Accessed 24 Nov 2021.
337. He K, Zhang X, Ren S, Sun J. Identity mappings in deep residual networks. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag; 2016. p. 630–45. doi:10.1007/978-3-319-46493-0_38.
338. Ganaie MA, Hu M, Malik AK, Tanveer M, Suganthan PN. Ensemble deep learning: A review. *Eng Appl Artif Intell.* 2021;115. doi:10.1016/j.engappai.2022.105151.
339. Sagi O, Rokach L. Ensemble learning: A survey. *WIREs Data Min Knowl Discov.* 2018;8. doi:10.1002/widm.1249.
340. Ju C, Bibaut A, van der Laan M. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *J Appl Stat.* 2018;45:2800–18.
341. Kandel I, Castelli M, Popovič A. Comparing Stacking Ensemble Techniques to Improve Musculoskeletal Fracture Image Classification. *J Imaging.* 2021;7:100. doi:10.3390/jimaging7060100.
342. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in {P}ython. *J Mach Learn Res.* 2011;12:2825–30.
343. van der Velden BHM, Kuijff HJ, Gilhuijs KGA, Viergever MA. Explainable artificial intelligence (XAI) in deep learning-based medical image analysis. *Medical Image Analysis.* 2022;79:102470.
344. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *Int J Comput Vis.* 2016;128:336–59. doi:10.1007/s11263-019-01228-7.
345. Chattopadhyay A, Sarkar A, Howlader P, Balasubramanian VN. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In: *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*. Institute of Electrical and Electronics Engineers Inc.; 2018. p. 839–47.
346. Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M. Striving for Simplicity: The All Convolutional Net. 3rd Int Conf Learn Represent ICLR 2015 - Work Track Proc. 2014. <http://arxiv.org/abs/1412.6806>. Accessed 27 Sep 2022.
347. Simonyan K, Vedaldi A, Zisserman A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. 2nd Int Conf Learn Represent ICLR 2014 - Work Track Proc. 2013. <http://arxiv.org/abs/1312.6034>. Accessed 8 Apr 2022.
348. Sundararajan M, Taly A, Yan Q. Axiomatic Attribution for Deep Networks. 34th Int Conf Mach Learn ICML 2017. 2017;7:5109–18. <http://arxiv.org/abs/1703.01365>. Accessed 27 Sep 2022.
349. Ribeiro M, Singh S, Guestrin C. “Why Should I Trust You?”: Explaining the Predictions of Any Classifier. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Stroudsburg, PA, USA: Association for Computational Linguistics; 2016. p. 97–101. doi:10.18653/v1/N16-3020.
350. Hykes S, Docker Inc. Docker. 2013. <https://www.docker.com/>. Accessed 22 Sep 2022.
351. Nüst D, Sochat V, Marwick B, Eglén SJ, Head T, Hirst T, et al. Ten simple rules for writing Dockerfiles for reproducible data science. *PLOS Comput Biol.* 2020;16:e1008316. doi:10.1371/journal.pcbi.1008316.
352. Zhang B. An Explorative Study of GitHub Repositories of AI Papers. 2019. <http://arxiv.org/abs/1903.01555>. Accessed 23 Sep 2022.
353. Cito J, Ferme V, Gall HC. Using docker containers to improve reproducibility in software and web engineering research. In: *International Conference on Web Engineering*. Springer Verlag; 2016. p. 609–12. doi:10.1007/978-3-319-38791-8_58.
354. Norgeot B, Quer G, Beaulieu-Jones BK, Torkamani A, Dias R, Gianfrancesco M, et al. Minimum information about clinical artificial intelligence modeling: the MI-CLAIM checklist. *Nature Medicine.* 2020;26:1320–4. doi:10.1038/s41591-020-1041-y.
355. Conventional Commits. <https://www.conventionalcommits.org/en/v1.0.0/>. Accessed 29 Sep 2022.
356. Christie T, MkDocs Team. MkDocs. <https://www.mkdocs.org/>. Accessed 29 Sep 2022.
357. Mazzucotelli T. mkdocstrings: Automatic documentation from sources, for MkDocs. <https://github.com/mkdocstrings/mkdocstrings>. Accessed 29 Sep 2022.
358. Google LLC. Style guides for Google-originated open-source projects. <https://google.github.io/styleguide/pyguide.html>. Accessed 29 Sep 2022.
359. black: The uncompromising Python code formatter. <https://github.com/psf/black>. Accessed 30 Sep 2022.
360. flake8: a python tool that glues together pycodestyle, pyflakes, mccabe, and third-party plugins to check the style and quality of some python code. <https://github.com/PyCQA/flake8>. Accessed 30 Sep 2022.
361. pylint: It’s not just a linter that annoys you! <https://github.com/PyCQA/pylint>. Accessed 30 Sep 2022.
362. Virmani M. Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In: *5th International Conference on Innovative Computing Technology, INTECH 2015*. Institute of Electrical and Electronics Engineers Inc.; 2015. p. 78–82.
363. Chen L. Continuous delivery: Huge benefits, but challenges too. *IEEE Softw.* 2015;32:50–4.
364. Python Software Foundation. Python Package Index - PyPI.

- <https://pypi.org/>.
365. Martin RC. *Agile Software Development: Principles, Patterns, and Practices*. 2nd edition. Pearson Education; 2003.
366. Maceachern SJ, Forkert ND. Machine learning for precision medicine. *Genome*. 2021;64:416–25. doi:10.1139/gen-2020-0131.
