

UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA



DIPARTIMENTO  
DI INGEGNERIA  
DELL'INFORMAZIONE

MASTER THESIS IN COMPUTER ENGINEERING

# Supapixel Labeling in Medical Image Segmentation

MASTER CANDIDATE

**Somayeh Rezaei**

Student ID 2013082

SUPERVISOR

**Prof. Xiaoyi Jiang**

Westfälische Wilhelms-Universität Münster, WWU

CO-SUPERVISOR

**Prof. Albero Pretto**

Università degli Studi di Padova, UNIPD

ACADEMIC YEAR  
2022/2023



*I would like to express my greatest gratitude to my beloved partner, who supported me throughout this challenging journey. Truly, It would not have been possible to conquer the obstacles on this road without your sincere and wholehearted love and sacrifice. I am so fortunate to have such a wonderful man beside me, and I am tremendously grateful for this gift.*

*I would also like to extend my heartfelt thanks to my parents for their unwavering support and encouragement. Your guidance and belief in me have been instrumental in shaping my path. Your love has been my anchor, and I am deeply appreciative of all you've done.*

*Best Regards,*



## Abstract

Nowadays, most methods for image segmentation consider images in a pixel-wise manner, which is a huge job and also time-consuming. On the other hand, superpixel labeling can make the segmentation task easier in some aspects. First, superpixels carry more information than pixels because they usually follow the edges present in the image. Furthermore, superpixels have perceptual meaning, and finally, they can be very useful in computationally demanding problems, since by mapping pixels to superpixels we are reducing the complexity of the problem. In this thesis, we propose to do superpixel-wise labeling on two medical image datasets including ISIC Lesion Skin and Chest X-ray, then we feed them to the U-Net Convolutional Neural Network (CNN) DoubleU-Net and Dual-Aggregation Transformer (DuAT) network to segment our images in term of superpixels. Three different methods of labeling are used in this thesis: Superpixel labeling, Extended Superpixel Labeling (Distance-base Labeling), and Random Walk Superpixel labeling. The Superpixel labeled ground truths are used just for training. For the evaluation, we consider the original image and also the original binary ground truth. We considered four different superpixel algorithms, namely Simple Linear Iterative Clustering (SLIC), Felsenszwalb Huttenlocher (FH), QuickShift (QS), and Superpixels Extracted via Energy-Driven Sampling (SEEDS). We evaluate the segmentation result with metrics such as Dice Coefficient, Precision, Intersection Over Union (IOU), and Sensitivity. Our results show the accuracy of 0.89 and 0.95 percent in dice coefficient for skin lesion and chest X-ray datasets respectively.

**Key Words:** Superpixels, Medical Images, U-Net, DoubleU-Net, Image segmentation, CNN, DuAT, SEEDS.



# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Algorithms</b>	<b>xvii</b>
	<b>xvii</b>
<b>List of Acronyms</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 An overview of medical image segmentation and deep neural networks . . . . .	1
1.2 Related works . . . . .	4
<b>2 Background of Semantic Medical Image Segmentation</b>	<b>7</b>
2.1 Applications of Deep Learning in Medical Image Segmentation . .	8
2.2 Convolutional Neural Network . . . . .	10
2.2.1 Convolution . . . . .	11
2.2.2 Stride and Padding . . . . .	15
2.2.3 ReLU Layers . . . . .	16
2.2.4 Pooling Layers . . . . .	17
2.2.5 Fully Connected Layer . . . . .	18
2.2.6 Dropout Layers . . . . .	18
2.3 U-Net . . . . .	19
2.4 DoubleU-Net . . . . .	22
2.4.1 DoubleU-Net Architecture . . . . .	23
2.5 DuAT Network . . . . .	26
2.5.1 Transformer Encoder . . . . .	27

## CONTENTS

2.5.2	Selective Boundary Aggregation . . . . .	27
2.5.3	Global-to-Local Spatial Aggregation . . . . .	28
<b>3</b>	<b>Background of Superpixel Labeling</b>	<b>31</b>
3.1	Superpixel Methods . . . . .	31
3.1.1	Existing Superpixel Methods . . . . .	33
3.2	SLIC . . . . .	35
3.2.1	A. Algorithm . . . . .	36
3.2.2	B. Distance measure . . . . .	38
3.2.3	C. Post-processing . . . . .	41
3.3	FH . . . . .	41
3.4	Quickshift . . . . .	42
3.4.1	Procedure . . . . .	42
3.4.2	Important Aspects . . . . .	43
3.5	SEEDS . . . . .	43
3.5.1	Initialization . . . . .	44
3.5.2	Proposing Pixel-level and Block-level Movements . . . . .	45
3.5.3	Evaluating Pixel-level and Block-level Movements . . . . .	45
3.5.4	Termination . . . . .	47
<b>4</b>	<b>Our Method</b>	<b>49</b>
4.1	Preprocessing and Data Augmentation . . . . .	50
4.2	Superpixel Labeling . . . . .	52
<b>5</b>	<b>Results</b>	<b>57</b>
5.1	Dataset . . . . .	57
5.2	Evaluation Metrics . . . . .	60
5.2.1	Dice coefficient . . . . .	61
5.2.2	IOU . . . . .	63
5.2.3	Precision and Sensitivity . . . . .	65
5.3	Experiment Setup and Configuration . . . . .	65
5.4	Quantitative and Qualitative results . . . . .	68
<b>6</b>	<b>Conclusions and Future Works</b>	<b>75</b>
	<b>References</b>	<b>77</b>
	<b>Acknowledgments</b>	<b>87</b>



# List of Figures

2.1	Convolutional neural networks architecture . . . . .	11
2.2	Learned features from a Convolutional Neural Network . . . . .	12
2.3	Three dimensional Input representation of CNN . . . . .	12
2.4	Convolution as alternative for fully connected network . . . . .	13
2.5	Effects of different convolution matrix . . . . .	14
2.6	Multiple layers which each of them correspond to different filter but looking at the same region in the given image. . . . .	15
2.7	Common types of nonlinearity. . . . .	16
2.8	example of max-pooling with 2x2 filter and stride 2 . . . . .	17
2.9	Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped. . . . .	19
2.10	U-net architecture (example for 32x32 pixels in the lowest reso- lution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. . . . .	21
2.11	Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires im- age data within the blue area as input. Missing input data is extrapolated by mirroring . . . . .	22
2.12	Block diagram of the proposed DoubleU-Net architecture . . . . .	24

LIST OF FIGURES

2.13 The overall architecture of Dual-Aggregation Transformer Network (DuAT). The entire model is divided into three parts: (a) pyramid vision transformer (PVT) as backbone; (b) pyramid Global-to-Local Spatial Aggregation (GLSA) Module; (c) Selective Boundary Aggregation (SBA) module and it shown on the red box . . . . . 26

2.14 Overview of the Global-to-Local Spatial Aggregation Module GLSA, it is composed of global spatial attention (GSA) and local spatial attention (LSA) . . . . . 29

3.1 (a) Chest MRI Original Image, superpixel boundaries and mean RGB color of superpixel region (a) Skin Lesion Original Image, superpixel boundaries and mean RGB color of superpixel region . . . . . 32

3.2 Summary of Existing Superpixel Algorithms. . . . . 33

3.3 Reducing the superpixel search regions. The complexity of SLIC is linear in the number of pixels in the image  $O(N)$ , while the conventional k-means algorithm is  $O(kNI)$  where  $I$  is the number of iterations. This is achieved by limiting the search space of each cluster center in the assignment step. (a) In the conventional k-means algorithm, distances are computed from each cluster center to every pixel in the image.(b) SLIC only computes distances from each cluster center to pixels within a  $2S \times 2S$  region. Note that the expected superpixel size is only  $S \times S$ , indicated by the smaller square. This approach not only reduces distance computations but also makes SLIC's complexity independent of the number of superpixels. . . . . 37

3.4 SLIC supervoxels computed for a video sequence. (top) frames from a short video sequence of a flag waving. (bottom left) A volume containing the video. The last frame appears at the top of the volume. (bottom right) A supervoxel segmentation of the video. Supervoxels with orange cluster centers are removed for display purpose . . . . . 40

3.5	Initialization. Example of initialization with 12 superpixels and blocks of different sizes. The initialization occurs from left to right: first the smallest blocks are initialized, and then concatenated $2 \times 2$ to form larger blocks. The largest blocks are concatenated $2 \times 2$ to create the initial superpixels. This rectangular grid (in this case $4 \times 3$ ) is the starting point of the SEEDS algorithm. . .	44
3.6	Left: algorithm. Right: movements at pixel-level and at block-level	45
3.7	Block and pixel movements. This figure shows an example of the evolution of the superpixel boundaries while going through the iterations of the SEEDS algorithm (in the case of 12 superpixels). From left to right: The first image shows the initialization as a grid. The subsequent images show the block updates from large to small. The last image shows the pixel-level update of the superpixel boundaries . . . . .	46
4.1	Standard Pipeline . . . . .	49
4.2	our pipeline . . . . .	50
4.3	Sample Augmented Picture . . . . .	53
4.4	a) original image b) three label superpixel ground truth . . . . .	53
4.5	generating superpixel labels. a) original image. b) superpixel map. c) intersection of image boundaries with superpixels. d) hard ground truth with related boundaries in red e) superpixel outside the boundary. f) superpixel intersects with the boundary. g) superpixel intersects with the boundary. h) superpixel labeled ground truth. . . . .	55
4.6	a) number of pixels inside the boundary is more than the number of pixels outside the boundary. b) number of pixels outside the boundary is more than the number of pixels inside the boundary. c) number of pixels inside and outside of the boundary are equal.	56
5.1	Original Image and Related Ground Truth . . . . .	58
5.2	Shenzhen dataset example. Top: CXR images; bottom: their binary masks . . . . .	59
5.3	The graphical representation of TP, FP, TN and FN . . . . .	61
5.4	Illustration of dice coefficient . . . . .	62
5.5	illustration of IOU . . . . .	63
5.6	Segmentation result with unet network for lesion skin dataset . .	70

LIST OF FIGURES

5.7	Segmentation result with Double unet network for lesion skin dataset . . . . .	71
5.8	Segmentation result with DuAT network for lesion skin dataset .	72
5.9	Segmentation result with DuAT network for X-ray chest dataset .	73

# List of Tables

- 5.1 Segmenation Results for ISIC2018 lesion skin boundary dataset on U-Net network . . . . . 69
- 5.2 Segmenation Results for ISIC2018 lesion skin boundary dataset on Double U-net . . . . . 69
- 5.3 Segmenation Results for ISIC2018 lesion skin boundary dataset on U-net and SEEDS Alg . . . . . 69
- 5.4 Segmenation Results for ISIC2018 lesion skin boundary dataset on DuAT . . . . . 70
- 5.5 Segmenation Results for Lung X-Ray on DuAT . . . . . 70



# List of Algorithms

1	SLIC Superpixel Segmentation . . . . .	38
---	--	----









# List of Acronyms

**CT** Computer Tomography

**CNN** Convolutional Neural Network

**ReLU** Rectified Linear Units

**SLIC** Simple Linear Iterative Clustering

**IOU** Intersection Over Union

**MRI** Magnetic resonance imaging

**PET** Positron Emission Tomography

**FH** Felsenszwalb Huttenlocher

**ASPP** Atrous Spatial Pyramid Pooling

**ML** Machine learning

**ANN** Artificial Neural Network

**DL** Deep Learning

**RCNN** Region-Based Convolutional Neural Network

**DCNN** Deep Convolutional Neural Networks

**FCN** Fully-connected Convolutional Neural Network

**CAD** Computer-Aided Diagnosis

**NN** Neural Networks

**DNN** Deep Neural Networks

**ISLES** Ischemic Stroke Lesion Segmentation

**MTOP** Mild Traumatic Brain Injury Outcome Prediction

**BRATS** Brain Tumor Segmentation

**MCCNN** Multi-cascaded Convolutional Neural Network

**CRF** Fully Connected Conditional Random Fields

**mIOU** mean Intersection Over Union

**QS** QuickShift

**ISIC** International Skin Imaging Collaboration

**ISBI** International Symposium of Biomedical Imaging

**MICCAI** Computing and Computer Aided Intervention

**ESL** Extended Superpixel Labeling

**IOU** Intersection Over Union

**MIS** Medical Image Segmentation

**CDS** clinical decision support

**TP** True Positive

**FP** False Positive

**TN** True Negative

**FN** False Negative

**DuAT** Dual-Aggregation Transformer

**SEEDS** Superpixels Extracted via Energy-Driven Sampling

# 1

## Introduction

### 1.1 AN OVERVIEW OF MEDICAL IMAGE SEGMENTATION AND DEEP NEURAL NETWORKS

Medical image segmentation stands as a pivotal task within medical image analysis, encompassing various imaging modalities like microscopy, X-ray, ultrasound, Computer Tomography (CT), Magnetic resonance imaging (MRI), and Positron Emission Tomography (PET). Its significance extends across applications, including radiotherapy for specific cancers, establishing its vital role in healthcare [22].

Think of it like creating a map of a puzzle – by segmenting an image, doctors and researchers can identify and focus on specific areas of interest within the body. This helps them study diseases, plan treatments, and make accurate diagnoses. For instance, in radiotherapy, which is a common treatment for certain cancers, accurate segmentation of medical images is essential. It helps doctors precisely target cancerous tissues while minimizing damage to healthy cells.

In simpler terms, imagine you have a picture of a fruit salad, and you want to know the exact size and position of each type of fruit in the salad. Medical image segmentation is like doing that but with the different parts of the human body, like organs and tissues. This process assists doctors in creating effective treatment plans and monitoring the progress of diseases over time.

The techniques used for medical image segmentation involve advanced com-

## 1.1. AN OVERVIEW OF MEDICAL IMAGE SEGMENTATION AND DEEP NEURAL NETWORKS

puter algorithms that analyze the image data pixel by pixel. These algorithms use patterns and colors to distinguish one part of the image from another. By doing this, they create a "map" that helps doctors navigate and understand the complexities of the human body.

Overall, medical image segmentation is like drawing boundaries on a picture to help doctors see and understand the details of different body parts. This plays a significant role in medical treatments, enabling accurate targeting of diseases and providing better care for patients.

Medical image segmentation aids clinicians in concentrating on specific disease areas, enabling them to extract intricate details for enhanced precision in diagnosing conditions. The primary obstacles linked to medical image segmentation encompass limited access to extensively annotated datasets, a scarcity of well-labeled high-quality images for training purposes, suboptimal image clarity, the absence of a standardized segmentation procedure, and considerable image variability among different patients. It is vital to quantitatively assess segmentation accuracy and uncertainty to gauge performance across various applications. This underscores the necessity for an automated, adaptable, and effective method for semantically segmenting medical images [7].

Furthermore, we can divide the clinical application of medical image segmentation into three parts as follows; **Diagnosis:** Accurate segmentation helps physicians detect and diagnose diseases, anomalies, or conditions in the body. For example, segmenting brain MRI images can aid in identifying brain tumors or lesions. **Treatment Planning:** In radiation therapy or surgery, precise segmentation allows doctors to target specific areas while avoiding healthy tissues, and minimizing damage. **Monitoring:** Over time, segmented images can be compared to track disease progression, treatment effectiveness, and patient recovery.

In the old days, doctors used certain ways to divide pictures of the body. They found lines and edges in the pictures and followed changes in those lines. They also used math to do this. Another method was using a picture map. It copied the shape from the map onto the picture, but it had to fit perfectly. This worked if the picture lines up just right. Some ways used computer models to help divide the pictures. How well these models worked depended on how accurate and reliable they were. However, when the shapes were not normal, like irregular things in the body, these methods didn't work well[22].

Machine learning (ML) has become increasingly popular in the field of

medicine lately[98][72][90]. A subset of ML called Artificial Neural Network (ANN) has attracted attention. ANN is like a digital version of the human brain and it consists of layers of interconnected neurons. These neurons have weights and biases that can be adjusted as the network learns from data. The idea is to replicate how our brains process information to perform advanced tasks[2, 8, 20, 3, 19, 15].

Deep Learning (DL) emerges as a relatively recent term stemming from significant advancements in the architectures and algorithms of ANN since 2006, specifically focusing on ANNs with multiple hidden layers. The term "deep" underscores the presence of these stacked layers, signifying a higher level of network depth. However, an inherent challenge lies in establishing a precise threshold that definitively qualifies a network as "deep," resulting in a somewhat ambiguous distinction between traditional ANN structures and the more intricate realm of Deep Learning. This distinction is subjective and varies among experts and researchers, contributing to ongoing discussions on the criteria for identifying a deep network. Deep Learning, while emphasizing multi-layered architectures, encompasses a broader concept of machine learning, involving the training of models to autonomously derive insights and make intelligent decisions from data. Deep Learning leverages these layered structures to automatically learn intricate patterns and representations from extensive datasets, rendering it particularly adept in tasks such as image recognition, natural language processing, and autonomous driving. The transition from conventional ANNs to the concept of Deep Learning signifies a transformative shift in neural network design and application, prioritizing layered structures for amplified learning and decision-making capabilities, although the definitive differentiation between these terms remains an evolving aspect within the realm of machine learning research and practical implementation[1]. DL has showcased substantial promise in the field of computer vision[10]. Using a data-centric methodology, DL employs extensive image characteristics to support a range of visual assignments, including image classification, object detection, and segmentation [18].

Motivated by the achievements of DL in the field of computer vision, researchers have put forth various approaches to expand the application of DL techniques in the area of medical imaging. Thus far, extensive investigation into DL has been conducted across several medical image-related fields, including image segmentation[9, 23, 11, 12, 58], image synthesis, image enhancement and

## 1.2. RELATED WORKS

correction, and registration.

By the way, in chapter 2 we will explain more about Semantic medical image segmentation and provide more technical information about networks such as U-Net, DoubleU-Net, and DuAT network. Then, in Chapter 3, we will go into the details about Superpixel labeling and its different kinds of methods. chapter 4 is devoted to elaborating on our method in medical image segmentation with deep neural networks and Superpixel labeling. Chapter 5 is assigned to demonstrate the result of the segmentation with various evaluation metrics. Lastly, in the final section chapter 6 we discuss our conclusion and future works. Before going to the next chapter, It is well suited to talk about some related works that have been done in the area of medical image segmentation.

### **1.2** RELATED WORKS

A noticeable number of researches in medical image segmentation have been carried out based on a deep learning model especially CNN and pixel-wise ground truth labeling: one of the popular convolutional networks for segmenting medical images was introduced by Olaf Ronneberger, et al in 2015[69].

Skin cancer segmentation, lung segmentation, and retina segmentation(CHASE-DB1 dataset) are experimented with by Md Zahangir Alom, et al[63] with the help of another improved U-Net structure called U-Net (R2U-Net). the mentioned model is the combination of U-Net, Residual Network, as well as Region-Based Convolutional Neural Network (RCNN)

Debesh Jha, et al, introduce a new design known as DoubleU-Net, a distinctive configuration that involves the arrangement of two U-Net architectures stacked on the top of each other[7]. The experiment has been done on two medical datasets; 2015 MICCAI, and CVC-ClinicDB[52].

MultiResUNet, a modification on U-Net, is proposed by Nabil Ibtehaz, et al[67]. In the MultiResUNet model, they replace the sequence of two convolutional layers with the MultiRes block, and the architecture is tested on five different medical datasets.

Alexandr A. Kalinin, et al[29], begin with the U-Net model as a starting point, and then they show how they can have results even better by using different advanced designs that use ImageNet-pre-trained encoders.

Moreover, another paper introduces a deep neural network (LCP-Net), sug-



gested by Dunlu Peng, et al[40] in medical image segmentation. this novel structure can perceive multi-scale context information of images.

Furthermore, there have been several studies addressing medical image segmenting with the help of convolutional neural networks and superpixel labeling. Superpixels hold more information than regular pixels because they fit better with the natural edges in images compared to square sections. Secondly, superpixels are meaningful because pixels in the same superpixel look similar in terms of how they appear.

Refer to the experiment done by Qingwu Shi, et all[75]. A multichannel convolutional neural network (CNN)-based fuzzy active contour model for medical image segmentation is introduced. Also, Medical images are labeled using the superpixel SLIC method.

Another approach that uses pseudo labels and superpixels is invented by Bethany H. Thompson, et al [31]. Indeed they merge superpixels with semi-supervised learning. This helps enhance the pseudo-labels while training. Worth mentioning that, they use U-Net as the machine learning model. The evaluation is also carried out on the BraTS 2020 dataset[30].

U-Net++ which is a developed version of the U-Net method, was published in 2018. The structure can be described as a deeply-supervised encode-decoder network that is linked together with a series of nested, dense skip pathways. furthermore, the evaluation is done on several medical images including nodule segmentation in the low-dose CT scans of the chest, nuclei segmentation in the microscopy images, liver segmentation in abdominal CT scans, and polyp segmentation in colonoscopy videos [24].



# 2

## Background of Semantic Medical Image Segmentation

Semantic medical image segmentation involves training computers to recognize and label different structures or areas within medical images, helping doctors analyze and understand these images more effectively. One focal domain of investigation within this domain pertains to segmenting medical images, which holds significant importance in various medical imaging investigations. Similar to other branches of research within computer vision, deep learning networks attain remarkable outcomes and typically surpass traditional non-deep methods in advanced medical imaging techniques. Deep neural networks are predominantly employed for classification tasks, where the network's result comprises a single label or probabilities values associated to labels for a provided input image[78].

