Universidade do Minho
Escola de Engenharia

Mariana Fontainhas Rodrigues

Brain Semantic Segmentation: A Deep Learning approach in Human and Rat MRI studies

Master's Degree Biomedical Engineering Dissertation
Field of Medical informatics

Dissertation supervised by:
Victor Alves
Ricardo Magalhães

06 November 2018

06 November 2018

DECLARAÇÃO

Nome: Mariana Fontainhas Rodrigues

Endereço eletrónico: marianafontainhas@gmail.com

Cartão de Cidadão: 14733567

Título da dissertação: Brain Semantic Segmentation: A Deep Learning approach in Human and Rat MRI studies

Orientadores: Victor Manuel Rodrigues Alves, Ricardo José da Silva Magalhães

Ano de Conclusão: 2018

Designação do Mestrado: Mestrado Integrado em Engenharia Biomédica

Área de Especialização: Informática Médica

Escola de Engenharia

DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA DISSERTAÇÃO

Universidade do Minho, _____/_____/_______

Assinatura: ____

06 November 2018

Acknowledgments

I want first to thank my supervisors Ricardo Magalhães and Victor Alves, who guided me throughout this experience.

I want to express my appreciation to Victor Alves, who I am truly grateful for his guidance. He was essential during the all process of the dissertation findings, discussions and writing. He was always available, motivated and ready to meet me and guide me through this experience. His knowledge in the deep learning field was essential. Thank you so much for all the time and all the Wednesday's mornings that we sit and chat about my dissertation.

Ricardo was a supervisor and a friend who let me to be in touch with the PHD students for a month in Paris. September of 2017 was a month that inspired me to work even harder next to a very special team who embraced me. I am grateful for meeting the all the researchers and the installations and MRI scanners (3T, 7T and 11.7T) in NeuroSpin and all the collaborators in Paris. I want also to thank again Ricardo and his team in ICVS lab for all the knowledge, kindness and friendship! Working in ICVS lab in neuroimaging and meeting all the portuguese team contributed to my grow as a professional and as a person. I've learned more than I could ever imagine with all of you. Thank you for integrating me in your team!

This work is part of the SIGMA project with the reference FCT-ANR/NEU-OSD/0258/2012, co-financed by the French public funding agency ANR (Agence Nationale pour la Recherche, APP Blanc International II 2012), the Portuguese FCT (Fundação para a Ciência e Tecnologia) and the Portuguese North Regional Operational Program (ON.2 – O Novo Norte) under the National Strategic Reference Framework (QREN), through the European Regional Development Fund (FEDER) as well as the Projecto Estratégico co-funded by FCT (PEst-C/SAU/LA0026-/2013) and the European Regional Development Fund COMPETE (FCOMP-01-0124-FEDER-037298).

I gratefully acknowledge the support of the NVIDIA Corporation with their donation of a Quadro P6000 board used in this research. This work was also supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT within the Project Scope: UID/CEC/00319/2013.

I want also to thank my family. Specially to my devoted parents, big brother, which always supported me no matter what, and that I know that I can always count on. For all your love, thank you.

I couldn’t forget to thank my friends. University years would never be the same without my close friends. Thanks for sharing with me memorable times like “Enterro da Gata” parties, lunches and dinners, car rides, walks, laughs and happy tears. You will always be in my heart. Grateful from the bottom of my heart, for everything.

Title

Brain Semantic Segmentation: A DL approach in Human and Rat in MRI studies

Abstract

Magnetic Resonance Imaging (MRI) provides information about anatomy and pathology. This type of technique is the most popular used for the study of rat and human brain. Classifying voxels according to the presence of relevant anatomic features is an important step in the pre-processing of the data. A precise delineation and automatic segmentation of the brain structures is required in preclinical rodent imaging field and can substitute the manual segmentation where time consuming or human-error problems can occur. Current solutions are based on traditional segmentation algorithms that raise accuracy issues and generally need human intervention during or after the segmentation process.

In the humans' field, most of the tools created in DL (deep learning) are used in tumour or lesion segmentation. Brain segmentation tissues are not as explored as oncology problems and lesions complications. In the rats' field, there are no segmentation studies in DL. It was decided to use a DL approach in Rats to solve some of the old techniques' problems.