367. Borgli H, Thambawita V, Smedsrud PH, Hicks S, Jha D, Eskeland SL, et al. HyperKvasir, a comprehensive multi-class image and video dataset for gastrointestinal endoscopy. *Sci Data*. 2020;7:16. doi:10.1038/s41597-020-00622-y.
368. Janowczyk A, Madabhushi A. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. *J Pathol Inform*. 2016;7. doi:10.4103/2153-3539.186902.
369. Cruz-Roa A, Basavanahally A, González F, Gilmore H, Feldman M, Ganesan S, et al. Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. In: *Medical Imaging 2014: Digital Pathology*. SPIE; 2014. p. 904103.
370. Nickparvar M. Brain Tumor MRI Dataset [Kaggle]. 2021. doi:10.34740/kaggle/dsv/2645886.
371. Ning W, Lei S, Yang J, Cao Y, Jiang P, Yang Q, et al. Open resource of clinical data from patients with pneumonia for the prediction of COVID-19 outcomes via deep learning. *Nat Biomed Eng*. 2020;4:1197–207. doi:10.1038/s41551-020-00633-5.
372. Warszawik-Hendzel O, Olszewska M, Maj M, Rakowska A, Czuwara J, Rudnicka L. Non-invasive diagnostic techniques in the diagnosis of squamous cell carcinoma. *J Dermatol Case Rep*. 2015;9:89–97. doi:10.3315/jdcr.2015.1221.
373. Adelson JD, Bourne RRA, Briant PS, Flaxman SR, Taylor HRB, Jonas JB, et al. Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to VISION 2020: the Right to Sight: an analysis for the Global Burden of Disease Study. *Lancet Glob Heal*. 2021;9:e144–60. doi:10.1016/S2214-109X(20)30489-7.
374. World Health Organization. Blindness and vision impairment. <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>. Accessed 27 Feb 2021.
375. Choi JY, Yoo TK, Seo JG, Kwak J, Um TT, Rim TH. Multi-categorical deep learning neural network to classify retinal images: A pilot study employing small database. *PLoS One*. 2017;12:e0187336. doi:10.1371/journal.pone.0187336.
376. Quellec G, Lamard M, Conze PH, Massin P, Cochener B. Automatic detection of rare pathologies in fundus photographs using few-shot learning. *Med Image Anal*. 2020;61:101660.
377. Zhu C, Byrd RH, Lu P, Nocedal J. L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. *ACM Trans Math Softw*. 1997;23:550–60. doi:10.1145/279232.279236.
378. Kaur P, Gosain A. Issues and challenges of class imbalance problem in classification. *Int J Inf Technol*. 2020;:1–7. doi:10.1007/s41870-018-0251-8.
379. Gao L, Zhang L, Liu C, Wu S. Handling imbalanced medical image data: A deep-learning-based one-class classification approach. *Artif Intell Med*. 2020;108:101935.
380. Zhengbo L, Zitang S, Qjhan L, Kamata S. Diverse Blending DNNs for Multi-Disease Detection. In: *Conference Presentation at the IEEE ISBI 2021 International Symposium on Biomedical Imaging*. 2021.
381. Ho E, Wang E, Sivajohan A, Youn S, Lane K, Chun J. Deep Ensemble Learning for Retinal Image Classification. In: *Conference Presentation at the IEEE ISBI 2021 International Symposium on Biomedical Imaging*. 2021.
382. Hu J, Shen L, Albanie S, Sun G, Wu E. Squeeze-and-Excitation Networks. *IEEE Trans Pattern Anal Mach Intell*. 2020;42:2011–23. doi:10.1109/TPAMI.2019.2913372.
383. Wang X, Gu Y, Lu S. Bag of Tricks for Multi-Disease Detection of Retina Images. In: *Conference Presentation at the IEEE ISBI 2021 International Symposium on Biomedical Imaging*. 2021.
384. Jin C, Chen W, Cao Y, Xu Z, Tan Z, Zhang X, et al. Development and evaluation of an artificial intelligence system for COVID-19 diagnosis. *Nat Commun*. 2020;11:1–14. doi:10.1038/s41467-020-18685-1.
385. Abbas A, Abdelsamea MM, Gaber MM. Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. *Appl Intell*. 2021;51:854–64. doi:10.1007/s10489-020-01829-7.
386. Tang Z, Zhao W, Xie X, Zhong Z, Shi F, Ma T, et al. Severity assessment of COVID-19 using CT image features and laboratory indices. *Phys Med Biol*. 2021;66:035015. doi:10.1088/1361-6560/abbf9e.
387. Silvani P, Colombo S, Scandroglio AM, Dell’Acqua A, Fominskiy E, Monti G, et al. Fast reshaping of intensive care unit facilities in a large metropolitan hospital in Milan, Italy: facing the COVID-19 pandemic emergency. *Crit Care Resusc*. 2020;22:91–4.
388. Goh KJ, Wong J, Tien JCC, Ng SY, Duu Wen S, Phua GC, et al. Preparing your intensive care unit for the COVID-19 pandemic: Practical considerations and strategies. *Critical Care*. 2020;24:215. doi:10.1186/s13054-020-02916-4.
389. Hôpitaux de Paris, Center Radboud University Medical. Stoic 2021 Challenge - Grand Challenge. 2021. <https://stoic2021.grand-challenge.org/stoic2021/>. Accessed 23 Sep 2022.
390. Wang X, Che Q, Ji X, Meng X, Zhang L, Jia R, et al. Correlation between lung infection severity and clinical laboratory indicators in patients with COVID-19: a cross-sectional study based on machine learning. *BMC Infect Dis*. 2021;21:192. doi:10.1186/s12879-021-05839-9.