These networks perform effectively due to certain structural features such as: activation function, different efficient optimization algorithms, and dropout as a regularizer for the network[63]. Starting from 2012, numerous models of Deep Convolutional Neural Networks (DCNN) have been introduced[63], including AlexNet[56], VGG[83] , GoogleNet[84] , Residual Net[49] , DenseNet[51] , and CapsuleNet[80] .An approach rooted in Deep Learning (specifically, CNNs) delivers cutting-edge results for tasks like classification and segmentation due to several factors. First, activation functions tackle training issues in DL methods. Second, dropout helps in regulating the networks. Third, numerous effective optimization techniques are available to train CNN models[56]. Nonetheless, in

## 2.1. APPLICATIONS OF DEEP LEARNING IN MEDICAL IMAGE SEGMENTAION

the majority of instances, models are investigated and assessed using extensive datasets like ImageNet [56] for classification tasks, where the outcomes involve single labels or probability values. Conversely, small architecturally variant models are applied for tasks related to semantic image segmentation. To illustrate, Fully-connected Convolutional Neural Network (FCN) also demonstrates remarkable outcomes for image segmentation tasks in computer vision [61]. An alternative version of FCN was introduced under the name SegNet[35].

Due to the remarkable achievements of DCNNs in the realm of computer vision, various adaptations of this approach find applications across diverse medical imaging modalities, encompassing tasks such as segmentation, classification, detection, registration, and the processing of medical information. Medical imaging emerges from various techniques such as CT, ultrasound, X-ray, and MRI. The objective of Computer-Aided Diagnosis (CAD) is to expedite and enhance diagnoses for a huge number of individuals simultaneously, aiming for improved treatment outcomes. Because manual segmentation methods are time-consuming and labor-intensive, there exists a substantial need for computer algorithms capable of swiftly and precisely performing segmentation tasks without human involvement. Nonetheless, medical image segmentation has its challenges, including insufficient data and imbalanced class distribution[63]. Often, the abundance of labels required for training, sometimes reaching thousands, is unavailable due to various factors [35].

Getting an expert in the field to label the dataset is pricey and takes a lot of time and effort. Sometimes, different methods are used to increase the available labeled samples, like changing the data in ways such as whitening, rotating, moving, and resizing (data transformation and augmentation) [16, 4, 13]. In the following we will talk more about CNN, FCN, U-Net and Double U-Net. In this chapter, we explain in details Neural Networks (NN), Deep Neural Networks (DNN), CNN, FCN, U-Net, Double-Unet and DuAT.

## **2.1** APPLICATIONS OF DEEP LEARNING IN MEDICAL IMAGE SEGMENTAION

Even though research on the segmentation of MRI and CT images encompasses various organs like the knee cartilage, prostate, and liver, a predominant

portion of the research has been focused on brain segmentation, specifically concerning tumors. This matter holds immense significance in surgical planning, aiming to define precise tumor boundaries for minimal resection during surgery. Excessive removal of crucial brain regions during surgery can result in neurological shortfalls including cognitive damage, motionlessness, and limb difficulty. Traditionally, the segmentation of medical anatomy was a manual task, involving clinicians drawing lines on every slice of the complete CT or MRI volume. Consequently, automating this laborious process becomes highly desirable[57]. Wadhwa et al. [88] delivered a concise overview of brain tumor segmentation in MRI images. Akkus et al.[27] authored an insightful survey on brain MRI segmentation, discussing various metrics and CNN architectures employed in the field. Additionally, they elucidate numerous competitions in detail along with their associated datasets, which encompass Ischemic Stroke Lesion Segmentation (ISLES), Mild Traumatic Brain Injury Outcome Prediction (MTOP), Brain Tumor Segmentation (BRATS).

Chen and colleagues [34] presented a CNN strategy designed for accurate BRATS. Their approach incorporates various techniques to enhance feature learning, including the utilization of the DeepMedic model, an innovative dual-force training scheme, a loss function based on label distribution, and a post-processing step using a Multi-Layer Perceptron. They assessed their methodology using the most up-to-date brain tumor segmentation datasets, namely BRATS 2017 and BRATS 2015 datasets. Hu and co-authors [50] introduced a brain tumor segmentation approach employing a Multi-cascaded Convolutional Neural Network (MCCNN) combined with Fully Connected Conditional Random Fields (CRF)s. The achieved outcomes exhibited remarkable performance in comparison to cutting-edge techniques in the field.

Moeskops and co-authors [65] utilized a set of three concurrently operating CNNs, each designed with a distinct 2D input patch size, to segment and categorize MRI brain images. These images, encompassing 35 adults and 22 pre-term infants, were classified into various tissue categories, including cerebrospinal fluid, grey matter, and white matter. Every patch was designed to capture different image aspects, with the advantage of incorporating three distinct input patch sizes; the larger sizes captured spatial features, while the smaller patch sizes focused on local textures. Overall, the algorithm yielded Dice coefficients ranging from 0.82 to 0.87, achieving a satisfactory level of accuracy.

While the majority of segmentation research employs 2D image slices, Milletrate

## 2.2. CONVOLUTIONAL NEURAL NETWORK

and colleagues [64] adopted a 3D CNN for segmenting MRI prostate images. They utilized the PROMISE2012 challenge dataset, using fifty MRI scans for training and thirty for testing. Drawing inspiration from Ronnerberger et al.'s U-Net architecture [79], they developed their own V-net. This model achieved a Dice coefficient score of 0.869, matching the scores of winning teams in the competition. To mitigate overfitting and create a deeper model with eleven convolutional layers, Pereira and team [73] employed deliberately small 3x3 filters. Their model employed MRI scans of 274 gliomas (a type of brain tumor) for training. They secured first place in the 2013 BRATS challenge and second place in the BRATS challenge of 2015.

Havaei et al. [48] also delved into glioma analysis using the 2013 BRATS dataset. They explored various 2D CNN architectures. In comparison to the winner of BRATS 2013, their algorithm performed more efficiently, executing in just 3 minutes instead of 100. Their model was founded on the concept of a cascaded architecture, earning it the name InputCascadeCNN.

Introducing techniques like CRFs, atrous spatial pyramid pooling, and up-sampled filters, Chen et al. [33] aimed to enhance localization accuracy and expand the field of view for filters at multiple scales. Their model, named DeepLab, achieved a mean Intersection Over Union (mIOU) of 79.7%. Their model also demonstrated excellent performance in the PASCAL VOC-2012 image segmentation task.

Lately, the utilization of various deep learning methods has become instrumental in automatically segmenting lung infections caused by COVID-19 in CT images. This approach aids in identifying the progression of COVID-19 infection [57, 89].

## **2.2** CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks, often abbreviated as CNNs, are a class of deep learning models that have revolutionized various fields, especially computer vision. These networks are inspired by the way the human visual system processes information, allowing them to excel in tasks involving visual data, such as image classification, object detection, and image segmentation.

Unlike traditional neural networks, CNNs are designed to automatically

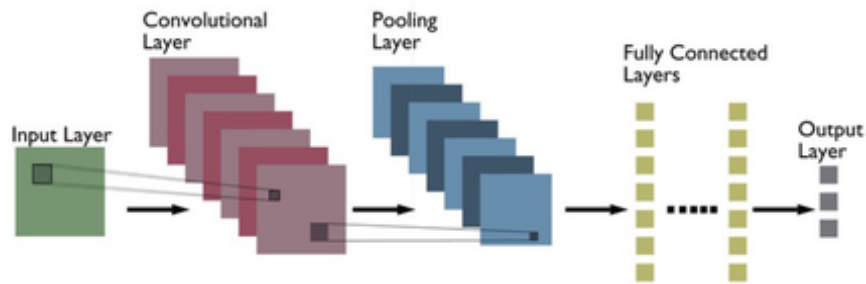


Figure 2.1: Convolutional neural networks architecture

and adaptively learn patterns and features directly from the input data. They consist of specialized layers that perform convolutions, which are mathematical operations that extract features from images. These layers are followed by pooling layers that help reduce the dimensionality of the data while retaining important information. The final layers are usually fully connected, enabling the network to make predictions based on the learned features. The picture 2.1 represents a typical CNN architecture[100].

CNNs have significantly advanced the field of computer vision by achieving remarkable accuracy in various complex tasks. Their ability to capture hierarchical features and patterns from raw data has enabled breakthroughs in image understanding, leading to applications in medical imaging, autonomous vehicles, image and video analysis, and more.

An additional significant aspect of CNNs is their ability to acquire abstract features as the input propagates through the deeper layers. For instance, when dealing with image classification, the initial layers might identify edges, followed by the recognition of more basic shapes in subsequent layers. Subsequently, as depicted in Figure 2.2 [17], the deeper layers might even identify higher-level features like faces. To develop a solid understanding of CNNs, we initiate our exploration by going deep into their fundamental components.

### 2.2.1 CONVOLUTION

Let's suppose that the input to our neural network has the illustrated configuration shown in Figure 2.3 [17]. This input could be an image, such as a color image from the CIFAR-10 dataset with dimensions of 32x32 pixels and a depth of 3 corresponding to the RGB channels. Alternatively, it could be a video, like a grayscale video with dimensions determined by resolution and frames as the

## 2.2. CONVOLUTIONAL NEURAL NETWORK

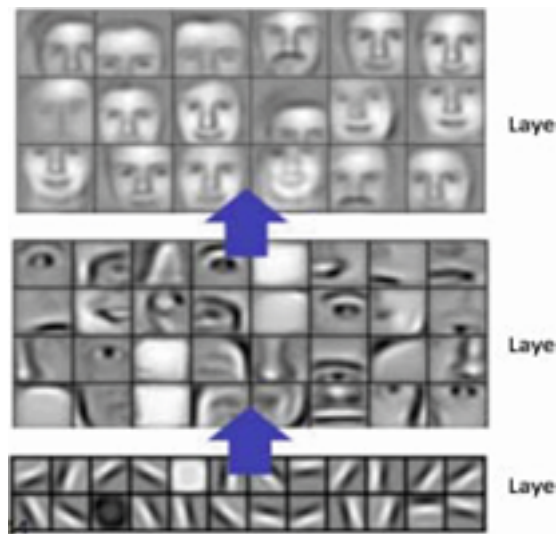


Figure 2.2: Learned features from a Convolutional Neural Network

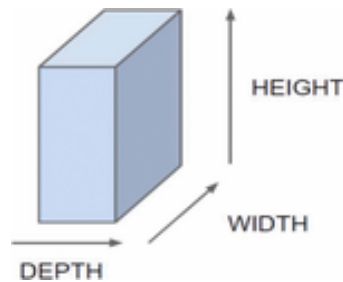


Figure 2.3: Three dimensional Input representation of CNN

depth. Additionally, it might even be an experimental video characterized by sensor values with dimensions of  $(L \times L)$  and varying depths corresponding to different time frames, as seen in references [5].

Why utilize convolution? Consider a scenario where the network is fed raw pixels as input. In this case, if we aim to link the input layer to only one neuron (for instance, within the hidden layer of a Multi-Layer Perceptron), a total of  $32 \times 32 \times 3$  weight connections would be required for the CIFAR-10 dataset, as an example.

If we introduce an additional neuron to the hidden layer, this would entail the requirement for another set of  $32 \times 32 \times 3$  weight connections, summing up to a total of  $32 \times 32 \times 3 \times 2$  parameters. To illustrate, more than 6000 weight parameters are employed to establish connections between the input and only two nodes. This might give the impression that two neurons might not be sufficient for performing any meaningful image classification tasks. To enhance efficiency, we can establish connections from the input image to the neurons in the subsequent



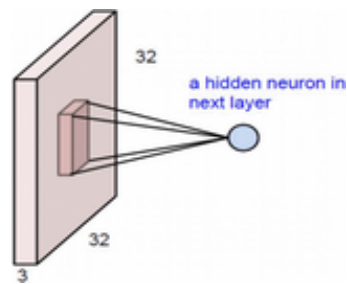


Figure 2.4: Convolution as alternative for fully connected network

layer while maintaining the same height and width values. This configuration could be suitable for processing tasks like edge detection within images. However, this arrangement would necessitate  $32 \times 32 \times 3$  by  $32 \times 32$  weight connections, equivalent to (3,145,728), as documented in references[21, 108].

Hence, in the pursuit of a more efficient approach, the concept emerged that instead of employing a complete connection, it would be advantageous to search for local regions within the image, rather than encompassing the entire image. As depicted in Figure 2.4 [17], a localized connection pattern is showcased for the subsequent layer. To elaborate, the hidden neurons in the subsequent layer exclusively receive inputs from the corresponding segments of the preceding layer. For instance, they could solely be linked to a  $5 \times 5$  array of neurons. Consequently, if the intention is to accommodate  $32 \times 32$  neurons in the subsequent layer, this would entail  $5 \times 5 \times 3$  by  $32 \times 32$  connections, amounting to a total of 76,800 connections (in comparison to the 3,145,728 connections required for full connectivity), as indicated in references[108, 104, 6, 14].

Despite the significant reduction in the number of connections, there remains a substantial number of parameters that need to be resolved. To further simplify, another assumption is to maintain constant weights for local connections across all neurons in the subsequent layer. This approach links neighboring neurons in the next layer with identical weights to the local area of the preceding layer. As a result, this approach further decreases the number of excess parameters and minimizes the weight count to just 75 ( $5 \times 5 \times 3$ ), enabling the connection of  $32 \times 32 \times 3$  neurons to the  $32 \times 32$  grid in the following layer. Numerous advantages arise from these straightforward assumptions. Initially, the quantity of connections diminishes dramatically from roughly 3 million to a mere 75 connections in the provided illustration. Additionally, an intriguing concept emerges: by maintaining consistent weights for local connections, it's akin to traversing a  $5 \times 5 \times 3$  window across the input neurons and associating the resulting output

## 2.2. CONVOLUTIONAL NEURAL NETWORK



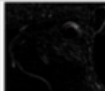




Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 2.5: Effects of different convolution matrix

with the respective location. This approach facilitates the detection and identification of features irrespective of their placements within the image. This is precisely why they are termed convolutions.

To demonstrate the remarkable impact of the convolution matrix, Figure 2.5 [17] illustrates the outcome when we manually select the connection weights within a 3x3 window.

As depicted in Figure 2.5, the matrix can be configured to identify edges within the image. These matrices are also referred to as filters due to their resemblance to conventional image processing filters. Nonetheless, within the CNN, these filters are initialized and subsequently refined through the training process to adopt shapes that are better suited for the specific task at hand.

To enhance the efficacy of this approach, it is viable to introduce additional layers subsequent to the input layer. Each layer can correspond to distinct filters, enabling the extraction of diverse features from the provided image. Figure 2.6 illustrates the connections of these layers. Each layer employs its own filter, thereby extracting unique features from the input. The neurons depicted in Figure 2.6 [17] employ distinct filters while focusing on the same segment of the input image[17].

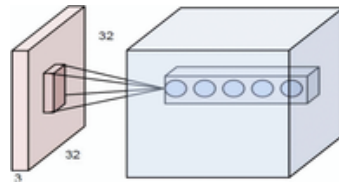


Figure 2.6: Multiple layers which each of them correspond to different filter but looking at the same region in the given image.

### 2.2.2 STRIDE AND PADDING

In a CNN, the stride is a hyperparameter that determines the step size at which the convolutional filter moves across the input data (e.g., an image) during the convolution operation. The stride value directly affects the spatial dimensions of the output feature map after the convolution.[91]

Here's how stride works:

- **Larger Stride:** When you use a larger stride value (greater than 1), the filter moves multiple steps at a time. This results in the output feature map being "downsampled" compared to the input. Fewer positions are considered, reducing the dimensions of the output. For example, if the input image is  $5 \times 5$  and the filter is  $3 \times 3$  with a stride of 2, the output will be  $2 \times 2$ .
- **Smaller Stride:** A smaller stride value (equal to 1) means the filter moves one step at a time. This produces a more detailed output feature map compared to using a larger stride. If you apply a  $3 \times 3$  filter with a stride of 1 to a  $5 \times 5$  image, the output will be  $3 \times 3$ .

The choice of stride impacts the trade-off between computational efficiency and the amount of spatial information retained in the output. Larger strides can reduce computation but might lead to loss of fine details.

Padding involves adding extra rows and columns of zeros around the input data before applying the convolution operation. This ensures that the filter can cover the edges of the input and helps maintain spatial dimensions in the output.

Here's how padding works[91]:

- **No Padding:** Without padding, the filter might not fit entirely on the edges of the input. This could result in a smaller output feature map compared to the input size.
- **Valid Padding (No Padding):** The filter only convolves where it fully overlaps with the input. This often leads to a smaller output.

## 2.2. CONVOLUTIONAL NEURAL NETWORK

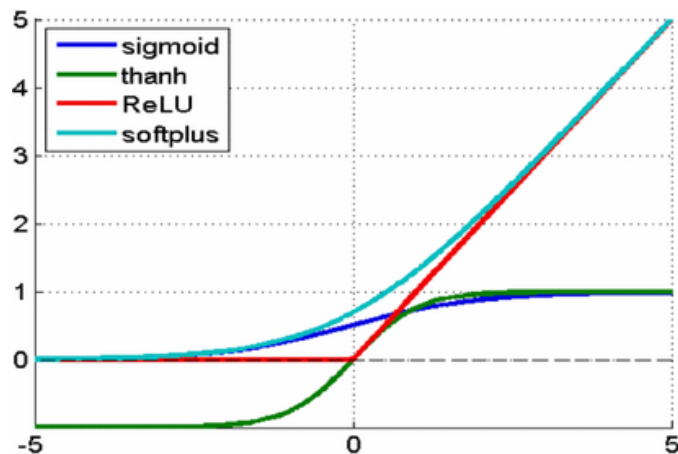


Figure 2.7: Common types of nonlinearity.

- Same Padding: Same padding adds enough zeros around the input so that the output has the same dimensions as the input (when the stride is 1). For example, if you apply a 3x3 filter with same padding to a 5x5 image, the output will also be 5x5.

### 2.2.3 RELU LAYERS

Following each convolutional layer, it is customary to immediately apply a nonlinear layer, also known as an activation layer. The purpose of this particular layer is to introduce nonlinearity into a system that has primarily conducted linear operations through the convolutional layers, involving element-wise multiplications and summations. Historically, nonlinear functions like tanh and sigmoid were utilized. Fig 2.7 [17], shows the common types of nonlinearity.

However, recently, Rectified Linear Units (ReLU) has been used more often for the following reasons. However, researchers discovered that ReLU layers are considerably more effective. The use of layers enables faster network training due to computational efficiency, with minimal impact on accuracy. Moreover, ReLU layers help mitigate the vanishing gradient problem, a challenge where lower network layers train slowly due to an exponential gradient decrease across layers. The ReLU layer employs the function  $f(x) = \max(0, x)$  to process all values in the input volume, effectively transforming negative activations to 0. This layer enhances the model's nonlinearity and the overall network without influencing the receptive fields of the convolutional layer[86].

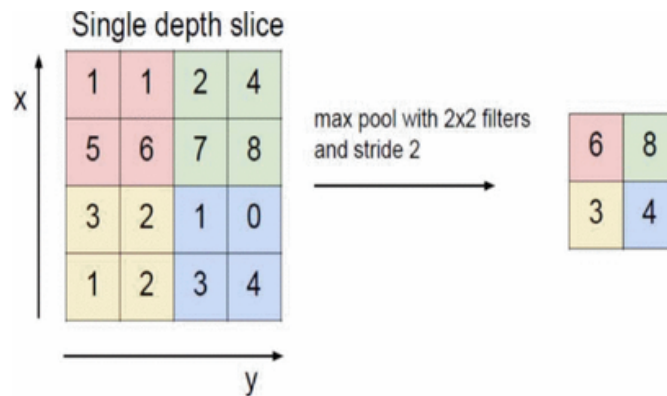


Figure 2.8: example of max-pooling with 2x2 filter and stride 2

However, recently, ReLU has been used more often for the following reasons. However, researchers discovered that ReLU layers are considerably more effective. The use of ReLU layers enables faster network training due to computational efficiency, with minimal impact on accuracy. Moreover, ReLU layers help mitigate the vanishing gradient problem, a challenge where lower network layers train slowly due to an exponential gradient decrease across layers. The ReLU layer employs the function  $f(x) = \max(0, x)$  to process all values in the input volume, effectively transforming negative activations to 0. This layer enhances the model's nonlinearity and the overall network without influencing the receptive fields of the convolutional layer[86].

#### 2.2.4 POOLING LAYERS

Following ReLU layers, developers might opt to utilize a pooling layer, which is also known as a downsampling layer. Within this group, multiple layer choices exist, although maxpooling stands out as the most commonly used. Essentially, maxpooling involves employing a filter, typically sized at 2x2, with a stride of equivalent length. This filter is applied to the input volume, producing the highest number within each subregion that the filter convolves across.

in Fig. 2.8 [17] Max-pooling is demonstrated. The max-pooling with 2x2 filter and stride 2 lead to down-sampling of each 2x2 blocks is mapped to 1 block (pixel).

## 2.2. CONVOLUTIONAL NEURAL NETWORK

### 2.2.5 FULLY CONNECTED LAYER

Fully-connected layers represent a fundamental component of CNNs. As their name suggests, each neuron within a fully-connected layer establishes connections with every neuron in the preceding layer. Typically situated towards the CNN's conclusion, these layers leverage the knowledge acquired from earlier convolutional and pooling layers to facilitate tasks like classification. For instance, in a scenario involving image classification of animals, the ultimate fully-connected layer can effectively utilize information from prior layers to classify an image based on the presence of animals like dogs, cats, birds, and more[38].