This dissertation will present an approach on semantic segmentation of white matter and gray matter in Human's images, evaluate the algorithm's performance with outliers. It will also present an FCN (fully convolutional network) solution for on semantic segmentation using rat's and human's MRI of anatomical features. A two-dimensional convolution (slice-by-slice) approach and a three-dimensional (volume) convolutions approach were evaluated. At the end, the results found, using FCN U-NET in rats' MRI for a 2D convolutions approach, DSC were 94.65 % for WM, 91.03% for GM and 76.89 % for cerebrospinal fluid. Using the 3D convolutions approach, the results using DSC found are 93.81 % for WM, 89.69 % for GM and 74.68 % for cerebrospinal fluid. The results using humans' MRI using DSC were 91.59% for WM and 84,58% for GM.

Resumo

Imagens de ressonância magnética providenciam informação acerca da anatomia humana e possíveis patologias existentes. Este é o tipo de técnica mais popular entre os estudos na área da neurociência, tanto em humanos como em roedores. A classificação de voxéis de acordo com a presença de informação anatómica relevante é um importante passo no pré-processamento de dados na comunidade científica na área da neurociência. Uma delineação precisa das várias estruturas do cérebro humano ou roedor é uma das requisições para a maioria dos estudos clínicos de imagens de ressonância magnética. A segmentação automática através de inteligência artificial pode vir a substituir ferramentas ou algoritmos semiautomáticos já existentes ou substituir também a segmentação manual que se trata de um processo muito demorado que está ligado a erro-humano.

O avanço tecnológico provocou um estudo mais aprofundado no Deep Learning (DL) a partir de 2012, provando que estas técnicas de inteligência artificial estão a revelar-se melhores do que o que já existe na área médica.

Dos estudos com ressonância magnética em humanos, a maioria das ferramentas criadas que utilizam DL são usadas na segmentação de tumores ou lesões cerebrais. A segmentação de tecidos cerebrais não está tão explorada como problemas oncológicos ou lesões cerebrais. Dos estudos com ressonância magnética em roedores, não existem ferramentas que utilizam as técnicas de DL. Tendo em conta que as técnicas de segmentação que já existem ainda têm muitas complicações e erros, foi decidido tentar uma abordagem de DL em Roedores, também.

Esta dissertação irá apresentar uma abordagem de segmentação semântica de massa branca e massa cinzenta utilizando técnicas de DL em humanos. Irá também verificar a capacidade de generalização com casos de pacientes idosos. Irá ser apresentado uma técnica de DL nas imagens de ressonância magnética em roedores para a segmentação semântica de massa branca, massa cinzenta e líquido cérebroespinal. No final irá ser comparado as técnicas entre as duas espécies e também entre a utilização de convoluções com duas dimensões e de convoluções com três dimensões nos roedores. No final, os resultados encontrados utilizando uma FCN em Ratazanas numa abordagem 2D, os valores de DSC foram 94,65 % para massa branca, 91.03% para massa cinzenta e 76.89 % para o líquido cérebroespinal. Na abordagem

3D ,os valores de DSC encontrados foram 93.81 % % para massa branca, 89.69 % para massa cinzenta e 74.68 % para o líquido cérebroespinal. Os resultados utilizando as imagens humanas, foram 91.59% para massa branca e 84,58% para massa cinzenta.