391. Xiong Y, Sun D, Liu Y, Fan Y, Zhao L, Li X, et al. Clinical and High-Resolution CT Features of the COVID-19 Infection: Comparison of the Initial and Follow-up Changes. *Invest Radiol*. 2020;55:332–9. doi:10.1097/RLI.0000000000000674.
392. Raoufi M, Ahmad Safavi Naini SA, Azizan Z, Jafar Zade F, Shojaeian F, Boroujeni MG, et al. Correlation between Chest Computed Tomography Scan Findings and Mortality of COVID-19 Cases: a Cross sectional Study. *Arch Acad Emerg Med*. 2020;8:e57. doi:10.22037/aaem.v8i1.719.
393. Jungo A, Scheidegger O, Reyes M, Balsiger F. pymia: A Python package for data handling and evaluation in deep learning-based medical image analysis. *Comput Methods Programs Biomed*. 2021;198:105796.
394. Steiner DF, Nagpal K, Sayres R, Foote DJ, Wedin BD, Pearce A, et al. Evaluation of the Use of Combined Artificial Intelligence and Pathologist Assessment to Review and Grade Prostate Biopsies. *JAMA Netw Open*. 2020;3:2023267. doi:10.1001/jamanetworkopen.2020.23267.
395. Nagpal K, Foote D, Liu Y, Chen PHC, Wulczyn E, Tan F, et al. Development and validation of a deep learning algorithm for improving Gleason scoring of prostate cancer. *npj Digit Med*. 2019;2:48. doi:10.1038/s41746-019-0112-2.
396. Robert Koch Institut. Krebs - Prostatakrebs (ICD-10 C61). 2019. https://www.krebsdaten.de/Krebs/DE/Content/Krebsarten/Prostatatakrebs/prostatatakrebs_node.html. Accessed 8 Nov 2022.
397. Gleason DF. Classification of prostatic carcinomas. *Cancer Chemother reports*. 1966;50:125–8.
398. Singh A, Sengupta S, Lakshminarayanan V. Explainable

- Deep Learning Models in Medical Image Analysis. *J Imaging*. 2020;6:52. doi:10.3390/jimaging6060052.
399. Talmon J, Ammenwerth E, Brender J, de Keizer N, Nykänen P, Rigby M. STARE-HI-Statement on reporting of evaluation studies in Health Informatics. *Int J Med Inform*. 2009;78:1–9.
400. Yang Y, Hu Y, Zhang X, Wang S. Two-Stage Selective Ensemble of CNN via Deep Tree Training for Medical Image Classification. *IEEE Trans Cybern*. 2021;:1–14. doi:10.1109/TCYB.2021.3061147.
401. Logan R, Williams BG, Ferreira da Silva M, Indani A, Scholnicov N, Ganguly A, et al. Deep Convolutional Neural Networks With Ensemble Learning and Generative Adversarial Networks for Alzheimer's Disease Image Data Classification. *Front Aging Neurosci*. 2021;13. doi:10.3389/fnagi.2021.720226.
402. Rajaraman S, Siegelman J, Alderson PO, Folio LS, Folio LR, Antani SK. Iteratively Pruned Deep Learning Ensembles for COVID-19 Detection in Chest X-rays. <https://twitter.com/ChestImaging>. Accessed 26 Nov 2020.
403. Mohammed M, Mwambi H, Mboya IB, Elbashir MK, Omolo B. A stacking ensemble deep learning approach to cancer type classification based on TCGA data. *Sci Rep*. 2021;11:15626. doi:10.1038/s41598-021-95128-x.
404. Das A, Mohapatra SK, Mohanty MN. Design of deep ensemble classifier with fuzzy decision method for biomedical image classification. *Appl Soft Comput*. 2022;115:108178. doi:10.1016/j.asoc.2021.108178.
405. Sułot D, Alonso-Caneiro D, Ksieniewicz P, Krzyzanowska-Berkowska P, Iskander DR. Glaucoma classification based on scanning laser ophthalmoscopic images using a deep learning ensemble method. *PLoS One*. 2021;16:e0252339. doi:10.1371/journal.pone.0252339.
406. Zhang J, Wang Y, Sun Y, Li G. Strength of ensemble learning in multiclass classification of rockburst intensity. *Int J Numer Anal Methods Geomech*. 2020;44:1833–53.
407. Rajaraman S, Zamzmi G, Antani SK. Novel loss functions for ensemble-based medical image classification. *PLoS One*. 2021;16:e0261307. doi:10.1371/journal.pone.0261307.
408. Galdran A, Carneiro G, González Ballester MA. Balanced-MixUp for Highly Imbalanced Medical Image Classification. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Science and Business Media Deutschland GmbH; 2021. p. 323–33. doi:10.1007/978-3-030-87240-3_31.
409. Calvo-Zaragoza J, Rico-Juan JR, Gallego AJ. Ensemble classification from deep predictions with test data augmentation. *Soft Comput*. 2020;24:1423–33. doi:10.1007/s00500-019-03976-7.
410. Shanmugam D, Blalock D, Balakrishnan G, Guttag J. Better Aggregation in Test-Time Augmentation. 2020. <http://arxiv.org/abs/2011.11156>. Accessed 10 Dec 2021.
411. Seçkin Ayhan M, Berens P. Test-time Data Augmentation for Estimation of Heteroscedastic Aleatoric Uncertainty in Deep Neural Networks. In: *International conference on Medical Imaging with Deep Learning (MIDL) 2018*. 2018.
412. Kandel I, Castelli M. Improving convolutional neural networks performance for image classification using test time augmentation: a case study using MURA dataset. *Heal Inf Sci Syst*. 2021;9:33. doi:10.1007/s13755-021-00163-7.