### 2.2.6 DROPOUT LAYERS

Dropout layers serve a highly specific purpose within neural networks. Furthermore, sometime overfitting might happen once training a network, which arises when a network becomes so finely adjusted to its training data that its performance diminishes on test data. The concept behind dropout is elegantly straightforward: this layer selectively deactivates a random subset of activations within that layer by assigning them a value of zero. It's that simple. But why employ such a basic, seemingly superfluous, and counterintuitive approach? The rationale behind this lies in compelling the network to embrace redundancy. In other words, the network should ideally produce accurate classifications or outputs for specific instances even in the absence of some activations. This practice ensures that the network doesn't overly adapt to the training data, effectively addressing the issue of overfitting. It's crucial to note that this layer is solely utilized during training and is not active during the testing phase.

The expression "dropout" pertains to dropping out units (both hidden and visible) within a neural network. When a unit is dropped out, it signifies a momentary removal from the network, along with all its associated incoming and outgoing connections, as depicted in Figure 2.9 [68].

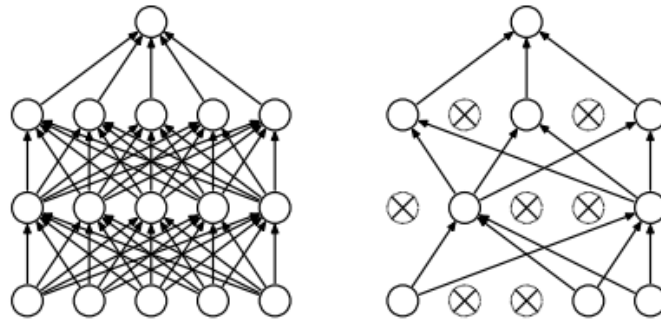


Figure 2.9: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

### 2.3 U-NET

Over the last two years, deep convolutional networks have shown they're really good at recognizing things in pictures, doing better than other methods. Some recent studies, like ones cited in references[56, 46], highlight this progress. You see, we've known about convolutional networks for a while, but they weren't as great because of limited training data and network complexity.

However, things changed a lot with the work done by Krizhevsky and colleagues[56]. They did something smart: they trained a big network with 8 layers and lots of settings. This network was taught using a famous dataset called ImageNet, which has a huge collection of 1 million training images. This clever combo of a powerful network and a big dataset made a big difference.

Since then, there's been a bunch of progress. Even fancier and bigger networks have been created and trained[83]. These new networks build on what Krizhevsky and team did and show that people are still trying to make deep learning for recognizing images even better. This whole journey from where we started with convolutional networks has totally changed how we understand and use them for looking at pictures. It's been quite a journey of learning and improving in the world of image recognition!

Convolutional networks are typically used for tasks like putting images into categories, where the result is just one label. However, in a lot of visual tasks, especially when working with medical images, we want more than just categories – we want to know exactly where each category is in the picture. This is called localization, and it means giving a category to every little part of the image, like each pixel.

### 2.3. U-NET

Here's the tricky part: getting thousands of images to teach the network from is usually really tough in medical tasks. So, Ciresan and their team came up with a smart solution [35]. They trained a network like you'd slide a window across the picture. For each tiny piece (or "patch") of the image, the network learned which category it belonged to. This way, even with a limited number of images, they taught the network to understand every corner of the picture.

The cool thing is that this network not only figures out where things are but also does a great job at it. Plus, they used way more patches for training than they would have used whole images, making their training data even more powerful. Because of this, their network aced the EM segmentation challenge at ISBI 2012, beating others by a big margin.

The strategy adopted by Ciresan and their research team [35] does bring to light certain limitations that warrant consideration. The foremost drawback is centered around the speed of their approach. The necessity to execute the network independently for each individual patch contributes to a noticeable decrease in processing speed. This becomes particularly noticeable when dealing with images composed of numerous patches. Moreover, the presence of overlapping patches results in a repetitive computation process, further exacerbating the computational time. The cumulative effect of these factors underscores the need for efficiency enhancements in this approach.

Beyond this computational concern, another trade-off becomes evident in the approach. This compromise revolves around the dual challenge of achieving localization accuracy and the use of context. This duality is especially pronounced when varying patch sizes are employed. When larger patches are chosen, the requirement to introduce additional max-pooling layers arises. While these layers contribute to information abstraction and hierarchy development, they can inadvertently lead to a reduction in localization accuracy. On the contrary, selecting smaller patches mitigates the challenge of localization accuracy, but does so at the expense of contextual understanding. Striking a harmonious balance between these contrasting aspects thus emerges as a significant aspect of optimizing the model's performance.

Addressing these limitations, more recent methodologies, as presented in references[47, 81], offer innovative solutions to ameliorate the concerns raised by Ciresan's approach. These novel methods introduce an evolution in the design of classifier outputs. Rather than relying solely on the insights gleaned from a single layer, these approaches leverage the collective wisdom of multi-



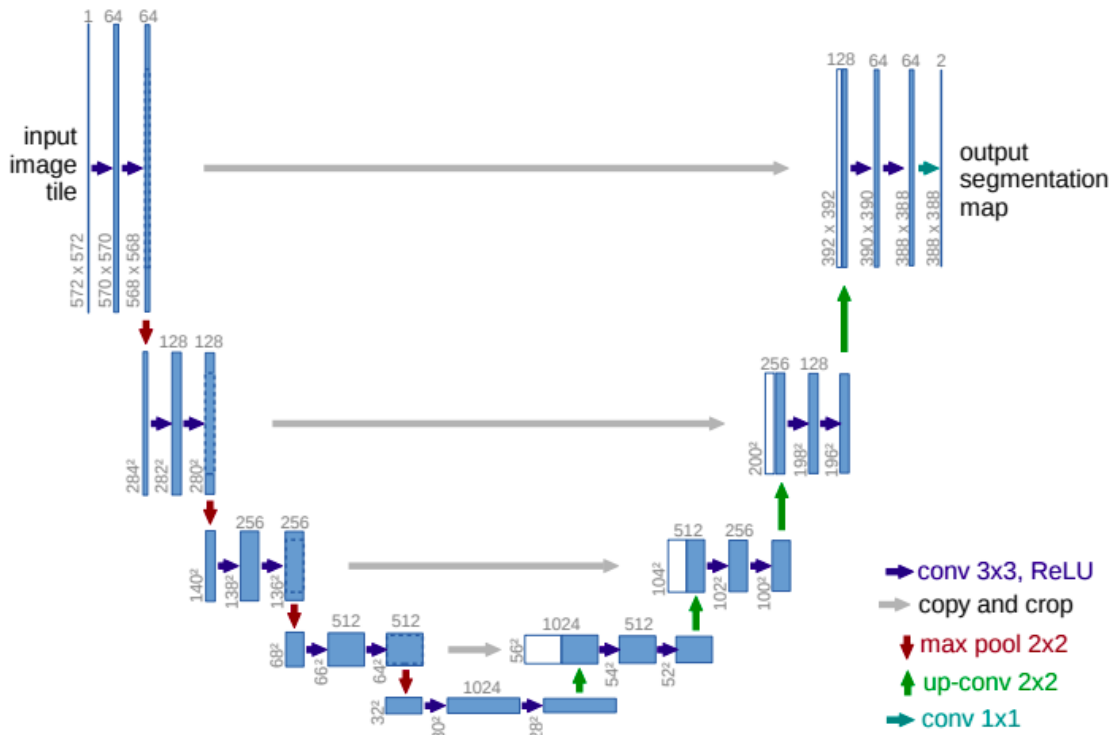


Figure 2.10: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

ple layers within the network. This refined approach enables a more holistic understanding of the input data. By synthesizing information from different levels of abstraction, these methods manage to concurrently achieve localization accuracy and a comprehensive perception of the surrounding features. This advancement underscores the dynamic nature of neural network design, where the fusion of information from various layers generates a more refined and nuanced interpretation of complex visual data.

The arrangement of the network is depicted in Figure 2.10 [70]. It comprises a contracting path on the left and an expansive path on the right. The contracting path adheres to the standard convolutional network architecture. It involves repeatedly applying two 3x3 convolutions (unpadded convolutions), followed by a rectified linear unit (ReLU), and a 2x2 max pooling operation with a step size of 2 for downsampling. With each downsampling step, the number of feature channels is doubled. Each step in the expansive path encompasses

## 2.4. DOUBLEU-NET

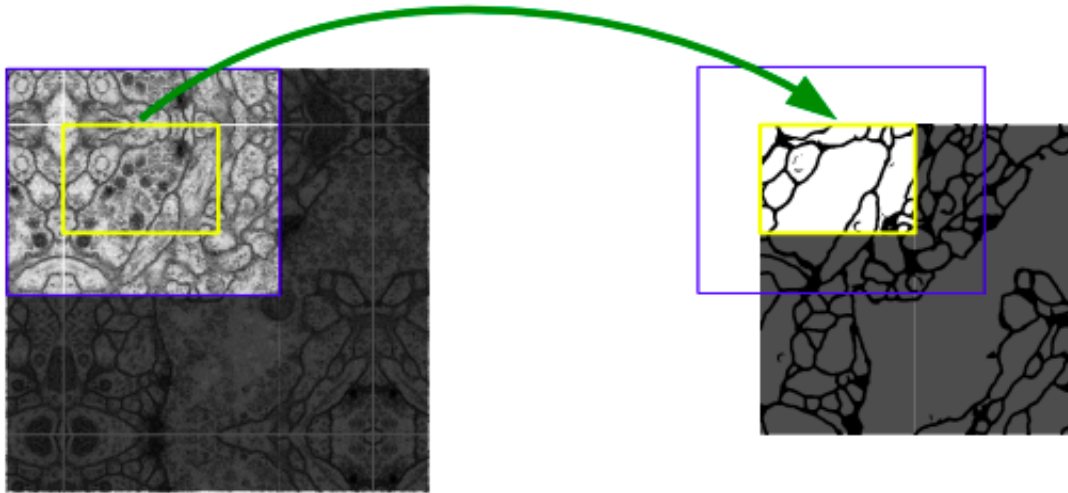


Figure 2.11: Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

an enlargement of the feature map followed by a  $2 \times 2$  convolution (referred to as "up-convolution") that reduces the number of feature channels by half. This is then combined with the appropriately cropped feature map from the contracting path, followed by two  $3 \times 3$  convolutions, each succeeded by a ReLU activation. Cropping is required due to the loss of pixels along the borders after each convolution. The final layer employs a  $1 \times 1$  convolution to map each 64-component feature vector to the desired class count. Overall, the network comprises a total of 23 convolutional layers[70]. For a smooth arrangement of the output segmentation map (as shown in Figure 2.11 [70], it's crucial to pick the input tile dimensions in a way that ensures all the  $2 \times 2$  max-pooling actions are performed on a layer with an even x- and y-size.

## 2.4 DOUBLEU-NET

Utilizing an encoder-decoder approach such as U-Net and its variations has gained popularity in addressing medical image segmentation challenges. To enhance U-Net's effectiveness across diverse segmentation tasks, a novel architecture termed DoubleU-Net was introduced by Debesh Jha et al[7] in 2020. This innovative design consists of two U-Net structures stacked sequentially. The initial U-Net employs a pre-trained VGG-19 as its encoder, leveraging previously

acquired features from ImageNet for adaptable transfer to new tasks. To enhance semantic comprehension efficiently, an additional U-Net module is integrated at the base. Furthermore, they incorporate Atrous Spatial Pyramid Pooling (ASPP) to capture contextual context within the network.

### 2.4.1 DOUBLEU-NET ARCHITECTURE

The diagram presented in Figure 2.12 [7] provides an outline of the proposed architectural framework. The visual representation illustrates that the DoubleU-Net commences with an encoder sub-network employing the VGG-19 model, which is subsequently succeeded by a decoder sub-network. What sets the DoubleU-Net apart from the conventional U-Net is the incorporation of distinctive components in the initial network (referred to as "NETWORK 1"). These components encompass the utilization of VGG-19, denoted by the yellow highlighting, ASPP highlighted in blue, and a decoder block accentuated in light green. Notably, the encoder of NETWORK 1 as well as the decoder blocks within both NETWORK 1 and NETWORK 2 incorporate the squeeze-and-excite block, as established in reference [53]. Moreover, an element-wise multiplication operation is executed between the output of NETWORK 1 and its corresponding input within the same network [7]. The difference between DoubleU-Net and U-Net within the second network (NETWORK 2) is solely attributed to the incorporation of ASPP and the squeeze-and-excite block. All other elements remain consistent.

Within the scope of NETWORK 1, the initial step involves directing the input image to the adapted U-Net architecture. This U-Net undertakes the task of generating a mask that serves as a prediction, labeled as Output1. Subsequently, a multiplication operation ensues, involving the input image and the resultant Output1 mask. This hybrid result then operates as the input for a second iteration of the modified U-Net. This secondary U-Net generates a distinct mask, denoted as Output2. In the final stages, the masks Output1 and Output2 are seamlessly combined through concatenation. This amalgamation serves the purpose of facilitating an insightful assessment of the qualitative differences between the interim mask (Output1) and the ultimate anticipated mask (Output2)[7].

It is assumed that further enhancement is attainable for the output feature map generated by NETWORK 1. This improvement is believed to be achievable

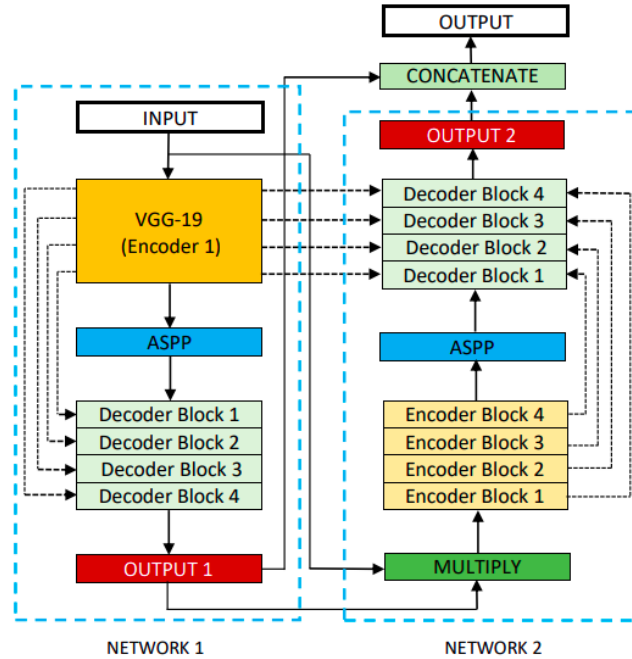


Figure 2.12: Block diagram of the proposed DoubleU-Net architecture

by retrieving the input image alongside its corresponding mask once more. Subsequently, combining this retrieved input with Output2 through concatenation is expected to yield a more refined segmentation mask compared to the previous iteration. This fundamental rationale serves as the driving force behind the incorporation of two U-Net architectures within the proposed framework. The integration of the squeeze-and-excite block within these networks contributes to the reduction of redundant information, facilitating the propagation of the most pertinent and meaningful data[7].

The incorporation of the ASPP technique is motivated by its recognized efficacy in modern segmentation architectures. ASPP's propensity to extract high-resolution feature maps is instrumental in elevating overall performance, as documented in contemporary studies [37].

#### A. ENCODER EXPLANATION

In the context of DoubleU-Net, the first encoder (encoder1) adopts a pre-trained VGG-19 model, whereas the second encoder (encoder2) is constructed from scratch. The primary goal of each encoder is to effectively encode the information encapsulated within the input image. Within the encoder2 archi-

ture, each encoder block executes a dual  $3 \times 3$  convolution operation, with each convolution followed by a batch normalization step. The integration of batch normalization serves a dual purpose: it mitigates the effects of internal co-variate shift and imparts regularization to the model. Subsequent to the convolution and normalization steps, an activation function in the form of ReLU is applied. This introduction of non-linearity enhances the model's capacity to capture intricate relationships within the data. Notably, a squeeze-and-excitation block follows the ReLU activation, significantly elevating the quality of the resulting feature maps.

Following the enhancement through the squeeze-and-excitation mechanism, a strategic max-pooling process is implemented. Max-pooling entails the use of a  $2 \times 2$  window with a stride of 2, effectively reducing the spatial dimensions of the feature maps. This dimension reduction contributes to a more compact representation while simultaneously retaining essential information. Collectively, the combination of convolution operations, normalization, activation functions, squeeze-and-excitation blocks, and max-pooling substantiates the systematic approach adopted by the encoder2 to extract and refine the feature map from the input data [7].

## B. DECODER EXPLANATION

As depicted in Figure 2.12, the overall network employs two decoders, featuring subtle adjustments compared to the original U-Net decoder. Within this framework, each decoder block undertakes a  $2 \times 2$  bi-linear up-sampling operation on the input feature. This operation effectively doubles the dimensions of the input feature maps. Subsequently, an essential step involves the concatenation of pertinent skip connection feature maps from the encoder to the output feature maps. In the initial decoder, solely a skip connection from the first encoder is utilized. Conversely, the second decoder capitalizes on skip connections from both encoders, a measure aimed at preserving spatial resolution while concurrently heightening the quality of the output feature maps.

Following the concatenation process, two consecutive  $3 \times 3$  convolution operations are performed on the feature maps. Each convolution operation is succeeded by batch normalization and a subsequent ReLU activation function. This sequence of operations contributes to feature refinement and non-linear transformation. Furthermore, a squeeze-and-excitation block is implemented,

## 2.5. DUAT NETWORK

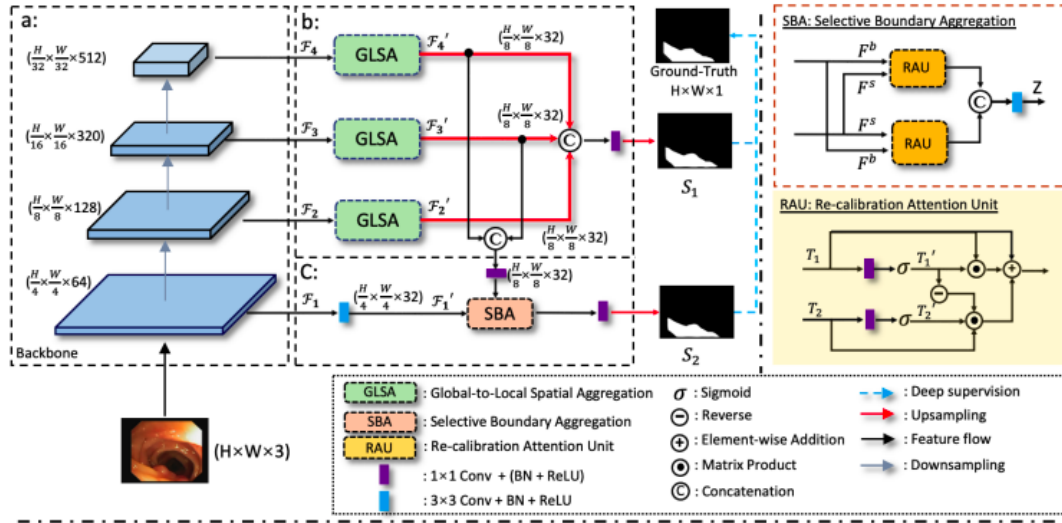


Figure 2.13: The overall architecture of Dual-Aggregation Transformer Network (DuAT). The entire model is divided into three parts: (a) pyramid vision transformer (PVT) as backbone; (b) pyramid Global-to-Local Spatial Aggregation (GLSA) Module; (c) Selective Boundary Aggregation (SBA) module and it shown on the red box

further enhancing the feature maps' quality by selectively emphasizing relevant information. The concluding step involves the application of a convolution layer with a sigmoid activation function. This layer is instrumental in generating the mask tailored to the modified U-Net at hand. The mask, serving as a prediction, encapsulates the spatial distribution of salient features for the specific U-Net configuration under consideration[7].

## 2.5 DuAT NETWORK

Proposed by Feilong Tang, et al[44], introduces a Dual-Aggregation Transformer Network called DuAT, characterized by two innovative designs, namely, the Global-to-Local Spatial Aggregation and Selective Boundary Aggregation modules, which are beneficial for locating large and small objects, respectively.

According to Figure 2.13[44], DuAT model comprises three main components: a pyramid vision transformer (PVT) encoder, an SBA module, and a GLSA module.

### 2.5.1 TRANSFORMER ENCODER

Recent research studies [43] indicate that vision transformers exhibit superior performance and resilience to input disturbances, such as noise, compared to CNNs. Motivated by these findings, we employ a Transformer based on a pyramid structure as the encoder. Specifically, we utilize the pyramid vision transformer (PVT) as the encoder module to extract multi-level feature maps extraction.

$$\{F_i \mid i \in (1, 2, 3, 4)\} \quad (2.1)$$

These feature maps consist of  $F_1$ , which captures detailed boundary information of the target, while  $F_2$ ,  $F_3$ , and  $F_4$  offer high-level features.