Index

ACKNOWLEDGMENTS IV

ABSTRACT VII

RESUMO VIII

INDEX X

FIGURES LIST XIV

TABLES LIST XIX

NOTATION AND TERMINOLOGY XX

ACRONYMS XXI

CHAPTER 1- INTRODUCTION 22

1.1 Clinical investigation using Magnetic Resonance 22

1.2 Computer vision 23

1.3 Machine Learning and Deep Learning 24

1.4 DL in Medical Imaging 25

1.5 Context and Motivation 25

1.6 Related Work - Humans 26

1.7 Related Work - Rats 28

1.8 Research Questions 28

1.9 Document's Organization 29

CHAPTER 2- TECHNOLOGIES AND CONCEPTS 31

2.1 MRI 31

2.2 Deep Learning 32

2.2.1 Artificial Neural Networks 32

2.2.2 Model Optimization 36

2.2.3 CNN Building Blocks 38

2.2.4 DL Tasks 39

2.2.5 Architectures structure 39

2.3 Pre-processing methods 44

2.3.1 Linear Normalization 44

2.3.2 Histogram Equalization 44

2.3.3 Select pre-processing tools 45

2.4 Evaluation metrics 46

2.5 Visualizing the Learning Process 48

2.5.1 Feature Maps 49

2.5.2 Tensorboard 49

CHAPTER 3- RATS EXPERIMENT 51

3.1 Rats Dataset 51

3.2 Rat Brain Semantic Segmentation – First Approach 52

3.3 Rat Semantic Segmentation – Second approach 54

3.3.1 Rat Brain Extraction 54

3.3.2 Rat Semantic Segmentation after BE 55

3.4 Voxels' quantification using Deep learning regression 56

CHAPTER 4- HUMANS EXPERIMENT 59

4.1 Humans Dataset 59

4.2 Human Brain Extraction – First approach 62

4.3 Human Brain Extraction – Second approach 64

4.3.1 Classification DL Model Brain and No-Brain Axial Slice 64

4.3.2 Brain Extraction Segmentation using slices with brain tissue 67

4.4 Human Brain Semantic Segmentation 67

4.4.1 WGM segmentation 68

4.4.2 WGMSC segmentation 68

4.5 Outlier influence in the results 72

CHAPTER 5- DISCUSSION AND CONCLUSION 76

5.1 Discussion on Rat's experiment 78

5.1.1 BE and WGMCSF analysis 78

5.2 Discussion on Human's experiment 79

5.2.1 BE and WGM analysis 79

5.2.1.1 WGMSC analysis 80

5.2.2 Outliers' influence analysis 80

5.3 Humans and Rats DL model performance 83

5.4 Analyzing the effect of U-NET architecture changes 83

5.5 Feature maps analysis 85

5.6 Conclusions 91

REFERENCES 95

APPENDIX 102

Appendix A - DL layer types 102

Appendix B – Some important concepts during DL training 105

Appendix C - Architectures 107

Figures List

Figure 2.1 - Proton's alignment from natural state to direction of the magnetic field 31

Figure 2.2 - Rat and human axial slice MRI 32

Figure 2.3 - DL's neuron inspired by biological Human's neuron[44] 33

Figure 2.4 - Activation in Human's Neurons[45] 34

Figure 2.5 - Activation Function in DL’s Neuron[46] 34

Figure 2.6 - Filters from different layers. (a) Filters from first layers. (b) Filters from middle layers. (c) Filters from higher layers[47]. 35

Figure 2.7 - Loss-function example graph [42] 37

Figure 2.8 – Some examples of DL's tasks in medical imaging 39

Figure 2.9 - Example of some difficulties in classification problem using common photos 40

Figure 2.10 - Example of intra-class variation problem in brain slices 40

Figure 2.11 - Localization of a cat using a simple architecture using AlextNet [58] 41

Figure 2.12 - The essential role of Upsampling in segmentation[58] 41

Figure 2.13 - AlexNet using an Upsampling layer to perform segmentation task[58] 42

Figure 2.14 - Example of architectures using the technique called "skip process" [58] 42

Figure 2.15 – DeconvNet architecture [58] 42

Figure 2.16 - U-NET architecture illustration 43

Figure 2.17 - Normalization process in MRI image 44

Figure 2.18 – Histogram of a MRI brain axial slice 45

Figure 2.19 - Histogram equalization effects on brain image [61] 45

Figure 2.20- Histogram equalization result using the current dataset 46

Figure 2.21 - DSC function 48

Figure 2.22 - Example of the labels created using argmax 48

Figure 2.23 - Example of feature maps created from DL algorithm after 15 epochs 49

Figure 2.24 - TensorBoard view of the training's performance 49

Figure 3.1 – Dataset used for each Rats case. (a) A Rat axial slice brain. (b) Label mask for brain tissue. (c) Semantic segmentation WGMCSF mask (In green is the WM, in purple the GM and in blue the CSF) 52