413. Molchanov D, Lyzhov A, Molchanova Y, Ashukha A, Vetrov D. Greedy Policy Search: A Simple Baseline for Learnable Test-Time Augmentation. In: Peters, Jonas and Sontag D, editor. *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR; 2020. p. 1308–1317.
414. Ghogh B, Crowley M. The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial. 2019. <http://arxiv.org/abs/1905.12787>. Accessed 17 Dec 2021.
415. Valverde S, Cabezas M, Roura E, González-Villà S, Pareto D, Vilanova JC, et al. Improving automated multiple sclerosis lesion segmentation with a cascaded 3D convolutional neural network approach. *Neuroimage*. 2017;155:159–68. doi:10.1016/j.neuroimage.2017.04.034.
416. Zhang Y, Lai H, Yang W. Cascade UNet and CH-UNet for Thyroid Nodule Segmentation and Benign and Malignant Classification. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Science and Business Media Deutschland GmbH; 2021. p. 129–34. doi:10.1007/978-3-030-71827-5_17.
417. Tang M, Zhang Z, Cobzas D, Jagersand M, Jaremko JL. Segmentation-by-detection: A cascade network for volumetric medical image segmentation. In: *Proceedings - International Symposium on Biomedical Imaging*. IEEE Computer Society; 2018. p. 1356–9.
418. Chang C-C, Lin C-J. LIBSVM: A Library for Support Vector Machines. www.csie.ntu.edu.tw/. Accessed 4 Nov 2021.
419. Sollich P, Krogh A. Learning with ensembles: How overfitting can be useful. doi:doi/10.5555/2998828.2998855.
420. Kuncheva LI, Whitaker CJ. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach Learn*. 2003;51:181–207. doi:10.1023/A:1022859003006.
421. Cao Y, Geddes TA, Yang JYH, Yang P. Ensemble deep learning in bioinformatics. *Nature Machine Intelligence*. 2020;2:500–8. doi:10.1038/s42256-020-0217-y.
422. An P, Xu S, Harmon S, Turkbey E, Sanford T, Amalou A, et al. CT Images in Covid-19 [Data set]. *The Cancer Imaging Archive*. 2020. doi:https://doi.org/10.7937/ncia.2020.gqry-nc81.
423. Roth H, Xu Z, Diez CT, Jacob RS, Zember J, Molto J, et al. Rapid Artificial Intelligence Solutions in a Pandemic - The COVID-19-20 Lung CT Lesion Segmentation Challenge. *Res Sq*. 2021. doi:10.21203/rs.3.rs-571332/v1.
424. Kather JN, Weis CA, Bianconi F, Melchers SM, Schad LR, Gaiser T, et al. Multi-class texture analysis in colorectal cancer histology. *Sci Rep*. 2016;6. doi:10.1038/srep27988.
425. Kather JN, Zöllner FG, Bianconi F, Melchers SM, Schad LR, Gaiser T, et al. Collection of textures in colorectal cancer histology. 2016. doi:10.5281/ZENODO.53169.
426. Chowdhury MEH, Rahman T, Khandakar A, Mazhar R, Kadir MA, Mahub Z Bin, et al. Can AI Help in Screening Viral and COVID-19 Pneumonia? *IEEE Access*. 2020;8:132665–76.
427. Rahman T, Khandakar A, Qiblawey Y, Tahir A, Kiranyaz S, Abul Kashem S Bin, et al. Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images. *Comput Biol Med*. 2021;132:104319.
428. Diabetic Retinopathy Detection | Kaggle. <https://www.kaggle.com/c/diabetic-retinopathy-detection/overview>. Accessed 29 Oct 2021.
429. Cuadros J, Bresnick G. EyePACS: An adaptable telemedicine system for diabetic retinopathy screening. *J Diabetes Sci Technol*. 2009;3:509–16. doi:10.1177/193229680900300315.
430. Breiman L. Random forests. *Mach Learn*. 2001;45:5–32. doi:10.1023/A:1010933404324.
431. Rennie JDM, Shih L, Teevan J, Karger DR. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: *Proceedings of the Twentieth International Conference on Machine Learning*. 2003. p. 616–623.
432. McKinney W. Data Structures for Statistical Computing in Python. In: *Proceedings of the 9th Python in Science Conference*.

2010. p. 56–61. doi:10.25080/Majora-92bf1922-00a.
433. Kibirige H, Lamp G, Katins J, gdowning, austin, matthias-k, et al. Plotmine. 2021. doi:10.5281/ZENODO.4636791.
434. Liu Y, Long F. Acute lymphoblastic leukemia cells image analysis with deep bagging ensemble learning. In: *Lecture Notes in Bioengineering*. Springer; 2019. p. 113–21. doi:10.1007/978-981-15-0798-4_12.
435. Dwork C, Feldman V, Hardt M, Pitassi T, Reingold O, Roth A. Generalization in Adaptive Data Analysis and Holdout Reuse. *Adv Neural Inf Process Syst*. 2015;2015-January:2350–8. <http://arxiv.org/abs/1506.02629>. Accessed 12 Dec 2021.
436. Xu Y, Goodacre R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. *J Anal Test*. 2018;2:249–62. doi:10.1007/s41664-018-0068-2.
437. Geman S, Bienenstock E, Doursat R. Neural Networks and the Bias/Variance Dilemma. *Neural Comput*. 1992;4:1–58.
438. Yang Z, Yu Y, You C, Steinhart J, Ma Y. Rethinking Bias-Variance Trade-off for Generalization of Neural Networks. 37th Int Conf Mach Learn ICML 2020. 2020;PartF168147-14:10698–708. <http://arxiv.org/abs/2002.11328>. Accessed 12 Dec 2021.