### 2.5.2 SELECTIVE BOUNDARY AGGREGATION

Shallow and deep-layer features have complementary characteristics. Shallow layers possess fewer semantics but offer intricate details, featuring distinct boundaries and minimal distortion. On the other hand, deeper layers encompass extensive semantic information. Merging low-level features directly with high-level ones may lead to redundancy and inconsistency. To tackle this issue, DuAT introduces the SBA module. This module selectively combines boundary and semantic information, enabling a more refined depiction of object contours and aiding in recalibrating object positions. DuAT developed a new Re-calibration attention unit (RAU) block that intelligently captures shared representations from two inputs ( $F^s, F^b$ ) before merging them, unlike previous fusion methods. Figure 2.13 illustrates how the shallow and deep-level information is separately directed into two RAU blocks. This setup compensates for the absence of spatial boundary information in high-level semantic features and the lack of semantic information in low-level features. Afterward, the outputs from these RAU blocks are combined following a  $3 \times 3$  convolution, resulting in a robust combination of diverse features that improves the quality of the raw features. The process conducted by the RAU block function, denoted as  $\text{PAU}(\cdot, \cdot)$ , is expressed as shown below:

$$T'_1 = W_\theta(T_1), \quad T'_2 = W_\phi(T_2) \quad (2.2)$$

## 2.5. DUAT NETWORK

$$PAU(T_1, T_2) = T'_1 \odot T_1 + T'_2 \odot T_2 \odot (\ominus(T'_1)) + T_1 \quad (2.3)$$

The resulting maps  $T_1, T_2$  are obtained through point-wise multiplication denoted by  $\odot$ . The operation  $\ominus(\cdot)$  acts in reverse by subtracting the feature  $T'_1$ . This refines the initial imprecise and coarse estimation, enhancing it into a precise and comprehensive prediction map. To perform the linear mapping, a convolutional operation is employed using a kernel size of  $1 \times 1$ . In conclusion, the SBA procedure can be stated in below mathematical formula:

$$Z = C3 \times 3(\text{Concat}(PAU(F_s, F_b), PAU(F_b, F_s))) \quad (2.4)$$

The function  $C3 \times 3(\cdot)$  represents a  $3 \times 3$  convolutional operation followed by batch normalization and a ReLU activation layer.  $F_s$  contains deep-level semantic information, formed after merging the third and fourth layers from the encoder. On the other hand,  $F_b$  represents the initial layer, preserving rich boundary details from the backbone network. The operation  $\text{Concat}(\cdot)$  denotes concatenation along the channel dimension.

### 2.5.3 GLOBAL-TO-LOCAL SPATIAL AGGREGATION

The attention mechanism emphasizes information relevant to the optimization goal while dampening irrelevant details. To encompass both global and local spatial characteristics, DuAT introduces the GLSA module, merging outcomes from distinct local and global attention units. Illustrated in Figure 2.14, this approach of having two separate streams efficiently retains both local and non-local modeling capabilities.

To achieve a balance between accuracy and computational efficiency, DuAT opts to segregate channels. Precisely, the feature map containing 64 channels is evenly divided into two groups. These groups are individually directed into the Global Spatial Attention (GSA) module and the Local Spatial Attention (LSA) module. The resulting outputs from these two attention units are ultimately combined by concatenation, followed by a  $1 \times 1$  convolutional layer. The Global Spatial Attention (GSA) module highlights the extensive connections among pixels within the spatial context, serving as a complementary addition to local spatial attention. The LSA module adeptly identifies and extracts local features within the spatial dimension of the provided feature map, particularly empha-



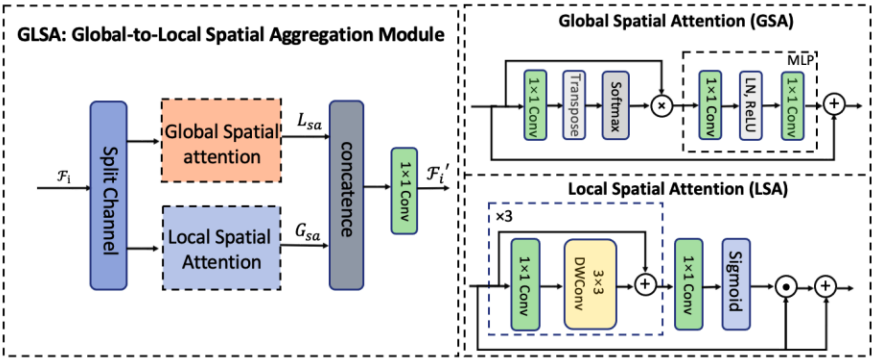


Figure 2.14: Overview of the Global-to-Local Spatial Aggregation Module GLSA, it is composed of global spatial attention (GSA) and local spatial attention (LSA)

sizing the region of interest, which could encompass details related to smaller objects.



# 3

## Background of Superpixel Labeling

### 3.1 SUPERPIXEL METHODS

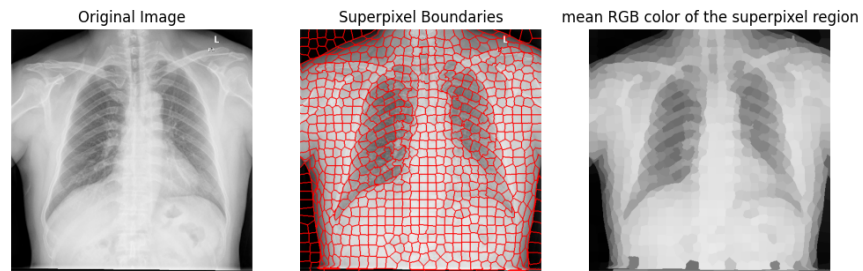
Superpixel algorithms group pixels into coherent and meaningful atomic regions, offering a flexible alternative to the fixed structure of the pixel grid. These regions capture redundant image information, serve as a convenient basis for deriving image features, and notably simplify subsequent image processing tasks. They have become integral components in various computer vision algorithms, such as leading solutions for multi-class object segmentation in the PASCAL VOC Challenge, depth estimation, segmentation, body model estimation, and object localization. Below pictures; figure 3.1, shows superpixel boundaries overlaid on two different medical images including skin lesion image and also chest MRI.

Numerous techniques exist for generating superpixels, each with distinct merits and limitations that can align better with specific applications. For instance, when maintaining fidelity to image boundaries is crucial, the graph-based approach described might be optimal. Conversely, if the goal is to construct a graph from superpixels, a method producing a more standardized lattice, could be preferable[76].

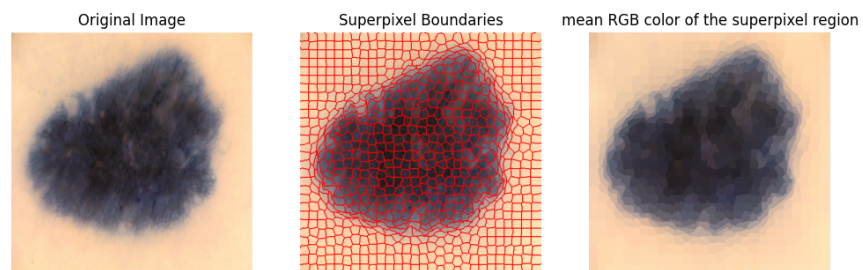
Although establishing a universally ideal approach across all applications is challenging, certain attributes are generally desirable for superpixel methods:

- Strong adherence to image boundaries.

### 3.1. SUPERPIXEL METHODS



(a) Chest MRI



(b) Skin Lesion

Figure 3.1: (a) Chest MRI Original Image, superpixel boundaries and mean RGB color of superpixel region (a) Skin Lesion Original Image, superpixel boundaries and mean RGB color of superpixel region

	GS04 [8]	NC05 [23]	Graph-based			Gradient-ascent-based				
			SL08 [21]	GCa10 <sup>b</sup> [26]	Gcb10 <sup>b</sup> [26]	WS91 [28]	MS02 [4]	TP09 <sup>b</sup> [15]	QS09 [25]	SLIC
Adherence to boundaries										
<i>Under-segmentation error</i> (rank)	0.23	0.22	-	0.22	0.22	-	-	0.24	0.20	<b>0.19</b>
<i>Boundary recall</i> (rank)	<b>0.84</b>	0.68	-	0.69	0.70	-	-	0.61	0.79	0.82
Segmentation speed										
320 × 240 image	1.08s <sup>a</sup>	178.15s	-	5.30s	4.12s	-	-	8.10s	4.66s	<b>0.36s</b>
2048 × 1536 image	90.95s <sup>a</sup>	N/A <sup>c</sup>	-	315s	235s	-	-	800s	181s	<b>14.94s</b>
Segmentation accuracy (using [11] on MSRC)	74.6%	75.9%	-	-	73.2%	-	-	62.0%	75.1%	<b>76.9%</b>
Control over amount of superpixels	No	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes
Control over superpixel compactness	No	No	No	No <sup>d</sup>	No <sup>d</sup>	No	No	No	No	Yes
Supervoxel extension	No	No	No	Yes	Yes	Yes	No	No	No	Yes

Figure 3.2: Summary of Existing Superpixel Algorithms.

- Efficiency in computation for use as a preliminary step in reducing computational complexity, along with low memory overhead and ease of use.
- Enhanced speed and quality of segmentation results when employed for segmentation purposes.

### 3.1.1 EXISTING SUPERPIXEL METHODS

Methods aimed at producing superpixels can be generally divided into two main groups: graph-based techniques and gradient ascent approaches. In the following sections, we delve into well-known techniques within each of these categories, encompassing some methods that weren't initially developed with the explicit purpose of generating superpixels. Figure 3.2 [76] furnishes both qualitative and quantitative overviews of the methods under review, along with their comparative performance evaluations.

#### A. GRAPH-BASED ALGORITHMS

NC05, known as the Normalized cuts algorithm, employs a recursive approach to partition an image's entire pixel graph, utilizing contour and texture hints. This process involves minimizing a global cost function applied to the edges that define the partition boundaries. Notably, NC05 yields highly uniform and visually appealing superpixels. Nevertheless, it exhibits limited adherence to boundaries, and its computational efficiency is relatively slow, especially for larger images. There have been efforts to enhance its speed[55].

GS04, introduced by Felzenszwalb and Huttenlocher, presents an alternative method rooted in graph-based principles, effectively employed for superpixel

### 3.1. SUPERPIXEL METHODS

generation. This approach engages in an agglomerative clustering of pixels represented as nodes within a graph. This results in superpixels formed as minimum spanning trees of constituent pixels. While GS04 effectively aligns with image boundaries, the superpixels it generates exhibit notable variation in sizes and shapes. The method demonstrates an  $O(N \log N)$  complexity, rendering it practically swift. Yet, it lacks explicit capabilities for controlling the number of superpixels or their compactness[71].

**SL08**, developed by Moore et al., presents an approach for generating grid-conforming superpixels by identifying optimal paths, referred to as seams, that divide the image into vertical or horizontal segments. The discovery of these optimal paths involves a graph cuts technique akin to the one used in Seam Carving[28].

**GCa10** and **GCb10**, as discussed in[102], present an approach by Veksler et al. that employs a global optimization strategy similar to the texture synthesis concept presented in[87]. The process involves assembling superpixels through the merging of overlapping image patches, ensuring that each pixel exclusively associates with a single overlapping area. The authors introduce two versions of their approach, catering to the creation of compact superpixels (GCa10) and constant-intensity superpixels (GCb10).

## B. GRADIENT-ASCENT-BASED ALGORITHMS

Beginning with an initial coarse pixel clustering, gradient ascent techniques iteratively enhance the clusters until reaching a convergence point, ultimately creating superpixels.

**MS02**, detailed in[96], employs mean shift, an iterative procedure for identifying local density function maxima, to detect modes within an image's color or intensity feature space. Pixels that converge to the same mode collectively form the superpixels. Although MS02 serves as an older technique, it generates superpixels with non-uniform sizes and irregular shapes.

**QS08**, described in Quick shift[26], adopts a mode-seeking segmentation method. The process commences with medoid shift initialization for segmentation. Subsequently, each point within the feature space is relocated to its nearest neighbor, optimizing Parzen density estimation. While QS08 demonstrates notable adherence to boundaries, its computational efficiency is limited, characterized by  $O(dN^2)$  complexity (where  $d$  is a small constant[26]). Unfor-

tunately, QS08 lacks the provision for direct manipulation of superpixel size or quantity. QS08 has been previously employed in works related to object localization and motion segmentation.

**WS91**, outlined in the watershed methodology[62], employs a gradient ascent technique that initiates from local minima to generate watersheds—dividing lines for catchment basins. The resultant superpixels frequently display irregularity in both size and shape, lacking effective boundary conformity. This [62] approach exhibits noteworthy speed  $O(N \log N)$  complexity, yet it doesn't afford direct manipulation of the superpixel quantity or compactness.

**TP09**, known as the Turbopixel method, employs a gradual expansion of seed positions through geometric flow based on level-set techniques [25]. This flow, driven by local image gradients, strives to ensure a uniform distribution of superpixels across the image plane. Unlike WS91, TP09 maintains consistent superpixel size, compactness, and adherence to boundaries. TP09 employs algorithms with varying complexities, but according to the authors, it demonstrates an approximate  $O(N)$  behavior [25] in practice. Nonetheless, it's one of the slower algorithms assessed and displays relatively limited boundary adherence.

## 3.2 SLIC

In this section, we will explain the SLIC superpixel algorithm. SLIC was proposed by Radhakrishna Achanta et al[76] in 2011. SLIC is a modified version of the k-means algorithm used to create superpixels in images. It differs from k-means in two key ways:

- Firstly, it drastically minimizes the number of distance calculations needed for optimization by confining the search area to a region that corresponds to the size of the superpixel. This simplification results in a computational complexity that scales linearly with the number of pixels ( $N$ ), regardless of the number of superpixels ( $k$ ).
- Secondly, SLIC employs a weighted distance metric that takes into account both color and spatial proximity. This combined distance measure not only ensures color and spatial coherence but also offers a means to adjust the superpixel size and compactness.

## 3.2. SLIC

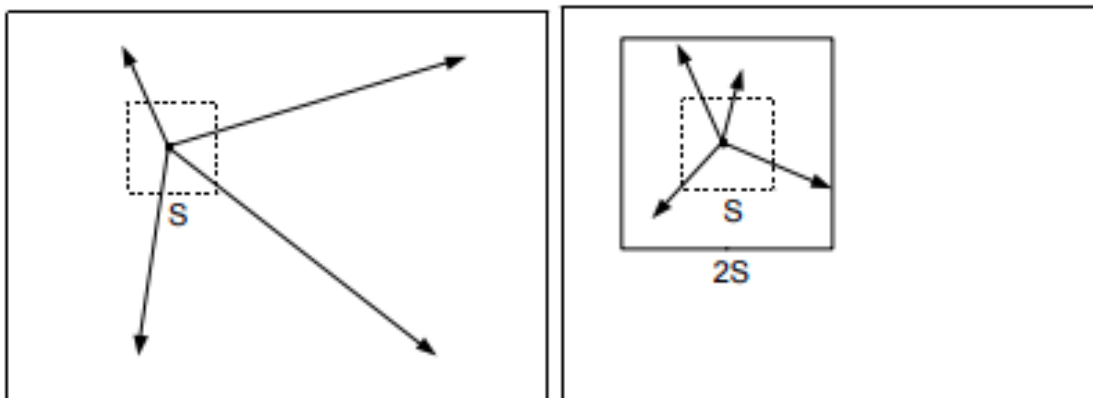
### 3.2.1 A. ALGORITHM

SLIC offers a user-friendly and comprehensible approach. By default, the algorithm relies on a single parameter, denoted as "k" which signifies the desired quantity of roughly uniform superpixels. When working with color images in the CIELAB color space, the clustering process kicks off with an initialization phase. During this step, initial cluster centers denoted as  $C_i$  are established. These centers are represented as  $C_i = [l_i, a_i, b_i, x_i, y_i]^T$ , where  $l_i$ ,  $a_i$  and  $b_i$  represent color values, and  $x_i$  and  $y_i$  indicate pixel coordinates. These centers are strategically positioned on a grid with an interval labeled as  $S$  which amounts to the square root of the total number of pixels divided by the number of desired superpixels  $S = \sqrt{\frac{N}{k}}$ . To attain approximately equivalent superpixel sizes, these centers are then relocated to seed points that correspond to areas of lowest gradient within a 3x3 neighborhood. This shift ensures that superpixels are not centered on edges and mitigates the risk of basing a superpixel on a noisy pixel.

In the assignment phase, each pixel  $i$  is matched with the nearest cluster center whose search region encompasses its location, as illustrated in Figure 3.3[76]. This step is pivotal for accelerating the algorithm, as constraining the search region's size significantly reduces the number of required distance calculations. Consequently, this results in a substantial speed advantage over conventional k-means clustering, where each pixel needs to be compared against all cluster centers. This optimization is feasible due to the introduction of a distance measure  $D$ , which determines the closest cluster center for every pixel. This concept is elaborated upon in Section 3.2.2. As the anticipated spatial span of a superpixel is an approximate region of size  $S \times S$  the quest for similar pixels is conducted within a  $2S \times 2S$  region around the superpixel's center.

After associating each pixel with its closest cluster center, a subsequent update phase modifies the cluster centers to become the average  $[labxy]^T$  vector encompassing all pixels assigned to the particular cluster. The computation of a residual error  $E$  between the updated cluster center positions and their previous positions is conducted using the L2 norm. The process of assigning and updating can be iteratively performed until the error converges. To ensure connectivity, a post-processing step is implemented, which involves reassigning disconnected pixels to adjacent superpixels. The entirety of the algorithm is succinctly outlined in Algorithm 1[76].





(a) k-means searches the entire image

(b) SLIC searches a limited region

Figure 3.3: Reducing the superpixel search regions. The complexity of SLIC is linear in the number of pixels in the image  $O(N)$ , while the conventional k-means algorithm is  $O(kNI)$  where  $I$  is the number of iterations. This is achieved by limiting the search space of each cluster center in the assignment step. (a) In the conventional k-means algorithm, distances are computed from each cluster center to every pixel in the image. (b) SLIC only computes distances from each cluster center to pixels within a  $2S \times 2S$  region. Note that the expected superpixel size is only  $S \times S$ , indicated by the smaller square. This approach not only reduces distance computations but also makes SLIC's complexity independent of the number of superpixels.

---

**Algorithm 1** SLIC Superpixel Segmentation

---

```

/*Initialization*/
Initialize cluster centers ( $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ ) by sampling pixels at regular
grid steps  $S$ .
Move cluster centers to the lowest gradient position in a  $3 \times 3$  neighborhood.
Set label  $l(i) = -1$  for each pixel  $i$ .
Set distance  $d(i) = \infty$  for each pixel  $i$ .
repeat
  for each cluster center  $C_k$  do
    for each pixel  $i$  in a  $2S \times 2S$  region around  $C_k$  do
      Compute the distance  $D$  between  $C_k$  and  $i$ .
      if  $D < d_i$  then
        Set  $d_i = D$ 
        Set  $l_i = k$ 
      end if
    end for
  end for
  Compute new cluster centers.
  Compute residual error  $E$ .
until  $E \leq \text{threshold}$ 

```

---

**3.2.2** B. DISTANCE MEASURE

SLIC superpixels align with clusters in the color-space of the  $labxy$  image plane. This situation poses a challenge when determining the distance measure  $D$ , which might not be immediately evident. In Algorithm 1,  $D$  calculates the separation between a pixel  $i$  and the center  $C_k$  of a cluster. The color of a pixel is denoted in the CIELAB color space  $[lab]^T$ , where its potential value range is established. Conversely, the pixel's position  $[xy]^T$  can encompass a range of values that fluctuates based on the image's dimensions.

Merely defining  $D$  as the Euclidean distance in the five-dimensional  $labxy$  space would lead to irregularities in the clustering behavior for varying superpixel sizes. When dealing with larger superpixels, spatial distances become more significant than color proximity, resulting in spatial closeness being relatively more important than color. Consequently, this leads to the creation of compressed superpixels that do not align effectively with the image boundaries. Conversely, for smaller superpixels, the opposite holds true.

In order to merge these two distances into a unified measure, it becomes essential to normalize color proximity and spatial proximity based on their

respective maximum distances within a cluster, denoted as  $N_s$  and  $N_c$ . By carrying out this normalization, the expression for  $D'$  is formulated as follows:

$$\begin{aligned} d_c &= \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \\ d_s &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \\ D' &= \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2} \end{aligned} \quad (3.1)$$

The largest anticipated spatial distance within a specific cluster ought to align with the sampling interval, represented as  $N_s = S = \sqrt{\frac{N}{K}}$ . However, establishing the maximum color distance  $N_c$  is less straightforward, given that color distances can exhibit notable discrepancies across clusters and images. This challenge can be circumvented by setting  $N_c$  as a constant  $m$ , which results in the following transformation of Eq. 3.1[76]:

$$D' = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2} \quad (3.2)$$

This simplification leads to the distance measure that we utilize in practical applications.

By adopting this definition of  $D$ , the parameter  $m$  enables us to control the balance of significance between color similarity and spatial proximity. With a larger  $m$ , spatial proximity gains prominence, leading to more condensed superpixels characterized by a lower area-to-perimeter ratio. Conversely, a smaller  $m$  results in superpixels that conform more closely to image boundaries, although their size and shape may be less consistent. When working within the CIELAB color space, the value of  $m$  can be selected from the range of 1, 40.