Figure 3.2 - Number of Rats cases for each set 52

Figure 3.3 - Semantic Segmentation in Rat's MRI brain 53

Figure 3.4 - Semantic Segmentation using full MRI image. (a) Ground truth. (b) Generated mask. 53

Figure 3.5 - DL pipeline for semantic segmentation with brain extraction 54

Figure 3.6 - Processing Pipeline for semantic segmentation, highlighting the BE segmentation 54

Figure 3.7 - Processing pipeline for semantic segmentation, highlighting the WGMCSF segmentation 56

Figure 3.8 - Two different approach of voxels quantification 57

Figure 4.1 - Example of a case of Human's Dataset. An MRI axial slice and its respective labels. 59

Figure 4.2 - Number of human's young cases MRI in each phase 60

Figure 4.3 - Number of humans' elderly cases MRI in each phase 60

Figure 4.4 - Segmentation errors of a mask generated using FSL BE 61

Figure 4.5 - ASEG Masks creating new and more precise BE masks 61

Figure 4.6 – BE segmentation using a single DL model 62

Figure 4.7 – Unbalanced data issue found using axial slices of MRI 63

Figure 4.8 - Brain segmentation result in Humans. Problems related with unbalanced data. 63

Figure 4.9 - DL pipeline to extract slices without brain 64

Figure 4.10 - Pipeline to extract the brain slices, highlighting the slice classification DL step 64

Figure 4.11 - Pipeline to extract the brain slices, highlighting the slices selection, after the DL classification 65

Figure 4.12 - Selection of the axial slices with brain tissue BE segmentation. 66

Figure 4.13 – Selection of slices Python code 66

Figure 4.14 - Pipeline to extract the brain slices, highlighting the Brain Segmentation DL step using slices with brain tissue 67

Figure 4.16 - Processing Pipeline for semantic segmentation of WM and GM 68

The objective of the WGMSC segmentation was to segment 4 different tissues the WM, GM, subcortical tissues and cerebellum. The same pipeline was used, where the brain is extracted before the semantic segmentation. Figure 4.17 illustrates the pipeline used for WGMSC segmentation. 68

Figure 4.18 - Processing pipeline for semantic segmentation of WM and GM, subcortical tissues and cerebellum 69

Figure 4.19– Keras output after changing dice loss function to asymmetric loss function 70

Figure 4.20 - Training values visualization after changing weights in loss function 71

Figure 4.21 – Comparison between young brain and extreme alteration[68] 72

Figure 4.22 - Ground truth and the respective result of an older case with the best DL model 73

Figure 4.23 - New dataset to train DL model again adding elderly cases 74

Figure 5.1 - Approach to prevent underfitting used in Rats and Humans semantic segmentation 76

Figure 5.2 - Training set's loss function for each epoch 77

Figure 5.3 - Validation set's loss function for each epoch 77

Figure 5.4 - Values of DSC to compare BE and WGMCF phase using 2D or 3D DL approach in Rats 78

Figure 5.5 - Values of DSC to compare BE, WGM and WGMSC segmentation of DL in Humans 79

Figure 5.6 – Test case of abnormal big ventricles in a young case (~30 years old) 79

Figure 5.8 – Case of bad labelled data. (a) Subcortical tissues MRI. (b) Wrongly segmented ground truth. (c) Generated mask. 80

Figure 5.9 - Values of DSC to compare BE and WGM in outliers and normal cases before and after the new training 81

Figure 5.10 - Comparison between ground truth, generated mask before training with outliers and generated mask after training with outliers 82

Figure 5.11 - BE of a younger brain with large ventricles before and after training with outliers 82

Figure 5.12 - Comparison between DSC values of Rats and Humans experiment 83

Figure 5.13 - Feature maps in slice number 20 of the U-NET architecture (4 x 4) 84

Figure 5.14 - Changed U-NET for a simpler approach 84

Figure 5.15 -Example of final feature maps of the first convolutional and max pooling layers using Rat's MRI 85

Figure 5.16 - Example of final feature maps of the first convolutional and max pooling layers using human's MRI 86