439. Neal B, Mittal S, Baratin A, Tantia V, Scicluna M, Lacoste-Julien S, et al. A Modern Take on the Bias-Variance Tradeoff in Neural Networks. 2018. <http://arxiv.org/abs/1810.08591>. Accessed 12 Dec 2021.
440. Gibson E, Hu Y, Huisman HJ, Barratt DC. Designing image segmentation studies: Statistical power, sample size and reference standard quality. *Med Image Anal*. 2017;42:44–59.
441. Niessen WJ, Bouma CJ, Vincken KL, Viergever MA. Error Metrics for Quantitative Evaluation of Medical Image Segmentation. Springer, Dordrecht; 2000. p. 275–84. doi:10.1007/978-94-015-9538-4_22.
442. Shaikh F, Dehmeshki J, Bisdas S, Roettger-Dupont D, Kubassova O, Aziz M, et al. Artificial Intelligence-Based Clinical Decision Support Systems Using Advanced Medical Imaging and Radiomics. *Curr Probl Diagn Radiol*. 2021;50:262–7.
443. Pedersen M, Verspoor K, Jenkinson M, Law M, Abbott DF, Jackson GD. Artificial intelligence for clinical decision support in neurology. *Brain Commun*. 2020;2. doi:10.1093/braincomms/fcaa096.
444. Chen H, Sung JY. Potentials of AI in medical image analysis in Gastroenterology and Hepatology. *J Gastroenterol Hepatol*. 2021;36:31–8. doi:10.1111/jgh.15327.
445. Dyck A, Penners R, Lichter H. Towards Definitions for Release Engineering and DevOps. In: 2015 IEEE/ACM 3rd International Workshop on Release Engineering. IEEE; 2015. p. 3–3. doi:10.1109/RELENG.2015.10.
446. Detlefsen N, Borovec J, Schock J, Jha A, Koker T, Di Liello L, et al. TorchMetrics - Measuring Reproducibility in PyTorch. *J Open Source Softw*. 2022;7:4101. doi:10.21105/joss.04101.
447. Zenodo - Research. Shared. <https://zenodo.org/>. Accessed 9 Jan 2022.
448. Müller D, Soto-Rey II, Kramer F. An Analysis on Ensemble Learning optimized Medical Image Classification with Deep Convolutional Neural Networks. *IEEE Access*. 2022;:1–1. doi:10.1109/access.2022.3182399.
449. Meyer P, Müller D, Soto-Rey I, Kramer F. COVID-19 image segmentation based on deep learning and ensemble learning. In: *Public Health and Informatics: Proceedings of MIE 2021*. IOS Press; 2021. p. 518–9. doi:10.3233/SHTI210223.
450. Abraham N, Khan NM. A Novel Focal Tversky Loss Function With Improved Attention U-Net for Lesion Segmentation. In: 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). IEEE; 2019. p. 683–7. doi:10.1109/ISBI.2019.8759329.
451. Caicedo JC, Goodman A, Karhohs KW, Cimini BA, Ackerman J, Haghghi M, et al. Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl. *Nat Methods*. 2019;16:1247–53. doi:10.1038/s41592-019-0612-7.
452. Introduction - Grand Challenge. <https://drive.grand-challenge.org/DRIVE/>. Accessed 12 May 2022.
453. Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*. 2020;21:6. doi:10.1186/s12864-019-6413-7.
454. Weerakkody Y, Hattingh C. Sagittal midline of the brain (an approach). *Radiopaedia.org*; 2015. doi:10.53347/rid-38058.
455. Orlando JI, Prokofyeva E, Blaschko MB. A Discriminatively Trained Fully Connected Conditional Random Field Model for Blood Vessel Segmentation in Fundus Images. *IEEE Trans Biomed Eng*. 2017;64:16–27.
456. Srinivasu PN, Balas VE. Self-Learning Network-based segmentation for real-time brain M.R. images through HARIS. *PeerJ Comput Sci*. 2021;7:1–24. doi:10.7717/PEERJ-CS.654.
457. Setiawan AW. Image Segmentation Metrics in Skin Lesion: Accuracy, Sensitivity, Specificity, Dice Coefficient, Jaccard Index, and Matthews Correlation Coefficient. In: *CENIM 2020 - Proceeding: International Conference on Computer Engineering, Network, and Intelligent Multimedia 2020*. Institute of Electrical and Electronics Engineers Inc.; 2020. p. 97–102.
458. Isensee F. Meetup: The promises of AI in medicine. 2019.
459. Sarma K V, Harmon S, Sanford T, Roth HR, Xu Z, Tetreault J, et al. Federated learning improves site performance in multicenter deep learning without data sharing. *J Am Med Informatics Assoc*. 2021;28:1259–64. doi:10.1093/jamia/ocaa341.
460. Li L, Fan Y, Tse M, Lin KY. A review of applications in federated learning. *Comput Ind Eng*. 2020;149:106854.
461. Müller L, Gairing SJ, Kloeckner R, Foerster F, Weinmann A, Mittler J, et al. Baseline Splenic Volume Outweighs Immuno-Modulated Size Changes with Regard to Survival Outcome in Patients with Hepatocellular Carcinoma under Immunotherapy. *Cancers (Basel)*. 2022;14:3574. doi:10.3390/cancers14153574.