For grayscale images, Equation 3.3[76] can be adjusted by assigning  $dc = \sqrt{(l_j - l_i)^2}$ . Additionally, the equation can be expanded to accommodate 3D supervoxels, as illustrated in Figure 3.4[76], by incorporating the depth dimension into the spatial proximity term of the equation.

$$D' = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2} \quad (3.3)$$

### 3.2. SLIC

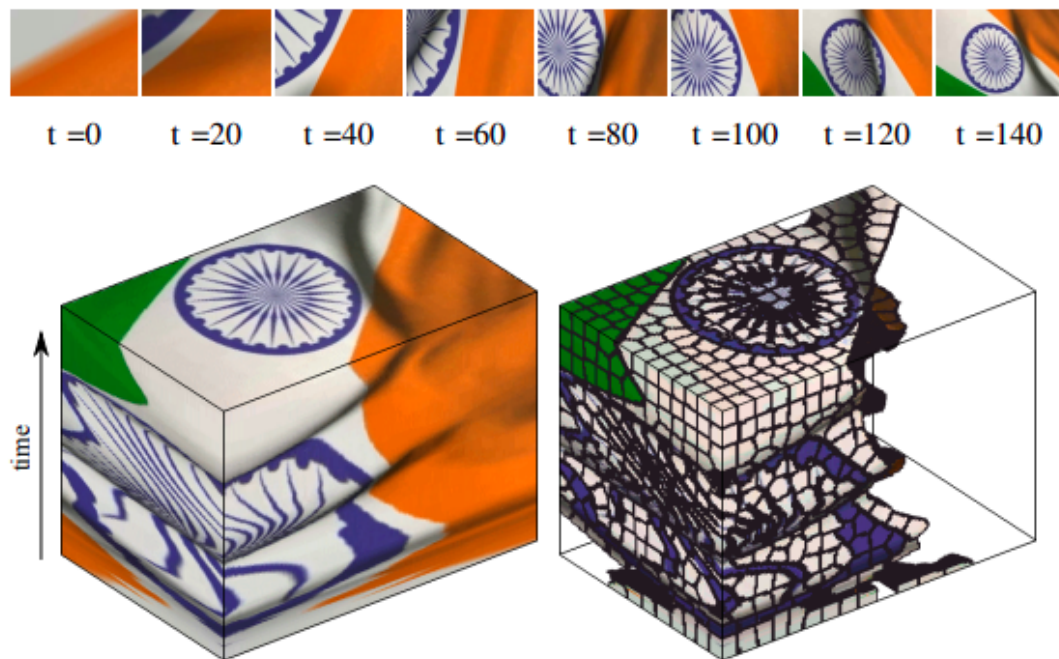


Figure 3.4: SLIC supervoxels computed for a video sequence. (top) frames from a short video sequence of a flag waving. (bottom left) A volume containing the video. The last frame appears at the top of the volume. (bottom right) A supervoxel segmentation of the video. Supervoxels with orange cluster centers are removed for display purpose

### 3.2.3 C. POST-PROCESSING

Similar to several other superpixel algorithms, SLIC does not inherently ensure connectivity. After the clustering process, certain pixels, referred to as "orphaned," might not be part of the same connected component as their cluster center. To address this, these pixels are allocated the label of the closest cluster center using a connected components algorithm.

## 3.3 FH

The FH superpixel algorithm is a popular method for generating superpixels in an image. It was introduced by Pedro F. Felzenszwalb and Daniel P. Huttenlocher in their paper "Efficient Graph-Based Image Segmentation"[45].

The FH algorithm is based on a graph-based approach to image segmentation. The idea is to treat the image as a graph, where each pixel is a node, and edges are defined based on color similarity and spatial proximity. The goal of the algorithm is to group pixels into regions of similar color and spatial proximity, resulting in superpixels. Here's a step-by-step explanation of how the FH algorithm works:

- **Edge Weight Calculation:** The first step involves calculating edge weights between neighboring pixels. Each pixel's color information and spatial position contribute to these weights. The color difference between two pixels is measured using a color distance metric, such as Euclidean distance in the RGB or Lab color space. The spatial distance between two pixels is the Euclidean distance between their positions.
- **Graph Construction:** Using the calculated edge weights, a graph is constructed. Pixels in the image represent nodes, and the edges between them represent connections based on their color and spatial similarity.
- **Graph Segmentation:** In the graph, each pixel initially forms its own segment. The algorithm then starts merging adjacent segments that have low edge weights. The merging process continues iteratively, with the algorithm focusing on merging segments that exhibit small color and spatial differences. The threshold for merging is determined based on the average edge weight of the segments being merged. This means segments that are more similar in color and spatial proximity are more likely to be merged.
- **Superpixel Generation:** After the merging process is complete, the resulting segments in the graph correspond to superpixels. These segments represent regions of similar color and spatial coherence, and they are effectively the superpixel regions.

### 3.4. QUICKSHIFT

- **Post-processing:** Depending on the application, some post-processing steps might be applied. These steps can involve refining the boundaries of the superpixels, enforcing a minimum size for the superpixels, or adjusting the superpixel shape to better align with object boundaries.

The FH superpixel algorithm's key features include its computational efficiency and its ability to generate superpixels without requiring a predefined number of segments. Its performance is generally robust for various types of images, capturing both texture and color-based regions effectively.

## 3.4 QUICKSHIFT

### 3.4.1 PROCEDURE

QS is a superpixel segmentation algorithm that was introduced by Vedaldi, A et al [105]. It is an iterative method that seeks to group similar pixels together based on color and spatial proximity. QS is known for its simplicity, speed, and adaptability to various image characteristics.

Here's a detailed explanation of how the QS superpixel method works:

- **Feature Space:** QS operates in a high-dimensional feature space that combines pixel color information (and potentially spatial position). Each pixel is treated as a point in this space.
- **Density Estimation:** The algorithm estimates the density of points in the feature space. For each pixel, QS calculates the similarity with its neighboring pixels. This similarity can be computed using color differences and spatial distances. Higher similarity implies pixels that are closer in color and space.
- **Mode Seeking:** The algorithm starts from every pixel and iteratively seeks "modes," which are densely populated regions in the density estimate. A mode is a pixel that has a high density value. The algorithm simulates a process where a particle (representing a pixel) moves to neighboring particles with higher density. This process continues until convergence is reached.
- **Similarity-Based Grouping:** Pixels that converge to the same mode are grouped together into a superpixel. If two pixels end up converging to the same mode, it indicates they are likely part of the same homogeneous region in the image.
- **Hierarchical Clustering (Optional):** QS can be used hierarchically by varying the "bandwidth" parameter. Multiple iterations with different bandwidth values can produce a hierarchy of superpixels at different scales.

- **Compactness Control (Optional):** QS offers a parameter that lets you control the compactness of the superpixels. By adjusting this parameter, you can influence whether the superpixels are more compact or elongated.
- **Post-processing (Optional):** Depending on your specific needs, you can apply post-processing steps like enforcing a minimum size for the superpixels or refining the superpixel boundaries.

### 3.4.2 IMPORTANT ASPECTS

- **Speed and Efficiency:** QS is known for its speed and efficiency due to its iterative mode-seeking process.
- **Adaptability:** It adapts to the image content and doesn't require a pre-defined number of superpixels.
- **Regular Shapes:** QS's superpixels might not be as regularly shaped as those from other algorithms.
- **Fine Details and Texture:** It might struggle with capturing fine details and texture variations in images.
- **Applications:** QS is used in a variety of computer vision tasks, such as object recognition, image segmentation, and video analysis.
- **Parameter Tuning:** Depending on your data and requirements, you might need to adjust parameters to achieve the desired results.

In summary, QS is a versatile and efficient superpixel algorithm that generates regions of similar appearance in an image. Its simplicity and speed make it a popular choice for a wide range of applications.

## 3.5 SEEDS

SEEDS incorporates a hill-climbing optimization technique for extracting superpixels. Hill-climbing is an iterative optimization algorithm that improves the solution by making small local changes at each step. If the proposed partitioning results in a higher energy function value, the solution will be updated. SEEDS represents the proposed partitioning as  $s \in S$ , and  $s_t$  represents the partitioning with the lowest energy found at time  $t$ . To generate a new partitioning  $s$ , they introduce local changes to  $s_t$  by moving pixels between neighboring superpixels. The hill-climbing algorithm is highly efficient as evaluating these small changes to the partitioning can be done quickly in practice.

### 3.5. SEEDS

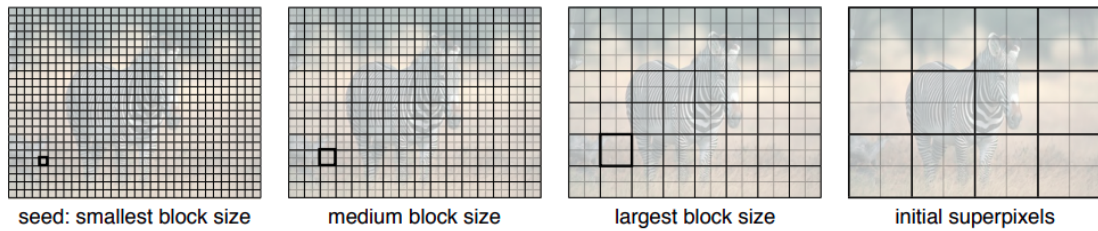


Figure 3.5: Initialization. Example of initialization with 12 superpixels and blocks of different sizes. The initialization occurs from left to right: first the smallest blocks are initialized, and then concatenated  $2 \times 2$  to form larger blocks. The largest blocks are concatenated  $2 \times 2$  to create the initial superpixels. This rectangular grid (in this case  $4 \times 3$ ) is the starting point of the SEEDS algorithm.

#### 3.5.1 INITIALIZATION

In the context of hill-climbing, starting from an effective initial partition is crucial to approach a solution that closely approximates the global best ( $s^*$ ). We propose employing a regular grid as an initial rough partition, aligning with the spatial limitations of the superpixels to form a partition in  $S$ . SEEDS’s experiments indicate that when comparing this grid with standard evaluation metrics, its performance is commendable: it achieves reasonable over-segmentation, albeit unable to accurately recover object boundaries. Notably, object boundaries are at most half the grid size away from the grid edges. This rationale justifies the use of hill-climbing optimization for superpixel extraction, given that the starting point is relatively close to the optimal solution.

Moreover, SEEDS initiate the pixel blocks utilized for block movements at various sizes and compute a color histogram for each block. Initially, it create the smallest block size, comprising  $2 \times 2$  or  $3 \times 3$  pixels. To create larger block sizes, it hierarchically combine the smaller blocks in a  $2 \times 2$  manner. The histograms corresponding to these larger blocks are derived by aggregating the histograms of the constituent smaller blocks, as illustrated in Figure 3.5[93].

The algorithm’s maximum block size is a quarter of the intended superpixel size. Consequently, the superpixels are initialized by combining  $2 \times 2$  blocks of this maximum size. This approach ensures the superpixels maintain a uniform size, regardless of the input image’s dimensions. Adjusting the initial block size and the number of block levels allows control over the desired quantity of superpixels.



```

 $s_t = \text{initialize}();$ 
while  $t < t_{stop}$  do
   $s = \text{Propose}(s_t);$ 
  if  $E(s) < E(s_t)$  then
     $s_t = s;$ 
  end
end
 $s^* = s_t;$ 

```

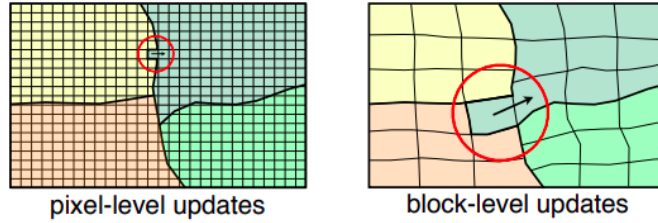


Figure 3.6: Left: algorithm. Right: movements at pixel-level and at block-level

### 3.5.2 PROPOSING PIXEL-LEVEL AND BLOCK-LEVEL MOVEMENTS

During each iteration, the algorithm suggests a fresh partitioning  $s$  based on the preceding one  $s_t$ . Alterations from  $s_t$  to  $s$  involve either individual pixels or clusters of pixels being relocated to an adjacent superpixel. Denoted as  $(A_k)^l$ , it signifies a potential set comprising one or more pixels meant for transfer from the superpixel  $A_k$  to its neighboring  $A_n$ . When addressing pixel-level updates,  $(A_k)^l$  includes a single pixel, while for block-level updates, it encompasses a small group of pixels, depicted in Figure 3.6[93]. During each iteration of the hill-climbing process, we create a new partitioning by randomly selecting  $(A_k)^l$  from all boundary pixels or blocks with an equal probability and assign the chosen  $(A_k)^l$  to a random neighboring superpixel  $A_n$ . If this process results in an invalid partitioning, only possible if a boundary movement splits a superpixel into two parts, it is disregarded.

Block-level updates are chosen for their efficiency, aiming to expedite convergence and circumvent local maximums. Although block-level updates incur higher computational costs, they relocate more pixels simultaneously. Consequently, initiating the algorithm with large block-level updates, progressing to smaller blocks, and culminating with pixel-level boundary adjustments is preferred. This hierarchical approach, transitioning from the largest to smaller block sizes and finally to individual pixels, is depicted in Figure 3.7[93]. Running the individual pixel updating for an extended duration enhances the accuracy of the resulting superpixels.

### 3.5.3 EVALUATING PIXEL-LEVEL AND BLOCK-LEVEL MOVEMENTS

In the following, the efficient evaluation of  $E(s)$ , and the efficient updating of the color distributions in case  $s$  is accepted will be described. SEEDS present a method to efficiently compute  $H(s)$  using the intersection distance. The pro-

### 3.5. SEEDS

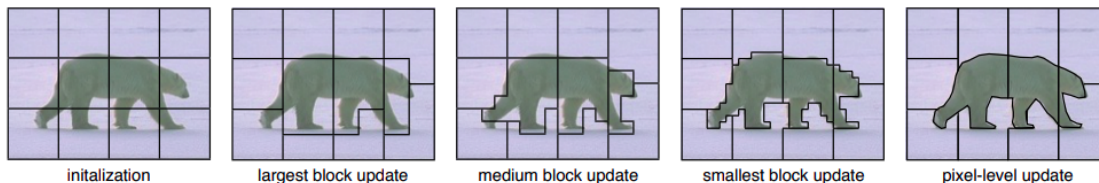


Figure 3.7: Block and pixel movements. This figure shows an example of the evolution of the superpixel boundaries while going through the iterations of the SEEDS algorithm (in the case of 12 superpixels). From left to right: The first image shows the initialization as a grid. The subsequent images show the block updates from large to small. The last image shows the pixel-level update of the superpixel boundaries

posed method introduces an efficient approach to evaluate  $H(s)$  based on the intersection distance between histograms. This method measures the similarity between two histograms by comparing the minimum values of their bins. The calculation involves a number of comparisons and sums equivalent to the number of bins in the histogram. Based on this method, assists in determining whether the energy function increases by computing two intersection distances. It relies on assumptions about the superpixels' sizes and the concentration of the histogram of candidate pixels  $(A_k)^l$  to be moved. If the size of  $(A_k)^l$  is considerably smaller than the superpixel size, and both superpixels are similar in size, the proposition holds true. Additionally, assuming the histogram of  $(A_k)^l$  is concentrated in a single bin, this proposition generally holds, especially when  $(A_k)^l$  represents a single pixel or a small block of pixels. Empirical results demonstrate that these assumptions are valid in 93 percent of cases during the algorithm's execution. Notably, when assessing a pixel-level update, the intersection computation can be achieved with a single memory access due to the structure of the pixel's color histogram, which typically activates only one bin.

The hierarchical approach used in updating boundaries enables the elimination of the boundary term while still achieving smooth superpixel boundaries. This strategy involves gradually updating boundaries from larger to finer scales, allowing for more efficient evaluation of the energy function  $E(s)$ . By omitting the boundary term, the method becomes more theoretically robust without the need for ad-hoc preferences that optimize subjective qualities. In the experimental section, the results are presented without using the boundary term, providing more efficient evaluation and theoretical soundness. However, for those seeking more control over the superpixel shapes, the SEEDS framework

allows incorporating the boundary term  $G(s)$  to influence the shape of the superpixels.

After accepting a new partition, the histograms of  $A_k$  and  $A_n$  need efficient updates. For pixel-level changes, updating involves incrementing and decrementing a single bin ( $j$ ) in the respective histograms. However, for block-level alterations, the update entails subtracting the histogram of  $(A_k)^l$  from that of  $A_k$  and adding it to  $A_n$  to achieve the modification.

### 3.5.4 TERMINATION

When the algorithm is stopped, a valid image partitioning is obtained, and its quality depends on the duration of the allowed runtime. Allowing the algorithm to run for longer periods typically results in higher values for the objective function. Termination often occurs during pixel-level updates of the boundaries. However, even if one opts to stop the algorithm very early, possibly during block-level updates, the algorithm still yields a valid partitioning of the image.

The ability to set  $t_{stop}$  based on the specific application's requirements or dynamically allocate a time budget on the go is a crucial feature. This property, often overlooked in the context of superpixel extraction, holds significant importance for online applications. Graph-based superpixel algorithms necessitate waiting until all graph cuts are added, while growing-based methods require waiting until the growing process is completed, which incurs non-negligible costs. In contrast, the hill-climbing approach, despite involving more iterations than previous methods, executes each iteration remarkably quickly. This rapid execution enables stopping the algorithm at any given time, as the time required to complete the ongoing iteration is insignificant.



# 4

## Our Method

In this research project, we use two specific types of CNN networks, U-Net and DoubleU-Net for the purpose of segmentation. Firstly, I will discuss the standard pipeline which has been widely used for image segmentation. Afterward, I will explain our new approach. The former pipeline is illustrated in 4.1. As it is shown in the picture, In the standard pipeline, we work with images and their corresponding ground truths. These ground truths are labeled pixel-wise. Then, both the original images and correspondense ground truts are given to a ML model. This provides us with both qualitative and quantitative results. The model's performance is also tested on new dataset, and the predictions are evaluated using various metrics which are suitable for image segmentation.

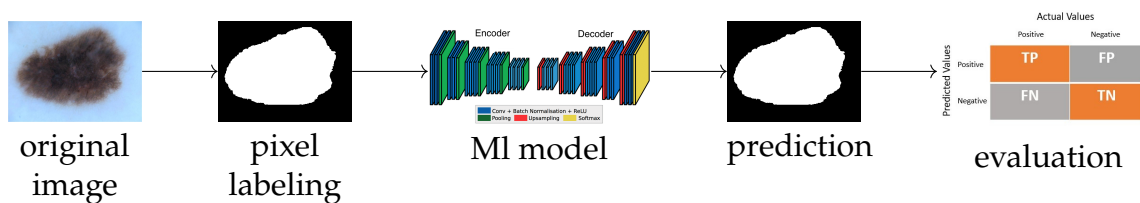


Figure 4.1: Standard Pipeline

Conversely, our pipeline 4.2 goes like this: First, we label the ground truths using larger groups called superpixels. Next, we select the right ML model and feed both the original images and the superpixel-labeled ground truths to the network. The results of the image segmentation are then evaluated with new test data set using a metric like the Dice Coefficient. mention that the evaluation will done between the segmentation results and the original pixel-wise ground

## 4.1. PREPROCESSING AND DATA AUGMENTATION

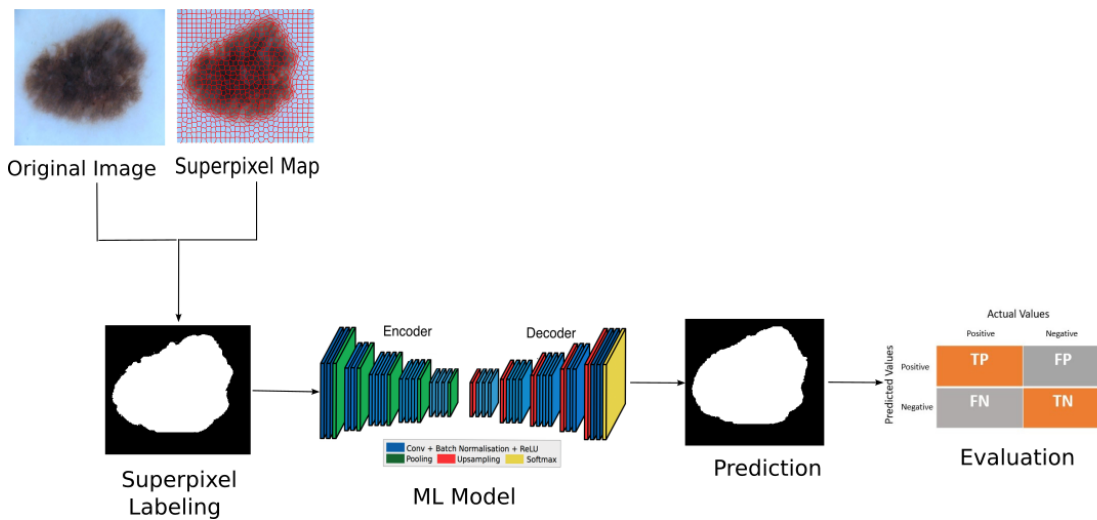


Figure 4.2: our pipeline

truths. It means that the superpixel labeled ground truths are used just in training part not in the test phase. furthermore, as mentioned we convert our ground truth in two ways called superpixel labeling and extended superpixel labeling which are going to be elaborated in section 4.2.

superpixels group pixels similar in color and other low-level properties. In this respect, superpixels address two problems inherent to the processing of digital images: firstly, pixels are merely a result of discretization; and secondly, the high number of pixels in large images prevents many algorithms from being computationally feasible. Ren and Malik introduce superpixels as more natural entities - grouping pixels which perceptually belong together while heavily reducing the number of primitives for subsequent algorithms.” You can read more about superpixels in

## 4.1 PREPROCESSING AND DATA AUGMENTATION

Preprocessing is a crucial step in preparing image datasets for segmentation using NN. Proper preprocessing can significantly improve the performance and convergence of your segmentation model. When subjected to any classification approaches, utilizing raw data results in unsatisfactory accuracy.

Resizing images before performing image segmentation with neural networks is important for several reasons:

**Consistency in Input Size:** Neural networks expect input data with consis-

tent dimensions. Resizing ensures that all images in your dataset have the same width and height. This consistency is crucial because neural networks process data in batches, and varying input sizes can lead to errors or inefficiencies during batch processing.