Figure 5.17 - Location in the U-NET of the feature maps created in the first CNN block 86

Figure 5.18 – Feature maps from second CNN block from U-NET 87

Figure 5.19 - Feature maps from third CNN block from U-NET 87

Figure 5.20 - Feature maps of fourth block from U-NET CNN both in humans and rats 88

Figure 5.21 - U-NET center feature maps of both humans and rats 89

Figure 5.22 - Feature maps from the first merge block in Rats and Humans 89

Figure 5.23 - Feature maps from final merge block in Rats and Humans 90

Figure 5.24 - Final generated masks in Humans and Rats 90

Figure 5.25 - Pipeline for subcortical tissues segmentation 94

Figure A.0.1 - Convolution layer applied in an image and a resulting feature map [71] 102

Figure A.0.2 – Max-pooling transformation 103

Figure A.0.3 - The effects of a dropout slice in a training network [73] 104

Figure A.0.4 - Example of a dense layer 104

Figure B.0.5 –The three different types of training difficulties [74]. 106

Figure B.0.6 - Overfitting case , A training graph of TensorBoard 106

Figure C.0.7 - Architecture U-NET used for 2D convolutions approach in Rats 107

Figure C.0.8 - U-NET architecture used in 3D convolutions approach in Rats 108

Figure C.0.9 - U-NET architecture 2D approach for semantic segmentation for Rats 109

Figure C.0.10 - U-NET architecture 3D convolutions approach for semantic segmentation for Rats 110

Figure C.0.11 - Classification NET used in classification of slices with brain and slice without brain 111

Figure C.0.12 - U-NET architecture used for brain segmentation in Humans (2D convolutions approach) 112

Figure C.0.13 - U-NET architecture used for brain semantic segmentation in Humans (2D convolutions approach) 113

Tables List

Table 2.1 - Activation Functions

Table 2.2 - Testing different types of pre-processing

Table 3.1 – Final results of BE segmentation in testing set of Rats

Table 3.2 - Final results of WGMCSF segmentation in testing set of Rats

Table 4.1 - Final results of BE segmentation in Humans using two-dimensional convolutions approach

Table 4.2 - Final results of WGM segmentation in Humans using two-dimensional convolutions approach

Table 4.3 - Mean of the number of voxels in each area of the training and validation set

Table 4.4 - Final results of WGMSC segmentation in Humans using two-dimensional convolutions approach

Table 4.5 - Final results of BE and WGM segmentation in Humans using outliers

Table 4.6 - Before and after testing images with new DL model trained with elderly patients, using 61 elderly for testing

Table 4.7 - Younger testing set results before and after using elderly patients in the training phase, using 43 young cases for testing

Table 5.1 - Different parameters used in different testing phases

Table 3.3 – Values of RMSE for each tissue. Comparison between segmentation and quantification using deep learning.

Notation and terminology

General Notation

The notation in this study follows the next rules:

- **Italic text** – for foreigner words or to emphasize a term or expression.
- **Bold text** – To highlight a word or expression.

Acronyms

ANN	*Artificial Neural Networks*
BE	*Brain Extraction*
CNN	*Convolution Neural Networks*
CSF	*Cerebrospinal Fluid Mask*
FCN	*Fully convolutional Networks*
DL	*Deep Learning*
DNN	*Deep Neural Networks*
DSC	*Dice Similarity coefficient*
ML	*Machine Learning*
MRI	*Magnetic Resonance Imaging*
WGM	*White and Gray Matter*
WGMCSF	*White, Gray matter and Cerebrospinal Fluid*
WGMSC	*White, Gray matter, Subcortical tissues and Cerebellum*
WM	*White matter*
GM	*Gray matter*

Chapter 1- Introduction

Biomedical engineering is the newest engineering that merges the engineering and healthcare field. Biomedical engineers often work on tissue regeneration, genetic engineering, pharmaceutical engineering, artificial body parts, medical devices, imaging methods[1]. Together with doctors, therapists, researchers, they develop informatics systems, equipment's and devices to improve healthcare field solving clinical problems. Biomedical engineering can have different advanced training in other fields such as clinical and biological research, the creation of mechanical prosthesis or equipment's, electronic devices development, medical informatics and others[2].