462. Müller L, Kloeckner R, Mähringer-Kunz A, Stoehr F, Düber C, Arnhold G, et al. Fully automated AI-based splenic segmentation for predicting survival and estimating the risk of hepatic decompensation in TACE patients with HCC. *Eur Radiol*. 2022;32:6302–13. doi:10.1007/s00330-022-08737-z.
463. Stacke K, Bhattacharya I, Tse JR, Brooks JD, Sonn GA, Rusu M. Correlated Feature Aggregation by Region Helps Distinguish Aggressive from Indolent Clear Cell Renal Cell Carcinoma Subtypes on CT. 2022. <http://arxiv.org/abs/2209.14657>. Accessed 15 Oct 2022.
464. De Ruijter J, Muijsers JJM, Van De Vosse FN, Van Sambeek MRHM, Lopata RGP. A Generalized Approach for Automatic 3-D Geometry Assessment of Blood Vessels in Transverse Ultrasound Images Using Convolutional Neural Networks. *IEEE Trans Ultrason Ferroelectr Freq Control*. 2021;68:3326–35.
465. Björnsson PA, Baker A, Fleps I, Pauchard Y, Palsson H, Ferguson SJ, et al. Fast and robust femur segmentation from computed tomography images for patient-specific hip fracture risk screening. *Comput Methods Biomech Biomed Eng Imaging Vis*. 2022. doi:10.1080/21681163.2022.2068160.
466. Björnsson PA, Helgason B, Palsson H, Sigurdsson S, Gudnason V, Ellingsen LM. Automated femur segmentation from computed tomography images using a deep neural network. In: *Gimi BS, Krol A, editors. Medical Imaging 2021: Biomedical Applications in Molecular, Structural, and Functional Imaging*. SPIE; 2021. p. 47. doi:10.1117/12.2581100.

467. Bhardwaj M, Kushwah VS, Neogi SG. A Review on Medical Image Analysis using Deep Learning. *Eng Technol J Res Innov.* 2022;4.
468. Li R, Sharma V, Thangamani S, Yakimovich A. Open-Source Biomedical Image Analysis Models: A Meta-Analysis and Continuous Survey. *Front Bioinforma.* 2022;2:76. doi:10.3389/fbinf.2022.912809.
469. Chirodea MC, Novac OC, Novac CM, Bizon N, Oproescu M, Gordan CE. Comparison of Tensorflow and PyTorch in Convolutional Neural Network - based Applications. In: 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). IEEE; 2021. p. 1–6. doi:10.1109/ECAI52376.2021.9515098.
470. He H. The State of Machine Learning Frameworks in 2019. *The Gradient.* 2019. <https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/>. Accessed 13 Oct 2022.
471. Zhang L, Li J, Li P, Lu X, Gong M, Shen P, et al. MEDAS: an open-source platform as a service to help break the walls between medicine and informatics. *Neural Comput Appl.* 2022;34:6547–67. doi:10.1007/s00521-021-06750-9.
472. Google LLC. Google Scholar. <https://scholar.google.com/>. Accessed 14 Oct 2022.
473. National Library of Medicine. PubMed. <https://pubmed.ncbi.nlm.nih.gov/>. Accessed 14 Oct 2022.
474. Yeung M, Rundo L, Sala E, Schönlieb C-B, Yang G. Focal Attention Networks: Optimising Attention for Biomedical Image Segmentation. In: 2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI). IEEE; 2022. p. 1–5. doi:10.1109/ISBI52829.2022.9761414.
475. Yeung M, Rundo L, Nan Y, Sala E, Schönlieb C-B, Yang G. Calibrating the Dice loss to handle neural network overconfidence for biomedical image segmentation. 2021;1. <http://arxiv.org/abs/2111.00528>. Accessed 14 Oct 2022.
476. Yeung M, Yang G, Sala E, Schönlieb C-B, Rundo L. Incorporating Boundary Uncertainty into loss functions for biomedical image segmentation. 2021. <http://arxiv.org/abs/2111.00533>. Accessed 14 Oct 2022.
477. Jumutc V, Bližņuks D, Lihachev A. Multi-Path U-Net Architecture for Cell and Colony-Forming Unit Image Segmentation. *Sensors.* 2022;22:990. doi:10.3390/s22030990.
478. Affane A, Kucharski A, Chapuis P, Freydier S, Lebre M-A, Vacavant A, et al. Segmentation of Liver Anatomy by Combining 3D U-Net Approaches. *Appl Sci.* 2021;11:4895. doi:10.3390/app11114895.
479. Heo J. Automatic Segmentation in Abdominal CT Imaging for the KiTS21 Challenge. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Science and Business Media Deutschland GmbH; 2022. p. 98–102. doi:10.1007/978-3-030-98385-7_13.
480. Ogrea V, Brad R. Multi-Organ Segmentation Using a Low-Resource Architecture. *Information.* 2022;13:472. doi:10.3390/info13100472.
481. Lambert Z, Petitjean C, Dubray B, Ruan S. SegTHOR: Segmentation of Thoracic Organs at Risk in CT images. 2020 10th Int Conf Image Process Theory, Tools Appl IPTA 2020. 2019. <http://arxiv.org/abs/1912.05950>. Accessed 15 Oct 2022.
482. Yeung M. Attention U-Net ensemble for interpretable polyp and instrument segmentation. *Nord Mach Intell.* 2021;1:47–9. doi:10.5617/nmi.9157.
483. Kumar Singh V, Abdel-Nasser M, Pandey N, Puig D. LungNFseg: Segmenting COVID-19 Infected Regions in Lung CT Images Based on a Receptive-Field-Aware Deep Learning Framework. *Diagnostics.* 2021;11:158. doi:10.3390/diagnostics11020158.