**Memory Efficiency:** Larger images consume more memory during training and inference. Resizing images to a manageable size can help you fit more data into your available memory, allowing you to train larger and more complex models without running into memory constraints.

**Reduced Computational Cost:** Smaller images require less computational resources for processing. This means that resizing images can lead to faster training and inference times. Faster processing is especially important when working with large datasets or resource-limited environments.

**Mitigating Overfitting:** Smaller images are less prone to overfitting. When dealing with limited data, using larger images might lead to the model memorizing the training data rather than generalizing patterns. Resizing can help prevent overfitting by reducing the effective complexity of the model.

**Handling Variable Image Sizes:** In some applications, your input images might come in various sizes. Resizing them to a consistent size makes it easier to design and train neural networks that can handle different images effectively.

**Preserving Aspect Ratio:** When resizing images, it's important to maintain the aspect ratio (the ratio of width to height). This ensures that objects in the images are not distorted. Resizing while preserving aspect ratio might lead to black borders (padding) on some sides of the image, but it's a better option than distorting the content.

**Matching Pretrained Models:** Many pre-trained neural network architectures, especially those for image classification, have specific input size requirements. Resizing your segmentation images to match these input sizes allows you to leverage pre-trained features, improving the performance of your segmentation model.

**Simpler Model Design:** Working with images of the same size simplifies the architecture of your neural network. You can set the input dimensions once and design the rest of the network without worrying about adapting to varying input sizes.

Due to the reasons mentioned above, we resize our images and also related ground truths to size of  $256 \times 256$ , before feeding them to our segmentation network. By this way, we prevent to have the above discrepancy happened.

## 4.2. SUPERPIXEL LABELING

Neural networks work better when input data is in a certain range, often close to 0. Normalizing pixel values to the  $[0, 1]$  range helps stabilize training. Additionally, it ensures that all features (pixels) contribute equally to the learning process, preventing some features from dominating due to larger scales.

In this project, The normalization is applied consistently to both the image and its corresponding ground truth. This is important because masks need to match the same normalization as the images they correspond to. This consistency ensures that the network receives normalized inputs and is able to learn meaningful relationships between them.

In the original images, pixel values range from 0 to 255, where 0 represents black (no intensity) and 255 represents the maximum intensity for a channel. To do so, we divide each pixel value by 255.0 which scales the values down to a range between 0 and 1. After the division, pixel values that were originally 255 become 1, pixel values that were 128 become around 0.5, and so on.

Acquiring and annotating medical datasets is a complex task. The majority of current datasets consist of a limited number of examples, posing a difficulty for training deep learning models on such data. A potential remedy to address the issue of insufficient data involves employing techniques for data augmentation, which can expand the sample count during the training process[36]. To implement this, we initiate by dividing the dataset into training, validation, and testing subsets. Subsequently, a range of data augmentation approaches such as center cropping, random rotation, transposition, elastic transformation, and more are applied to each of these subsets. each image transformed into 25 distinct images, resulting in total amount of 26 images encompassing the initial one. the same augmentation techniques were applied to both datasets. Figure 4.3 shows some of the augmented techniques including CenterCrop, ChannelShuffle, HueSaturation, OpticalDistortion and VerticalFlip applied to International Skin Imaging Collaboration (ISIC) dataset.

## **4.2** SUPERPIXEL LABELING

Considering that the objective of this project is to transform the existing ground truth into a superpixel-based representation and perform superpixel labeling, our initial focus has been on creating a new ground truth. To achieve this goal, our first step involves superpixel segmentation of the original image.



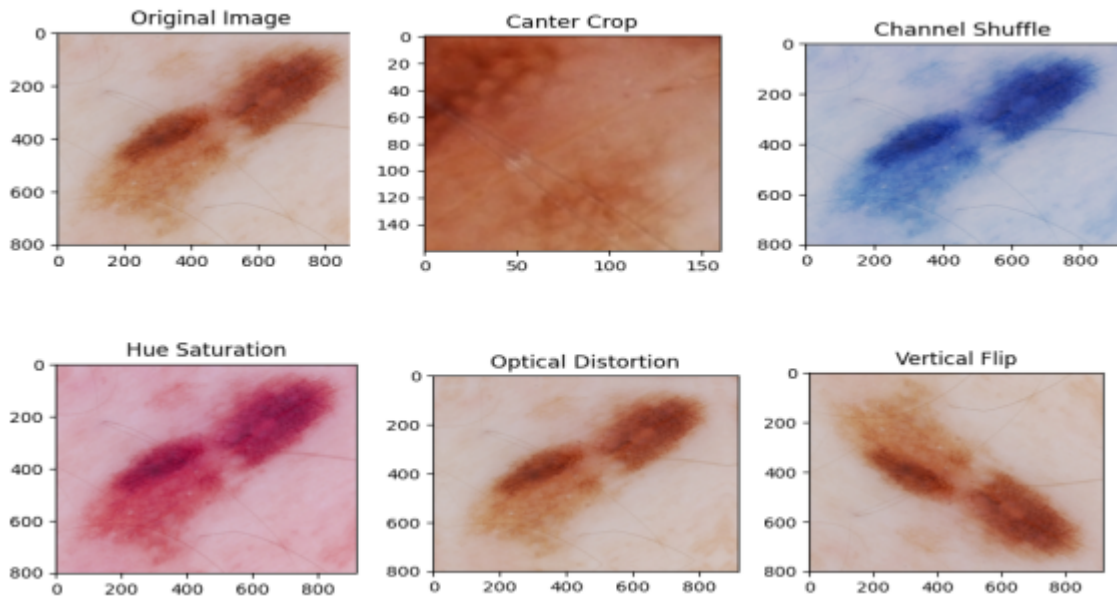


Figure 4.3: Sample Augmented Picture

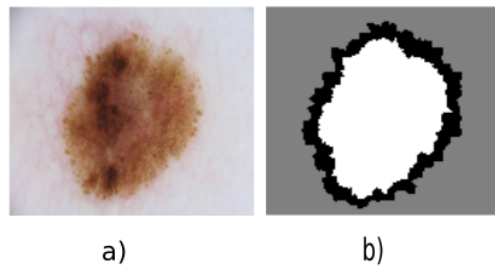


Figure 4.4: a) original image b) three lable superpixel ground truth

Subsequently, we identify superpixels that intersect with the image boundaries (which are distinctly visible in the binary ground truth). We will study the following approaches to automatically fill the boundary gap:

- No filling at all.
- A distance-based filling method.
- Use of the random walker segmentation algorithm.

In the first case there will be no labeling information available for boundary gap. figure 4.4 demonstrates the original image and the related three lable superpixel ground truth

Thus, the loss function used for training a DL model has to be adapted to exclude this part of the image.

## 4.2. SUPERPIXEL LABELING

The next two methods try to completely fill the gap in a meaningful manner. It thus transforms the superpixel-based sparse annotation into a pixel-based dense annotation. In distance-based filling we consider each pixel within an unlabeled superpixel. We determine its nearest distance to the foreground and background, and assign it to one of them dependent of which of them is nearer to the pixel.

The random walker segmentation algorithm was first proposed by L. Grady [59]. Today, it is one of the most popular interactive segmentation methods [103]. A user marks some seed pixels in each region. Then, the random walker method assigns all unseeded pixels to one of the regions using a probabilistic optimization scheme. This assignment is done by determining the region of the highest probability that a random walker starting from that pixel reaches the user provided scribbles (seeds). Random walker segmentation has found many applications [107]. In medical imaging, for instance, Biomedisa [101] is a recent platform for biomedical image segmentation that performs interpolation of sparsely pre-segmented slices based on simulation of random walker agents. Recent further development of random walker segmentation includes inonlocal random walker [106], hierarchical implementation for segmenting large volumetric biomedical images [97], and end-to-end learned random walker [94]. In our case all pixels of the labeled superpixels of type a) and b) can serve as seeds. Then, we use these seeds to obtain an assignment of the pixels of all unlabeled superpixels of type c).

For a visual representation of this methodology, Please refer to Figure 4.5.

Furthermore, We treat the instersected superpixel boudaries with another approach, we call Extended Superpixel Labeling (ESL).In this procedure The algorithm then creates a result mask image where each pixel is assigned a label/color based on its superpixel's characteristics. The loop iterates through all superpixels: for superpixels outside the boundary, their corresponding pixels are set to black (0, 0, 0). For superpixels inside the boundary, their corresponding pixels are set to white (255, 255, 255). For boundary superpixels (identified earlier), it further evaluates whether they have more pixels inside or outside the ground truth region. Based on the majority, they are colored white (inside) or black (outside). If the superpixel has an equal number of pixels inside and outside, it calculates the distance from both sides of the boundary using the `cv2.distanceTransform` function. Based on the distance comparison, it colors the superpixel. indeed, there is three differnt cases as below:

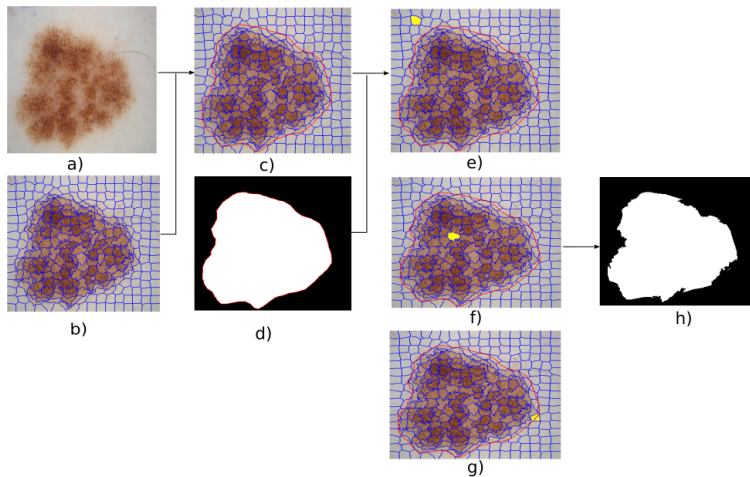


Figure 4.5: generating superpixel labels. a)original image. b)superpixel map. c)intersection of image boundaries with superpixels. d) hard ground truth with related boundaries in red e) superpixel outside the boundary. f) superpixel intersects with the boundary. g) superpixel intersects with the boundary. h) superpixel labeled ground truth.

Case 1: figure 4.6 a) If the number of pixels inside the boundary superpixel is more than the number of pixels outside the boundary superpixels , label 2 (foreground) is assigned to the superpixel.

Case 2: figure 4.6 b) If the number of pixels inside the boundary superpixel is less than the number of pixels outside the boundary superpixels , label 1 (background) is assigned to the superpixel.

Case 3: figure 4.6 c) If the number of pixels inside and outside of the boundary are equal, we will take the following approach: firstly we calculate the distance between each pixel ( $d_1$ ) and label 1 and label 2 ( $d_2$ ) as well. then the label assignment can be done in two below orders :

- 1) sort the list of ( $d_1$ )
- 2) select the first 50% of the sorted list for label 1 and the other 50% for label 2.

OR, we can do the assignment as below:

- 1) sort the list of ( $d_2$ )
- 2) select the first 50% of the sorted list for label 2 and the other 50% for label 1.

After creating new superpixel ground truth, we feed the segmentation network with both original images and related superpixel labeled ground truth. More details about this procedure will be explained in chapter 5.

## 4.2. SUPERPIXEL LABELING

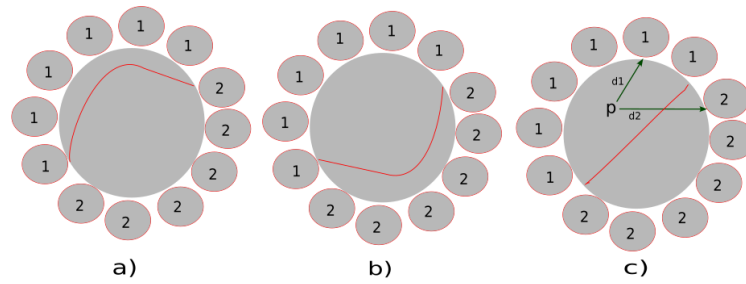


Figure 4.6: a) number of pixels inside the boundary is more than the number of pixels outside the boundary. b) number of pixels outside the boundary is more than the number of pixels inside the boundary. c) number of pixels inside and outside of the boundary are equal.

# 5

## Results

### 5.1 DATASET

Skin cancer stands as the widespread type of cancer in the United States, surpassing 8 billion dollar in yearly care expenses. Timely identification leads to a potential 99% five-year survival rate for its most lethal variant, melanoma. Conversely, a postponed diagnosis significantly reduces this rate to 23% [95].

Recognizing the significance of early identification, extensive efforts have been devoted to enhancing the precision and scope of diagnostic techniques. In the years 2016 and 2017, the ISIC [95], an international alliance responsible for curating the world's most extensive collection of publicly accessible dermoscopic images, hosted the inaugural public evaluations for melanoma detection within dermoscopic images. These evaluations, titled "Skin Lesion Analysis Towards Melanoma Detection," took place at the IEEE International Symposium of Biomedical Imaging (ISBI). The consecutive challenges appealed global engagement, increasing over 900 registrations and more than 350 submissions. At that time, these challenges stood as the most expansive standardized and comparative studies, producing innovative insights and numerous publications. Consequently, other groups have implicitly embraced these challenges as a widely accepted benchmark [95].

In this thesis, we use the 2018 version of ISIC dataset. This challenge took place in 2018 during the Computing and Computer Aided Intervention (MICCAI) conference in Granada, Spain. Alongside substantial expansions in

## 5.1. DATASET

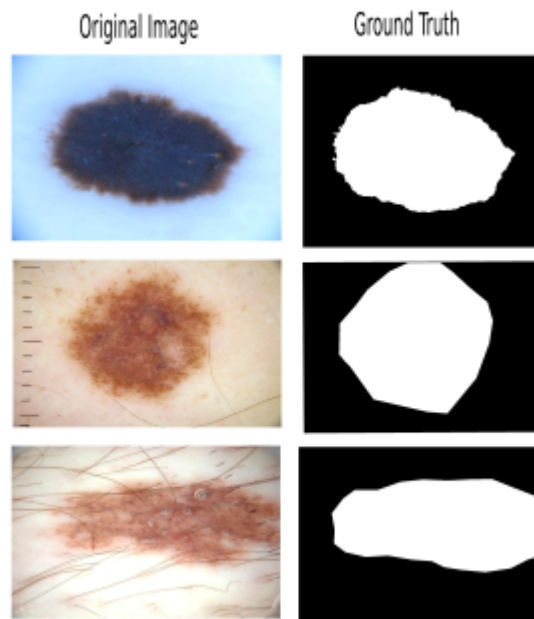


Figure 5.1: Original Image and Related Ground Truth

dataset size and diagnostic categories, significant modifications were made to assessment standards and research methodology. These changes were aimed at providing a more accurate representation of the intricate clinical situations faced in real-world scenarios

The ISIC2018 dataset consists of 2594 original dermatological images along with associated binary label images. This dataset includes actinic keratosis and intraepithelial neoplasia, benign keratosis, basal cell carcinoma, squamous skin diseases such as cell carcinoma, dermatofibroma, melanoma, mole, and vascular lesions[74]. Figure 5.1 shows some images with related ground truths.

The second dataset is chest X-ray dataset. The Chest X-ray (CXR) image is widely used in medical diagnosis due to its affordability, ease of access, and widespread availability. It can detect several issues at once, but it needs confirmation by radiologists. Yet, examining numerous CXRs is burdensome for medical staff, particularly radiologists, leading to time-consuming patient diagnosis procedures.

The suggested models in this thesis were trained and assessed using three specific datasets: Shenzhen[54][99], MC (Montgomery)[54][99], and JSRT (Japanese Society of Radiological Technology)[82].

The Shenzhen dataset, gathered in 2012 by Shenzhen People's Hospital and

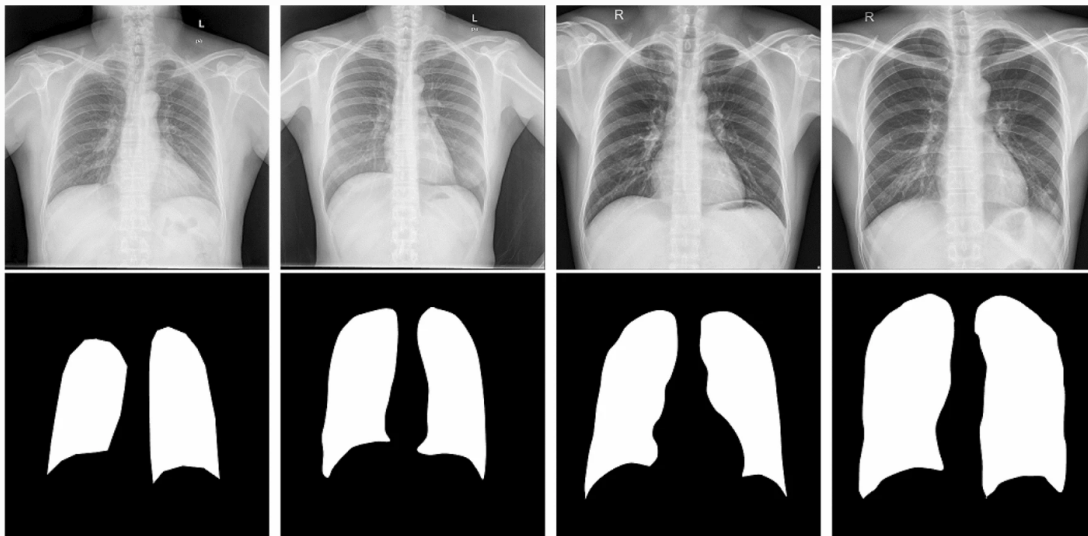


Figure 5.2: Shenzhen dataset example. Top: CXR images; bottom: their binary masks

Guangdong Medical College in China, is a publicly available collection. This dataset comprises 662 CXR images at a resolution of  $3000 \times 3000$ , each accompanied by manually annotated binary masks provided by radiologists. Among these images, 336 depict tuberculosis-affected cases, while 326 show normal conditions.

The MC dataset, created by the Montgomery County Department of Health and Human Services in the USA, comprises 138 CXR images. These images have varying resolutions, either  $4892 \times 4020$  or  $4020 \times 4892$ , and are accompanied by their respective masks. The annotations were overseen by a radiologist. Within this dataset, 58 images display tuberculosis-affected conditions, while 80 images depict normal cases. An example from the MC dataset.

The JSRT dataset was formulated collaboratively by the Japanese Society of Radiological Technology and the Japanese Radiological Society. This dataset comprises 247 CXR images, each with a resolution of  $2048 \times 2048$ . Among these, there are 93 images of normal conditions and 154 images displaying lung nodules, of which 54 are benign nodules and 100 are malignant nodules. The labeling process involved contributions from two human observers and a radiologist. Figure 5.2[54, 99] illustrates examples from the Shenzhen dataset. In this thesis, we use randomly 704 images with related ground truths for the aim of training.

## 5.2 EVALUATION METRICS

Evaluation methods and the effectiveness of model results vary significantly within computer vision across various research domains and practical uses. The area of Medical Image Segmentation (MIS) pertains to the automated detection and labeling of important medical regions such as organs or medical abnormalities (such as cancer or lesions)[60].

Several recent research works have highlighted that deep learning-based models for medical image segmentation have exhibited strong predictive abilities, matching the performance of radiologists. Medical professionals, particularly those in radiology and pathology, are actively working to incorporate these deep learning-driven MIS techniques into their clinical practices as systems for clinical decision support (CDS). These systems can assist in tasks such as diagnosis, treatment planning, risk evaluation, and the streamlining of time-intensive examination procedures. Given their direct influence on diagnostic and therapeutic choices, accurate and dependable assessment of MIS algorithms holds paramount importance [66].

However, over the past few years, a troubling trend has emerged in the scientific publication of MIS studies. This trend involves the deliberate selection of inappropriate metrics in order to showcase remarkably high scores, often approaching 100% . Research has revealed that this skewing of evaluation results stems from various issues, ranging from the incorrect implementation or usage of metrics to the absence of proper hold-out set sampling for trustworthy validation. Consequently, this situation has led to a scenario where numerous clinical research teams are encountering challenges when attempting to apply these models beyond research settings. The utilization of flawed metrics and the absence of standardized evaluation practices within the scientific community for measuring the performance of models in critical health-related procedures pose a significant threat to the quality and dependability of CDS systems [77, 42].

In the area of evaluating supervised segmentation, it is essential to grasp the concepts of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) False Negative. The comprehension of traditional assessment algorithms is structured primarily on these metrics. We provide a straightforward meaning of these abbreviations using Fig 5.3[92], while the definitions of TP, FP, TN, and FN can be located in the works of Taha and Hanbury (2015)[85] as well as Chang et al[32] (2009).



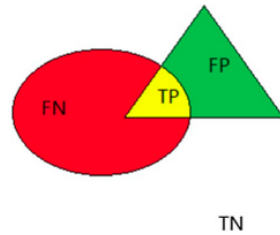


Figure 5.3: The graphical representation of TP, FP, TN and FN

As depicted in Figure 5.3, the elliptical region corresponds to the ground truth, while the triangular area represents the result of the segmentation. TP corresponds to instances where the segmentation result is 1 and the ground truth is also 1, indicated by the color yellow. TN demonstrates situations where the segmentation result is 0 and the ground truth is also 0, represented by white. FP refers to scenarios where the segmentation result is 1, but the ground truth is 0, denoted by the color green. FN signifies cases in which the segmentation result is 0, while the actual data is 1, portrayed in red [92].

Evaluating semantic segmentation is intricate, involving classification accuracy and localization precision evaluation. The objective is to quantify the similarity between predicted and annotated segmentations. In the realm of MIS literature, numerous evaluation metrics have emerged in the past three decades. Yet, only a select few have demonstrated appropriateness and standardization in usage [39].

With the exception of Hausdorff distance, the majority of metrics introduced rely on calculating a confusion matrix for binary segmentation tasks. This matrix includes counts for TP, FP, TN, and FN predictions. Excluding Cohen’s Kappa and Hausdorff distance, the range of values for all the metrics discussed extends from zero (indicating the worst performance) to one (indicating the best performance) [39].

We evaluate the result of our segmentation with dice coefficient, IOU, precision and sensitivity. in the following sub section we will discuss all mentioned metrics in details:

### 5.2.1 DICE COEFFICIENT

In the context of segmentation tasks, the Dice coefficient, also known as the Sørensen–Dice coefficient, is a similarity metric that is commonly used to

## 5.2. EVALUATION METRICS

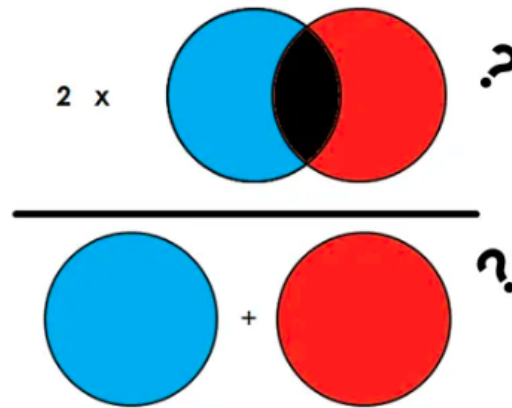


Figure 5.4: Illustration of dice coefficient

evaluate the performance and accuracy of segmentation algorithms. It measures the overlap between the predicted segmented region and the ground truth or reference segmented region.

*"The Dice coefficient is defined as the intersection of two times divided by the sum of pixels, also called F1 score" [74].*

The formula is defined in Equation 5.1. Also the illustration of dice coefficient can be seen accordingly in figure 5.4[41].

$$\text{Dice} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \quad (5.1)$$

The Dice coefficient value ranges from 0 to 1, where 0 indicates no overlap between the predicted and true masks, and 1 indicates a perfect match.

The Dice coefficient provides a balance between precision and recall. It's particularly useful in cases where the objects being segmented are relatively small or when there's a class imbalance between object and background pixels. A higher Dice coefficient indicates a better alignment between the predicted and ground truth masks [41].

There are some Advantages of Dice Coefficient for Medical Image Segmentation which are listed below:

- **Sensitivity to Overlap:** The Dice coefficient is particularly useful when you want to assess the degree of overlap between the segmented regions and the ground truth. It provides a measure of how well the segmentation aligns with the true anatomical structures.
- **Simplicity:** The formula for calculating the Dice coefficient is straightforward and easy to implement. This makes it a practical choice for comparing different segmentation algorithms.

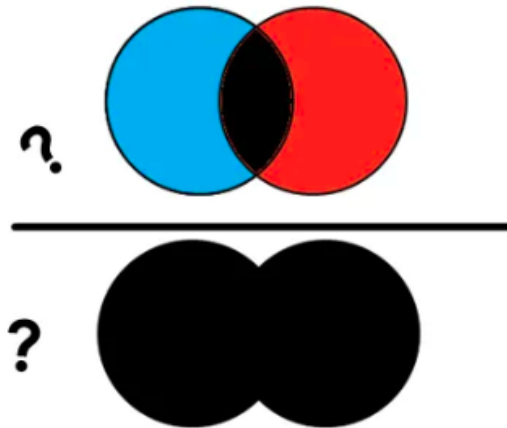


Figure 5.5: illustration of IOU

- **Binary Nature:** The Dice coefficient is well-suited for binary segmentation tasks, which is common in medical image analysis where you often need to identify specific structures or abnormalities.

### 5.2.2 IOU

IOU metric, also referred to as the Jaccard index, holds a significant place as one of the most frequently employed metrics in the realm of semantic segmentation. IOU quantifies the degree of overlap between a predicted segmentation and a ground truth label. It is calculated by dividing the area of overlap between the predicted segmentation and the label by the area of their joint difference. This is defined by Equation 5.2 [74].

*"The IOU indicator, also known as the Jaccard index, is one of the most commonly used metrics in semantic segmentation. Iou is the area of overlap between predicted segmentation and label divided by the difference between predicted segmentation and label Joint area" [74]*

Also the illustration of IOU can be seen accordingly in figure 5.5 [41].

$$\text{IOU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad (5.2)$$

IOU, also known as the Jaccard index, offers several benefits when used as an evaluation metric in medical image segmentation tasks:

## 5.2. EVALUATION METRICS

- **Segmentation Overlap Measurement:** IOU provides a direct measure of how well the segmented region aligns with the ground truth region. It quantifies the spatial agreement between the two, which is crucial in medical applications where accurate localization is essential.
- **Insensitive to Class Imbalance:** IOU is less sensitive to class imbalance than metrics like accuracy. In medical images, where the region of interest (e.g., tumors) is often small compared to the background, IOU can provide a more balanced assessment.
- **Boundary Accuracy:** IOU considers both the area of overlap and the area of non-overlap. This makes it sensitive to boundary accuracy, which is particularly important in medical images where precise localization matters.
- **Thresholding Flexibility:** IOU can be used with thresholded probability maps or binary masks, offering flexibility in handling various types of model outputs.
- **No Single Threshold Dependency:** Unlike some other metrics, IOU doesn't rely on a single threshold value for binary conversion. This makes it suitable for evaluating methods producing different levels of segmentation confidence.
- **Interpretable Range:** The IOU value ranges from 0 to 1, where 1 indicates a perfect match and 0 indicates no overlap. This makes it easy to interpret and compare across different datasets and tasks.
- **Complements Dice Coefficient:** IOU and Dice coefficient are related metrics, and using them together provides a comprehensive understanding of segmentation performance. IOU emphasizes region overlap, while Dice focuses on true positives.
- **Meaningful for Clinical Decision-Making:** IOU directly relates to the spatial agreement between the predicted and true segmentations. Clinically, this alignment can influence decisions, treatment planning, and patient outcomes.
- **Applicability to Multi-Class Segmentation:** IOU can be extended to multi-class segmentation tasks by calculating IOU scores for each class separately.
- **Standardization:** IOU is a widely used metric in the computer vision and medical image analysis communities, allowing for standardized evaluation and comparison across different methods and datasets.
- **Visual Explanation:** IOU values can be visualized as overlays on images, showing where the predicted and true segmentations align or differ, aiding in result interpretation.

### 5.2.3 PRECISION AND SENSITIVITY

Precision and sensitivity (also known as recall or true positive rate) are two important metrics used to evaluate the performance of image segmentation algorithms. These metrics help assess how accurately an algorithm can identify and delineate objects or regions of interest within an image. Let's dive into each of these metrics:

**Precision**(Positive Predictive Value): Precision measures the accuracy of positive predictions made by the segmentation algorithm. It answers the question: "Of all the pixels (or regions) that the algorithm predicted as belonging to the object of interest, how many were actually correct?" Precision is calculated using the following formula 5.3 [74]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (5.3)$$

A high precision indicates that the algorithm is good at avoiding false positives, meaning it accurately identifies the object without including many irrelevant pixels.

**Sensitivity**(Recall or True Positive Rate): Sensitivity, also known as recall or true positive rate, measures how well the algorithm captures all the relevant pixels belonging to the object of interest. It answers the question: "Of all the pixels that truly belong to the object of interest, how many were correctly identified by the algorithm?" Sensitivity is calculated using the below formula 5.4 [74]:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (5.4)$$

All in all, Precision focuses on the accuracy of positive predictions and reducing false positives. High precision means fewer false alarms. Sensitivity focuses on the algorithm's ability to capture all the relevant object pixels and minimizing false negatives. These two metrics are often used together to evaluate the overall performance of an image segmentation algorithm.

## 5.3 EXPERIMENT SETUP AND CONFIGURATION

The experiments in this project were conducted on a personal computer with the following hardware specifications:

**Processor:** 11th Gen Intel® Core™ i5-11400H @ 2.70GHz × 12. **Memory:**

### 5.3. EXPERIMENT SETUP AND CONFIGURATION

15.3 GiB. **Graphics:** NVIDIA Corporation / Mesa Intel® UHD Graphics (TGL GT1). Two NN architectures, namely U-Net and Double U-Net, were employed in this experiment. These networks are widely utilized for image segmentation tasks due to their effectiveness in capturing fine-grained details and spatial information. The NN models were compiled using the following parameters:

- **Optimizer:** The AdamW optimizer with a learning rate of 0.0001 was chosen to optimize the model's parameters during training. AdamW is known for its stability and convergence properties in DL tasks.
- **Loss:** The loss function dice-coef-loss was employed as the optimization objective. This loss function is commonly used in image segmentation tasks to optimize for the Dice coefficient, which measures the similarity between predicted and ground truth masks.
- **Metrics:** Several evaluation metrics, including Dice coefficient, precision, IOU, and sensitivity (recall), were used to monitor the model's performance during training. These metrics provide insights into various aspects of segmentation quality.
- **Run-eagerly:** Eager execution mode was enabled to facilitate debugging and inspection of intermediate results during training.

Also The training process was configured as follows:

- **Batch Size:** The batch size was set to 8. This means that the model processes eight samples (images and their corresponding masks) in each training iteration.
- **Number of Epochs:** The training process was scheduled to run for 50 epochs. An epoch represents one complete pass through the entire training dataset.
- **Training Data Generator:** The train-generator was used to generate batches of training data.
- **Validation Data Generator:** The val-generator was employed to generate batches of validation data, allowing for the evaluation of the model's performance on unseen data.
- **Steps per Epoch:** train-steps-per-epoch and val-steps-per-epoch specify the number of batches to process in each training and validation epoch, respectively.

Furthermore, Callbacks are essential tools in training deep learning models as they enable the monitoring of the training process and the application of various actions based on specific conditions. Here's an explanation of each of the callbacks we have defined in this experiment:

**ModelCheckpoint Callback:** This callback is responsible for saving the model's weights during training:

- `ModelCheckpoint` saves the model weights to a specified file whenever a certain condition is met (e.g., when validation loss decreases).
- `weight-path` is the path where the model weights will be saved.
- `monitor='val-loss'` specifies that the callback will monitor the validation loss.
- `save-best-only=True` ensures that only the weights corresponding to the best validation loss will be saved.
- `mode='min'` indicates that the callback is looking for the minimum validation loss.
- `save-weights-only=True` means that only the model weights will be saved (not the entire model).

**ReduceLROnPlateau Callback:** This callback adjusts the learning rate if the validation loss plateaus:

- `ReduceLROnPlateau` reduces the learning rate when the monitored quantity (validation loss) stops improving.
- `monitor='val-loss'` specifies that the callback will monitor the validation loss.
- `factor=0.5` reduces the learning rate by a factor of 0.5.
- `patience=3` defines the number of epochs with no improvement after which the learning rate will be reduced.
- `verbose=1` provides feedback about the learning rate reductions.
- `mode='min'` indicates that the callback is triggered when validation loss decreases.
- `epsilon=0.0001` sets a threshold for considering a change in validation loss as an improvement.
- `cooldown=2` specifies a cooldown period during which the learning rate won't be changed.
- `min-lr=1e-5` defines the minimum allowed learning rate.

**EarlyStopping Callback:** This callback stops training if a certain condition (e.g., no improvement in validation loss) persists for too long.

- `EarlyStopping` halts training when a monitored quantity (validation loss) doesn't improve.
- `monitor="val-loss"` specifies that the callback will monitor the validation loss.
- `mode="min"` indicates that it's looking for a decrease in validation loss.

## 5.4. QUANTITATIVE AND QUALITATIVE RESULTS

- `patience=10` sets the number of epochs with no improvement after which training will be stopped.

**TensorBoard Callback:** This callback is used for visualizing and monitoring training progress in TensorBoard.

- TensorBoard logs various metrics and visualizations to a specified directory for later analysis.
- `log-dir` is the directory where TensorBoard logs will be saved.
- `histogram-freq=1` logs histogram data for every epoch.
- `write-graph=True` logs the computation graph for visualization in TensorBoard.
- `write-images=True` logs visualizations of layer activations as images.
- `update-freq='epoch'` determines how often the logs are updated (in this case, once per epoch).
- `profile-batch=2` enables profiling of batch execution time.
- `embeddings-freq=1` logs embeddings data for visualization.

Worth mentioning that we allocate 10% of data to the validation set, while the remaining 90% is used used for training. This division allows us to train our machine learning model on one subset and evaluate its performance on the other, helping us assess how well our model generalizes to unseen data.

## 5.4 QUANTITATIVE AND QUALITATIVE RESULTS

As It is demonstrated in table 5.1 , the result of our segmentation algorithm on data set ISIC 2018 is shown as following: two different methods of algorithms as explained in chapter 4 is used in two categories of Superpixel labeling and extended Superpixel Labeling. Then the experiment is carried out on two segmentation network U-Net and Double-UNet. The evaluation metrics are Dice, IOU, Precision and Sensitivity.

also in fig 5.6, 5.7, 5.8 and 5.9, you can see the qualitative result of segmentation for skin and chest dataset respectively.



Labeling Method	Supersixel Alg	Dice	IOU	Precision	Sensitivity
Supersixel Labeling	Quick Shift	0.84	0.75	0.92	0.81
	SLIC	0.85	0.76	0.93	0.81
	FH	0.85	0.76	0.93	0.81
Extended Supersixel Labeling	Quick Shift	0.88	0.81	<b>0.89</b>	0.91
	SLIC	<b>0.89</b>	<b>0.82</b>	<b>0.89</b>	<b>0.92</b>
	FH	0.88	0.81	0.88	0.90
Full pixel		0.87	0.79	0.86	0.92

Table 5.1: Segmenation Results for ISIC2018 lesion skin boundary dataset on U-Net network

Labeling Method	Supersixel Alg	Dice	IOU	Precision	Sensitivity
Supersixel Labeling	Quick Shift	0.89	0.75	0.91	0.86
	SLIC	0.87	0.79	0.92	0.85
	FH	0.86	0.79	0.94	0.83
Extended Supersixel Labeling	Quick Shift	0.91	0.79	<b>0.93</b>	0.90
	SLIC	0.88	0.81	0.91	0.90
	FH	0.89	0.80	0.92	0.91
Full pixel		<b>0.93</b>	<b>0.87</b>	0.92	<b>0.95</b>

Table 5.2: Segmenation Results for ISIC2018 lesion skin boundary dataset on Double U-net

Labeling Method	Supersixel Alg	Dice	IOU	Precision	Sensitivity
Supersixel Labeling	SEEDS	0.8098	0.7055	<b>0.8637</b>	0.8187
	SLIC	0.8406	0.7494	0.8587	0.8743
Extended Supersixel Labeling	SEEDS	<b>0.8818</b>	<b>0.8042</b>	0.8532	<b>0.9432</b>
	SLIC	0.8568	0.7710	0.8348	0.9233
Supersixel Labeling with Random Walk	SEEDS	0.8356	0.7374	0.7871	0.9347
	SLIC	0.8594	0.7730	0.8289	0.9312
Full pixel		0.8524	0.7623	0.8199	0.9312

Table 5.3: Segmenation Results for ISIC2018 lesion skin boundary dataset on U-net and SEEDS Alg

#### 5.4. QUANTITATIVE AND QUALITATIVE RESULTS

Labeling Method	Superpixel Alg	Dice	IOU	Precision	Sensitivity
Superpixel Labeling	SEEDS	0.7625	0.6180	0.6585	0.9229
	SLIC	0.8071	0.6812	0.7240	0.9245
Extended Superpixel Labeling	SEEDS	0.8753	0.7922	0.8263	<b>0.9611</b>
	SLIC	0.8798	0.7981	0.8354	0.9579
Superpixel Labeling with Random Walk	SEEDS	0.8580	0.7680	0.8163	0.9460
	SLIC	<b>0.8814</b>	<b>0.8031</b>	<b>0.8466</b>	0.9503
Full pixel		0.8870	0.8109	0.8448	0.9620

Table 5.4: Segmenation Results for ISIC2018 lesion skin boundary dataset on DuAT

Labeling Method	Superpixel Alg	Dice	IOU	Precision	Sensitivity
Superpixel Labeling	SEEDS	0.8254	0.6575	0.9164	0.7635
	SLIC	0.7485	0.5987	0.8180	0.6988
Extended Superpixel Labeling	SEEDS	<b>0.9568</b>	<b>0.9180</b>	0.9509	<b>0.9642</b>
	SLIC	0.9548	0.9143	<b>0.9521</b>	0.9590
Superpixel Labeling with Random Walk	SEEDS	0.9378	0.8851	0.9238	0.9576
	SLIC	0.9476	0.9018	0.9370	0.9613
Full pixel		0.9537	0.9122	0.9339	0.9758

Table 5.5: Segmenation Results for Lung X-Ray on DuAT

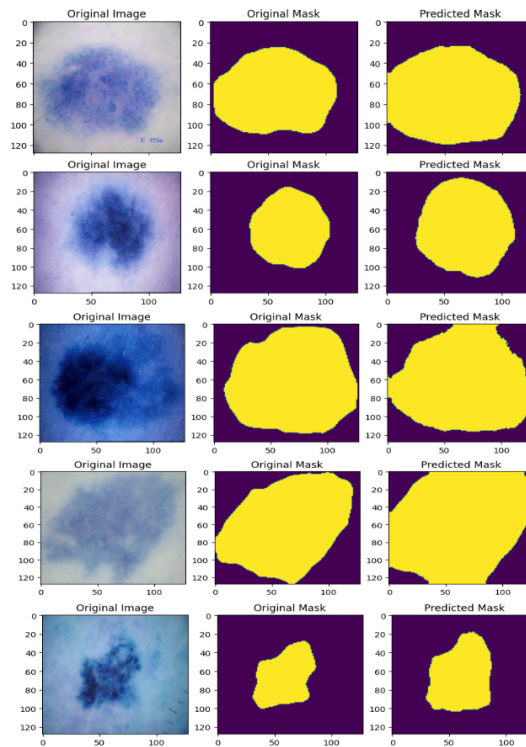


Figure 5.6: Segmentation result with unet network for lesion skin dataset

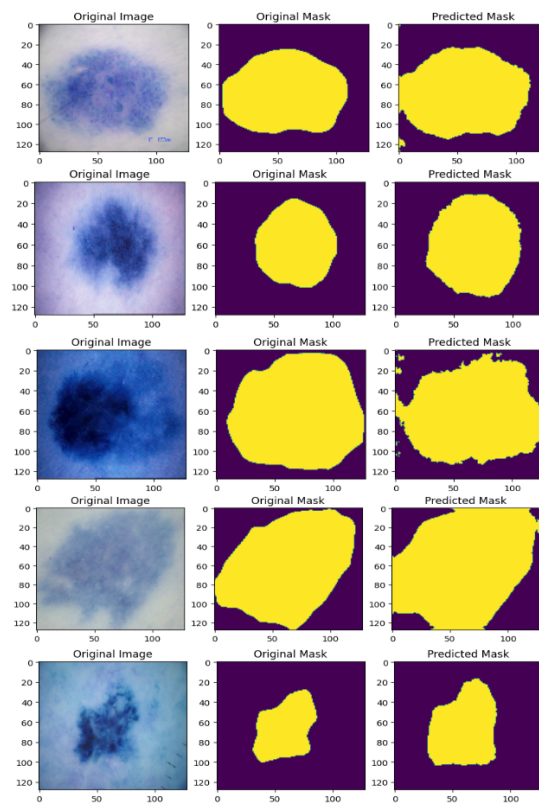


Figure 5.7: Segmentation result with Double unet network for lesion skin dataset

## 5.4. QUANTITATIVE AND QUALITATIVE RESULTS

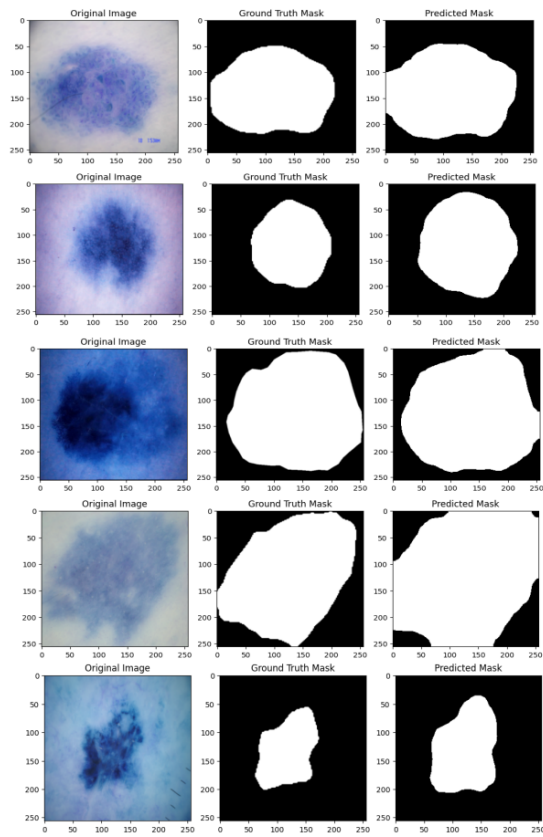


Figure 5.8: Segmentation result with DuAT network for lesion skin dataset

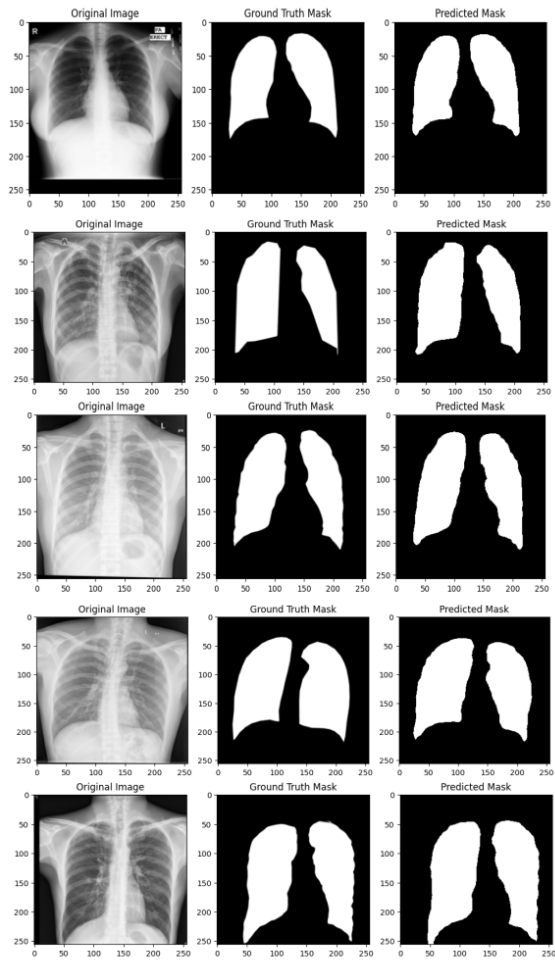


Figure 5.9: Segmentation result with DuAT network for X-ray chest dataset



# 6

## Conclusions and Future Works

As explained in detail in previous chapters, we did the Superpixel labeling with three different techniques including Superpixel labeling with three class labels, ESL, and Superpixel labeling with random walk. Also, the simulation is done on three different deep-learning neural networks consisting of U-net, Double U-net, and DuAT. Furthermore, to do the comparison between different Superpixel algorithms, the ground truths were labeled with four different Superpixel methods: SLIC, QS, FH, and SEEDS.