484. Wu L, Hu S, Liu C. Exponential-Distance Weights for Reducing Grid-like Artifacts in Patch-Based Medical Image Registration. *Sensors.* 2021;21:7112. doi:10.3390/s21217112.
485. Xu Y, Hu S, Du Y. Research on Optimization Scheme for Blocking Artifacts after Patch-Based Medical Image Reconstruction. *Comput Math Methods Med.* 2022;2022:1–17. doi:10.1155/2022/2177159.
486. Pfliederer A, Müller D, Kramer F. Nucleus Segmentation and Analysis in Breast Cancer with the MISenn Framework. 2022. <http://arxiv.org/abs/2206.08182>. Accessed 17 Oct 2022.
487. Paschali M, Conjeti S, Navarro F, Navab N. Generalizability vs. Robustness: Investigating Medical Imaging Networks Using Adversarial Examples. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer Verlag; 2018. p. 493–501. doi:10.1007/978-3-030-00928-1_56.
488. Cui X, Chang S, Li C, Kong B, Tian L, Wang H, et al. DEAttack: A differential evolution based attack method for the robustness evaluation of medical image segmentation. *Neurocomputing.* 2021;465:38–52. doi:10.1016/j.neucom.2021.08.118.
489. Schorr C, Goodarzi P, Chen F, Dahmen T. Neuroscope: An Explainable AI Toolbox for Semantic Segmentation and Image Classification of Convolutional Neural Nets. *Appl Sci.* 2021;11:2199. doi:10.3390/app11052199.
490. Yan Z, Wicaksana J, Wang Z, Yang X, Cheng K-T. Variation-Aware Federated Learning With Multi-Source Decentralized Medical Image Data. *IEEE J Biomed Heal Informatics.* 2021;25:2615–28. doi:10.1109/JBHI.2020.3040015.
491. Yadav S, Rathod R, Pawar SR, Pawar VS, More S. Application of Deep Convolutional Neural Network in Medical Image Classification. In: 2021 International Conference on Emerging Smart Computing and Informatics (ESCI). IEEE; 2021. p. 120–9. doi:10.1109/ESCI50559.2021.9396854.
492. Azizi S, Mustafa B, Ryan F, Beaver Z, Freyberg J, Deaton J, et al. Big Self-Supervised Models Advance Medical Image Classification. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE; 2021. p. 3458–68. doi:10.1109/ICCV48922.2021.00346.
493. Wang L, Dou Q, Fletcher PT, Speidel S, Li S, editors. Medical Image Computing and Computer Assisted Intervention – MICCAI 2022. Cham: Springer Nature Switzerland; 2022. doi:10.1007/978-3-031-16437-8.
494. Gyawali PK, Ghimire S, Bajracharya P, Li Z, Wang L. Semi-supervised medical image classification with global latent mixing. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Science and Business Media Deutschland GmbH; 2020. p. 604–13. doi:10.1007/978-3-030-59710-8_59.
495. Grand Challenge. <https://grand-challenge.org/>. Accessed 27 Oct 2022.
496. Heller N, Isensee F, Trofimova D, Tejpaul R, Papanikolopoulos N, Weight C. Kidney and Kidney Tumor Segmentation. Cham: Springer International Publishing; 2022. doi:10.1007/978-3-030-98385-7.
497. Bakas S, Farahani K, Linguraru MG, Anazodo U, Carr C, Flanders A, et al. The Brain Tumor Segmentation Challenge (2022 Continuous Updates & Generalizability Assessment). 2022. doi:10.5281/ZENODO.6362180.
498. Hôpitaux de Paris Center Radboud University Medical. STOIC 2021 Challenge Workshop. 2022.
499. Hirano H, Minagi A, Takemoto K. Universal adversarial attacks on deep neural networks for medical image classification. *BMC Med Imaging.* 2021;21:1–13. doi:10.1186/s12880-020-00530-y.
500. Khan S, Naseer M, Hayat M, Zamir SW, Khan FS, Shah M.

- Transformers in Vision: A Survey. *ACM Comput Surv.* 2022;54:1–41. doi:10.1145/3505244.
501. Zhou Z, Lu C, Wang W, Dang W, Gong K. Semi-Supervised Medical Image Classification Based on Attention and Intrinsic Features of Samples. *Appl Sci.* 2022;12:6726. doi:10.3390/app12136726.
502. Amann J, Blasimme A, Vayena E, Frey D, Madai VI. Explainability for artificial intelligence in healthcare: a multidisciplinary perspective. *BMC Med Inform Decis Mak.* 2020;20:310. doi:10.1186/s12911-020-01332-6.
503. Li L, Qin L, Xu Z, Yin Y, Wang X, Kong B, et al. Artificial Intelligence Distinguishes COVID-19 from Community Acquired Pneumonia on Chest CT. *Radiology.* 2020;2020:16. doi:10.1148/radiol.20200905.
504. Bach S, Binder A, Montavon G, Klauschen F, Müller K-R, Samek W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS One.* 2015;10:e0130140. doi:10.1371/journal.pone.0130140.
505. Mertes S, Huber T, Weitz K, Heimerl A, André E. GANterfactual - Counterfactual Explanations for Medical Non-Experts using Generative Adversarial Learning. *Front Artif Intell.* 2020;5. doi:10.3389/frai.2022.825565.
506. Bisong E. Google AutoML: Cloud Vision. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform.* Apress; 2019. p. 581–98. doi:10.1007/978-1-4842-4470-8_42.
507. Molino P, Dudin Y, Miryala SS. Ludwig: a type-based declarative deep learning toolbox. 2019. <http://arxiv.org/abs/1909.07930>. Accessed 28 Oct 2022.
508. Jin H, Song Q, Hu X. Auto-Keras: An Efficient Neural Architecture Search System. 2019. doi:10.1145/3292500.
509. Google LLC. Vision AI | Cloud Vision API | Google Cloud. <https://cloud.google.com/vision>. Accessed 28 Oct 2022.
510. Trivizakis E, Manikis GC, Nikiforaki K, Drevelegas K, Constantinides M, Drevelegas A, et al. Extending 2-D Convolutional Neural Networks to 3-D for Advancing Deep Learning Cancer Classification with Application to MRI Liver Tumor Differentiation. *IEEE J Biomed Heal Informatics.* 2019;23:923–30.
511. NifTK/NiftyNet: [unmaintained] An open-source convolutional neural networks platform for research in medical image analysis and image-guided therapy. <https://github.com/NifTK/NiftyNet>. Accessed 31 Oct 2022.
512. Lahoud J, Cao J, Khan FS, Cholakkal H, Anwer RM, Khan S, et al. 3D Vision with Transformers: A Survey. 2022. <http://arxiv.org/abs/2208.04309>. Accessed 28 Oct 2022.
513. Chen J, He Y, Frey EC, Li Y, Du Y. ViT-V-Net: Vision Transformer for Unsupervised Volumetric Medical Image Registration. 2021. <http://arxiv.org/abs/2104.06468>. Accessed 28 Oct 2022.
514. Müller D, Hartmann D, Soto-Rey I, Kramer F. AUCMEDI: Von der Insellösung zur einheitlichen und automatischen Klassifizierung von medizinischen Bildern. In: *Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie. 67. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e. V. (GMDS), 13. Jahreskongress der Technologie- und Methodenplattform für. Kiel: Düsseldorf: German Medical Science GMS Publishing House; 2022. DocAbstr. 132; 2022. doi:10.3205/22gmds051.*
515. Radboud University Medical Center, Ziekenhuis Groep Twente, University Medical Center Groningen, Norwegian University of Science and Technology. PI-CAI Challenge: Artificial Intelligence & Radiologists at Prostate Cancer Detection in MRI. 2022. <https://pi-cai.grand-challenge.org/PI-CAI/>. Accessed 30 Oct 2022.
516. Saha A, Twilt JJ, Bosma JS, van Ginneken B, Yakar D, Elschot M, et al. The PI-CAI Challenge: Public Training and Development Dataset. 2022. doi:10.5281/ZENODO.6517398.
517. Stoddart C. Is there a reproducibility crisis in science? *Nature.* 2016;:3–5. doi:10.1038/d41586-019-00067-3.
518. Bausell RB. *The Problem with Science.* Oxford University Press; 2021. doi:10.1093/oso/9780197536537.001.0001.
519. Laraway S, Snyckerski S, Pradhan S, Huitema BE. An Overview of Scientific Reproducibility: Consideration of Relevant Issues for Behavior Science/Analysis. *Perspect Behav Sci.* 2019;42:33–57. doi:10.1007/s40614-019-00193-3.
520. Fanelli D. Is science really facing a reproducibility crisis, and do we need it to? *National Academy of Sciences;* 2018. doi:10.1073/pnas.1708272114.
521. Bilbao I, Bilbao J. Overfitting problem and the over-training in the era of data: Particularly for Artificial Neural Networks. In: *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS).* IEEE; 2017. p. 173–7. doi:10.1109/INTELCIS.2017.8260032.
522. Algan G, Ulusoy I. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Syst.* 2021;215:106771. doi:10.1016/j.knsys.2021.106771.
523. Mutasa S, Sun S, Ha R. Understanding artificial intelligence based radiology studies: What is overfitting? *Clin Imaging.* 2020;65:96–9. doi:10.1016/j.clinimag.2020.04.025.
524. Hartmann D, Schmid V, Meyer P, Soto-Rey I, Müller D, Kramer F. MISm: A Medical Image Segmentation Metric for Evaluation of weak labeled Data. 2022. <http://arxiv.org/abs/2210.13642>. Accessed 21 Nov 2022.
525. Zhang K, Liu X, Shen J, Li Z, Sang Y, Wu X, et al. Clinically Applicable AI System for Accurate Diagnosis, Quantitative Measurements, and Prognosis of COVID-19 Pneumonia Using Computed Tomography. *Cell.* 2020;181:1423-1433.e11.
526. Chen J, Lu Y, Yu Q, Luo X, Adeli E, Wang Y, et al. TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation. 2021. <http://arxiv.org/abs/2102.04306>. Accessed 12 Nov 2022.
527. Kaissis GA, Makowski MR, Rückert D, Braren RF. Secure, privacy-preserving and federated machine learning in medical imaging. *Nat Mach Intell.* 2020;2:305–11. doi:10.1038/s42256-020-0186-1.
528. Ziller A, Usynin D, Braren R, Makowski M, Rueckert D, Kaissis G. Medical imaging deep learning with differential privacy. *Sci Rep.* 2021;11:13524. doi:10.1038/s41598-021-93030-0.
529. Kaissis G, Ziller A, Passerat-Palmbach J, Ryffel T, Usynin D, Trask A, et al. End-to-end privacy preserving deep learning on multi-institutional medical imaging. *Nat Mach Intell.* 2021;3:473–84. doi:10.1038/s42256-021-00337-8.
530. Uploaded by Unknown. goodreads: Quotes. <https://www.goodreads.com/quotes/9365146-the-science-of-today-is-the-technology-of-tomorrow>. Accessed 14 Nov 2022.