As shown in Table 5.1, with the U-net network and lesion skin data set, the best results concerning the Dice coefficient are achieved with a combination of ESL and SLIC algorithms. Regarding the IOU metrics, the more accurate results were achieved with the same mentioned combination. this also implies the Precision and sensitivity metrics. mention that the QS algorithm results were comparable with SLIC.

Regarding the second table 5.2: simulation of Double U-net on skin lesion data set, the full pixel guideline had a better performance in Dice, IOU, and sensitivity as well. but the precision was slightly more than pixel-wise labeling with a 0.1 percent in difference.

Refer to the second table 5.3: the simulation of U-net on skin lesion data set; the accuracy of 0.8818 and 0.8042 for Dice and IOU respectively, are noticeable with SEEDS and ESL techniques. Surprisingly, the precision accuracy was best when using the Superpixel labeling with the SEEDS algorithm. However, the sensitivity was higher (0.9432) once working again with the ESL and SEEDS algorithm.

As depicted in Table 5.4, DuAT works pretty well in the random walk technique with the SLIC algorithm. Dice, IOU, and precision results are as follows: 0.8814, 0.8031 and 0.8466.

The best result for the chest X-ray images was achieved with DuAT, ESL, and SEEDS algorithms.

In this research, we select our Deep Neural Networks according to the most recent algorithms and also their popularity in medical image segmentation. In future works, the simulation can be done on the other NN networks which might be more optimized for selected data sets. Also, the whole simulation can be done on more challenging medical data sets such as colonoscopy, CVC clinic, and Polyp where the segmentation is more challenging due to the skin color and also boundary intersection.



# References

- [1] Boldrini L et al. "Deep Learning: A Review for the Radiation Oncologist." In: *Frontiers in Oncology*. 2019, p. 9.
- [2] Bryce TJ et al. "Artificial neural network model of survival in patients treated with irradiation with and without concurrent chemotherapy for advanced carcinoma of the head and neck." In: *Int J Radiat Oncol Biol Phys*. 1998, 41:339–45.
- [3] Chen SF et al. "A neural network model to predict lung radiation-induced pneumonitis." In: *Med Phys*. 2007, 34:3420–7.
- [4] Çiçek et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer International Publishing. 2016.
- [5] D. Stutz et al. "Understanding Convolutional Neural Networks." In: 2014.
- [6] D. Yu et al. In: *Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition*. 2013, pp. 3366–3370.
- [7] Debesh Jha et al. "DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation". In: *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*. 2020, pp. 558–564.
- [8] Gulliford SL et al. "Use of artificial neural networks to predict biological outcomes for patients receiving radical radiotherapy of the prostate." In: *Radiother Oncol*. 2004, 71:3–12.
- [9] Hesamian MH et al. "Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges." In: *J Digit Imaging*. 2019, 32:582–96.
- [10] LeCun Y et al. "Deep learning". In: *Nature*. 2015, 521:436–44.

## REFERENCES

- [11] Lei Y et al. "CT prostate segmentation based on synthetic MRI-aided deep attention fully convolution network." In: *Med Phys*. 2019, doi: 10.1002/mp.13933.
- [12] Lei Y et al. "CT prostate segmentation based on synthetic MRI-aided deep attention fully convolution network." In: *Med Phys*. 2020, 47:530–40.
- [13] Milletari et al. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." In: *3D Vision (3DV), 2016 Fourth International Conference on. IEEE*. 2016.
- [14] O. Russakovsky et al. In: *ImageNet Large Scale Visual Recognition Challenge*. 2012.
- [15] Ochi T et al. "Survival prediction using artificial neural networks in patients with uterine cervical cancer treated by radiation therapy alone." In: *Radiology*. 2000, 217:142–.
- [16] Ronneberger et al. "U-net: Convolutional networks for biomedical image segmentation." In: *International Conference on Medical image computing and computer-assisted intervention. Springer, Cham*. 2015.
- [17] Saad ALBAWI et al. "Understanding of a convolutional neural network ." In: *International Conference on Engineering and Technology (ICET)*. 2017.
- [18] Shelhamer E et al. "Fully Convolutional Networks for Semantic Segmentation." In: *IEEE Trans Pattern Anal Mach Intell*. 2017, 39:640–51.
- [19] Su M et al. "An artificial neural network for predicting the incidence of radiation pneumonitis." In: *Med Phys*. 2005, 32:318–25.
- [20] Tomatis S et al. "Late rectal bleeding after 3D-CRT for prostate cancer: development of a neural-network-based predictive model." In: *Phys Med Biol*. 2012, 57:1399–412.
- [21] Y. Guo et al. "Understanding Convolutional Neural Networks." In: *Accepted Manuscript Deep learning for visual understanding.review To appear in : Neurocomputing*, 2015.
- [22] Yabo Fu et al. "A review of deep learning based methods for medical image multi-organ segmentation." In: *Physica Medica* 85. 2021, pp. 107–122.
- [23] Zhou T et al. "A review: Deep learning for medical image segmentation using multi-modality fusion." In: *Array*. 2019, 3–4:100004.

- [24] Zongwei Zhou et al. "UNet++: A Nested U-Net Architecture for Medical Image Segmentation." In: *Deep Learn Med Image Anal Multimodal Learn Clin Decis Support*. 2018, 11045:3–11.
- [25] A. Levinshtein et al. "Turbopixels: Fast superpixels using geometric flows." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 2009, 13(6):583–598.
- [26] A. Vedaldi et al. "Quick shift and kernel methods for mode seeking." In: *In European Conference on Computer Vision (ECCV)*. 2008.
- [27] Akkus Z et al. "Deep learning for brain MRI segmentation: state of the art and future directions." In: *J Digit Imaging*. 2017, 30(4):449–59.
- [28] Alastair Moore et al. "Superpixel Lattices". In: *IEEE Computer Vision and Pattern Recognition*. 2008.
- [29] Alexandr A. Kalinin et al. "Medical Image Segmentation Using Deep Neural Networks with Pretrained Encoders." In: *Deep Learning Applications*. 2020.
- [30] B.H. Menze et al. "The multimodal brain tumor image segmentation benchmark (brats)." In: *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*. 2015, IEEE Transactions on Medical Imaging, vol. 34, no. 10, pp.1993–2024.
- [31] Bethany H et al. "PSEUDO-LABEL REFINEMENT USING SUPERPIXELS FOR SEMI-SUPERVISED BRAIN TUMOUR SEGMENTATION". In: *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*. 2022, DOI: 10.1109/ISBI52829.2022.9761681.
- [32] Chang H-H et al. "Performance measure characterization for evaluating neuroimage segmentation algorithms". In: *Neuroimage*. 2009, 47(1):122–135.
- [33] Chen LC et al. "DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs." In: *IEEE Trans Pattern Anal Mach Intell*. 2017, 40(4):834–48.
- [34] Chen S et al. "Dual-force convolutional neural networks for accurate brain tumor segmentation." In: *Pattern Recogn*. 2019, 88:90–100.

## REFERENCES

- [35] Ciresan et al. "Deep neural networks segment neuronal membranes in electron microscopy images." In: *Advances in neural information processing systems*. 2012, pp.2852–2860.
- [36] D. Jha et al. "Kvasir-seg: A segmented polyp dataset". In: *International Conference on Multimedia Modeling (MMM)*. 2020, pp. 451–462.
- [37] D. Jha et al. "Resunet++: An advanced architecture for medical image segmentation". In: *in Proceeding of IEEE International Symposium on Multimedia (ISM)*. 2019, pp. 225–2255.
- [38] Dhatri Raval et al. "A Comprehensive assessment of Convolutional Neural Networks for skin and oral cancer detection using medical images." In: *Healthcare Analytics 3* 100199. 2023.
- [39] Dominik Müller et al. "TOWARDS A GUIDELINE FOR EVALUATION METRICS IN MEDICAL IMAGE SEGMENTATION". In: 2022.
- [40] Dunlu Peng et al. "LCP-Net: A local context-perception deep neural network for medical image segmentation." In: *Expert Systems with Applications*. 2021, p. 114234.
- [41] Ekin Tiu et al. "Metrics to Evaluate your Semantic Segmentation Model". In: *Towards Data Science*. 2019.
- [42] El Naqa IM et al. "Lessons learned in transitioning to AI in the medical imaging of COVID-19." In: *J Med Imaging*. 2021, 8(S1):010902.
- [43] Enze Xie et al. "Segformer: Simple and efficient design for semantic segmentation with transformers". In: *Advances in Neural Information Processing Systems*. 2021, p. 34.
- [44] Feilong Tang et al. "DuAT: Dual-Aggregation Transformer Network for Medical Image Segmentation". In: *Computer Vision and Pattern Recognition*. 2022.
- [45] Felzenszwalb et al. "Efficient graph-based image segmentation". In: *International Journal of Computer Vision*. 2004.
- [46] Girshick R. et al. "feature hierarchies for accurate object detection and semantic segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.
- [47] Hariharan B et al. "Hypercolumns for object segmentation and fine-grained localization." In: *arXiv:1411.5752 [cs.CV]*. 2014.

- [48] Havaei M et al. "Brain tumor segmentation with deep neural networks." In: *Med Image Anal.* 2017, 35:18–31.
- [49] He et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016.
- [50] Hu K et al. "Brain tumor segmentation using multi-cascaded convolutional neural networks and conditional random field." In: *IEEE Access.* 2019, 7:92615–29.
- [51] Huang et al. "Densely connected convolutional networks." In: *arXiv preprint arXiv:1608.06993.* 2016.
- [52] J. Bernal et al. "WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians." In: *Deep Learn Med Image Anal Multimodal Learn Clin Decis Support.* 2015, Med Imag Graph, vol43, pp.99–111.
- [53] J. Hu et al. "Squeeze-and-excitation networks". In: *computer vision and pattern recognition (CVPR).* 2018, pp. 7132–7141.
- [54] Jaeger et al. "Two public chest x-ray datasets for computer-aided screening of pulmonary diseases". In: *Quantit Imaging Med Surg.* 2014, 4(6):475–7.
- [55] Jianbo Shi et al. "Normalized cuts and image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI).* 2000, 22(8):888–905.
- [56] Krizhevsky et al. "ImageNet classification with deep convolutional neural networks." In: *Advances in neural information processing systems.* 2012.
- [57] Laith Alzubaidi1 et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." In: *Journal of Big Data.* 2021, 8:53.
- [58] Lei Y et al. "Echocardiographic Image Multi-Structure Segmentation using Cardiac-SegNet." In: *Med Phys.* 2021, doi: 10.1002/mp.14818.
- [59] Leo J. Grady et al. "Random Walks for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 2006, pp. 1768–1783.
- [60] Litjens G et al. "A survey on deep learning in medical image analysis." In: *Med Image Anal.* 2017, pp. 60–88.

## REFERENCES

- [61] Long J et al. "Fully convolutional networks for semantic segmentation." In: *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.
- [62] Luc Vincent et al. "Watersheds in digital spaces: An efficient algorithm based on immersion simulations." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1991, 13(6):583–598.
- [63] Md Zahangir Alom et al. "Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation." In: *Computer Vision and Pattern Recognition*. 2018.
- [64] Milletari F et al. "Fully convolutional neural networks for volumetric medical image segmentation." In: *In: 2016 fourth international conference on 3D vision (3DV)*. 2016, IEEE, p. 565–71.
- [65] Moeskops P et al. "Automatic segmentation of MR brain images with a convolutional neural network." In: *IEEE Trans Med Imaging*. 2016, 35(5):1252–61.
- [66] Müller D et al. "Robust chest CT image segmentation of COVID-19 lung infection based on limited data." In: *Informatics Med Unlocked*. 2021.
- [67] Nabil Ibtehaz et al. "MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation." In: *Neural Networks*. 2020, Pages 74–87.
- [68] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15. 2014, p. 1929–1958.
- [69] Olaf Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. 2015, pp 234–241.
- [70] Olaf Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. 2015, pp 234–241.
- [71] Pedro Felzenszwalb et al. "Efficient graph-based image segmentation." In: *International Journal of Computer Vision (IJCV)*. 2004, 59(2):167–181.

- [72] Pella A et al. "Use of machine learning methods for prediction of acute toxicity in organs at risk following prostate radiotherapy". In: *Med Phys*. 2011, 38:2859–67.
- [73] Pereira S et al. "Brain tumor segmentation using convolutional neural networks in MRI images." In: *IEEE Trans Med Imaging*. 2016, 35(5):1240–51.
- [74] Qiaoer Zhou et al. "Superpixel-Oriented Label Distribution Learning for Skin Lesion Segmentation". In: *Diagnostics*. 2022.
- [75] Qingwu Shi et al. "Multichannel convolutional neural network-based fuzzy active contour model for medical image segmentation." In: *Evolving Systems volume 13*. 2022, pp. 535–549.
- [76] Radhakrishna Achanta et al. "SLIC Superpixels Compared to State-of-the-art Superpixel Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2012, Volume: 34.
- [77] Renard F et al. "Variability and reproducibility in deep learning for medical image segmentation." In: *Sci Rep*. 2022, 10(1):1–16.
- [78] Reza Azad et al. "Bi-Directional ConvLSTM U-Net with Densely Connected Convolutions". In: *Image and Video Processing*. 2019.
- [79] Ronneberger et al. "Convolutional networks for biomedical image segmentation." In: *International conference on medical image computing and computer-assisted intervention*. 2015, Springer, p. 234–41.
- [80] Sabour et al. "Dynamic routing between capsules." In: *Advances in Neural Information Processing Systems*. 2017.
- [81] Seyedhosseini M. et al. "Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks." In: *Computer Vision (ICCV), IEEE International Conference*. 2013, pp. 2168–2175.
- [82] Shiraishi J et al. "Development of a digital image database for chest radiographs with and without a lung nodule". In: *Amer J Roentgenol*. 2000, 174(1):71–4.
- [83] Simonyan et al. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv: 1409.1556*. 2014.
- [84] Szegedy et al. "Going deeper with convolutions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

## REFERENCES

- [85] Taha AA et al. "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool." In: *BMC Med Imaging*. 2015, <https://doi.org/10.1186/s128015-0068-x>.
- [86] Vinod Nair et al. "Rectified Linear Units Improve Restricted Boltzmann Machines." In: *Department of Computer Science, University of Toronto, Toronto, ON M5S 2G4, Canada*. 2010.
- [87] Vivek Kwatra et al. "Graphcut textures: Image and video synthesis using graph cuts." In: *Transactions on Graphics, SIGGRAPH*. 2003, 22(3):277–286.
- [88] Wadhwa A et al. "A review on brain tumor segmentation of MRI images." In: *Magn Reson Imaging*. 2019, 61:247–59.
- [89] Yan Q et al. "COVID-19 chest CT image segmentation—a deep convolutional neural network solution." In: *arXiv preprint arXiv:2004.10987*. 2020.
- [90] Yang X et al. "Automated segmentation of the parotid gland based on atlas registration and machine learning: a longitudinal MRI study in head-and-neck radiation therapy." In: *International Journal of Radiation Oncology\* Biology\* Physics*. 2014, 90:1225–33.
- [91] Zewen Li et al. "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects". In: *IEEE Transactions on Neural Networks and Learning Systems*. 2022, Volume: 33, Issue: 12.
- [92] Zhaobin Wang et al. "Image segmentation evaluation: a survey of methods". In: *Artificial Intelligence Review*. 2020, 53:5637–5674.
- [93] Michael Van den Bergh et al. "SEEDS: Superpixels Extracted via Energy-Driven Sampling". In: *Computer Vision – ECCV*. 2012, pp 13–26.
- [94] Lorenzo Cerrone, Alexander Zeilmann, and Fred A. Hamprecht. "End-To-End Learned Random Walker for Seeded Image Segmentation". In: *IEEE Conference on Computer Vis. and Pattern Rec. (CVPR)*. 2019, pp. 12559–12568.
- [95] N et al. Codella. "Skin Lesion Analysis Toward Melanoma Detection 2018". In: *A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)*. 2019, arXiv 2019, arXiv:1902.03368.



- [96] D. Comaniciu and P et al. "Mean shift: a robust approach toward feature space analysis." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, 24(5):603–619.
- [97] Dominik Drees, Florian Eilers, and Xiaoyi Jiang. "Hierarchical Random Walker Segmentation for Large Volumetric Biomedical Images". In: *IEEE Transactions on Image Processing*. 2022, pp. 4431–4446.
- [98] Beam AL Kohane IS. "Translating Artificial Intelligence Into Clinical Care." In: *Journal of the American Medical Association*. 2016, 316:2368–9.
- [99] Karargyris A et al. Jaeger S. "Automatic tuberculosis screening using chest radiographs". In: *IEEE Trans Med Imaging*. 2014, 33(2):233–45.
- [100] Ajitesh Kumar. In: *Different types of CNN architectures explained: Examples, Data Anal.* 2022.
- [101] Philipp D. Lösel and Thomas van de Kamp et al. "Introducing Biomedisa as an open-source onlineplatform for biomedical image segmentation". In: *Nature Communications*. 2020, p. 5577.
- [102] et al. O. Veksler. "Superpixels and supervoxels in an energy optimization framework". In: *In European Conference on Computer Vision (ECCV)*. 2010.
- [103] Hiba Ramadan, Chaymae Lachqar, and Hamid Tairi. "A survey of recent interactive image segmentation methods". In: *Computational Visual Media*. 2020, pp. 355–384.
- [104] R. E. Turner. "Lecture 14 : Convolutional neural networks for computer vision." In: 2014.
- [105] A et al. Vedaldi. "Quick shift and kernel methods for mode seeking". In: *European Conference on Computer Vision*. 2008.
- [106] Hui Wang and Jianbing Shen et Al. "Adaptive Nonlocal Random Walks for Image Superpixel Segmentation". In: *IEEE Transactions on Circuits Systems and Video Technology*. 2020, pp. 822–834.
- [107] Zhaobin Wang et al. "Review of Random Walk in Image Processing". In: *Archives of Computational Methods in Engineering*. 2019, pp. 17–34.
- [108] J. Wu. In: *Introduction to Convolutional Neural Networks*. 2016, pp. 1–28.



# Acknowledgments

I would like to extend my heartfelt gratitude to the individuals and institutions who have supported me throughout my academic journey. Without their guidance, encouragement, and assistance, this thesis would not have been possible.

I am deeply thankful to my academic advisors, professor xiaoyi and professor Alberto Pretto, for their exceptional mentorship, insightful feedback, and constant support. Their expertise and dedication played a pivotal role in shaping the direction of this research.

I owe a profound debt of gratitude to my beloved partner, Toomaj, for his unwavering support, patience, and understanding. His encouragement and belief in me have been my constant motivation.

I am grateful to my parents, for their boundless love, sacrifices, and continuous encouragement. Their guidance and values have been the foundation of my journey.

My sincere appreciation goes to my friends and colleagues who provided valuable insights, engaging discussions, and a collaborative atmosphere that enriched my research.

I would also like to thank university of Padova and university of Munster for providing the resources and facilities that facilitated my research. To the participants of my study, who generously shared their time and insights, I extend my gratitude. Your contributions have enriched the depth of my research.

Each person mentioned has played a significant role in shaping this thesis, and I am honored to have been supported by such remarkable individuals and institutions.