In the last decades, the medical and Informatic fields have mixed, grown and changed dramatically together, improving the healthcare system[3]. Medical informatics studies and pursues the effective use of biomedical data, information and knowledge for scientific inquiry, problem-solving and decision making with the same continuous objective: To improve the healthcare[4].

In the beginning, medical informatics was something seen as “nice-to-have” and not “need-to-have” as it is nowadays. The informatics field has already evolved to the point where it can be used as a helping tool in diagnosis, in data storing, communication, resources management, sensor, signal and imaging informatics, bioinformatics and others[3].

1.1 Clinical investigation using Magnetic Resonance

Medical Imaging is a field which aims at developing non-invasive methods to create a visual representation of images inside the body for medical purpose. There are several different types of medical imaging modalities such as X-ray, Computer Assisted Tomography and Magnetic Resonance Imaging (MRI) [5].

Magnetic resonance imaging (MRI) scan is a procedure that uses strong magnetic uses a strong magnetic field and radio waves to create detailed images of the organs and tissues within the body [6]. Functional magnetic resonance imaging (fMRI) is also used in the research field, which uses MRI technology to measure cognitive activity by monitoring blood flow in certain brain areas. The blood flows where neurons are active, giving the researcher an insight of the activity of neurons in the brain. This technology helps to understand the function of normal and injured brains [6]. Standard MRI scans are useful for detecting anomalies in tissue structure. However, an fMRI scan can help detect anomalies in activity [6].

Importantly, countless studies of neuroimaging are made each year both in rats and humans in clinical and research contexts. Nowadays, the neuroimaging field is allowing investigators to access the brain 's structure and function in new and better ways.

Medical image data analysis is a part of the computer vision field and seeks to apply informatics new technologies to the medical imaging field. It emerged in the 1990s and has since then advanced tremendously. Neuro-imaging data analysis and population studies are important for the understanding of the brain's function, and how it reacts in a certain situation. It is also useful to study brain diseases such as degenerative disorders, tumours, amongst others, both in human and rat studies. Some were explored in the laboratory ICVS, University of Minho. That's the case of *Stress impact in resting state brain networks* [7] which studied the impacts of stress on the human brain. In this study, functional imaging acquisitions determined that there are some negative impacts on medical students when they are facing exams and feeling the most stressed. There's another similar study of stress studying stressed rats vs control rats, which has some of the same conclusions as the study with humans*: Maladaptive response to stress in rats* [8]. This kind of research helps the healthcare field to understand the effects of different contexts in the animal or human brain. Studies of the rodent's brain are used to understand how the human brain works as it allows for a more complete characterization and better experimental control. This type of studies need the intervention of segmentation tools.

1.2 Computer vision

Computer vision is a scientific field that allows machines to see, extract and identify useful information, analyzing and interpreting it from a single image or multi-dimensional data. It simulates the human eye using theoretical and algorithmic basics to achieve automatic visual interpretation [9] [10]. It is also used as an assistive technology a real-time interpreter, in many fields such as: personal mobility, sensory functions, daily living activities, orthotics and prosthetics, communication and skills training, recreation and sports, house, work, environment and others. [11]

Computer vision uses artificial intelligence to give the machines the ability to recognize patterns and to learn techniques on their own[9]. This new technology allied with medical images is playing a critical role at many stages of the clinical workflow starting with the diagnosis, treatment, monitoring diseases, and in computer-assisted intervention task.

1.3 Machine Learning and Deep Learning

Machine Learning methods are generally divided into supervised, unsupervised land reinforcement learning algorithms. The supervised algorithms start with a model that receives a data set of inputs in pairs x and y (where x is the original image and y is the representation of an instance of a set of classes). While the training phase is happening, the model tries to find [12]. There's also the unsupervised algorithms that process unclassified data and the system tries to find out for itself the patterns, making the main objective to find the hidden patterns. And at last, the reinforcement learning the system gets a reward when it makes an action. At the end the DL model learns what actions make the biggest amount of rewards[12].

DL began with an algorithmic approach from the early machine-learning crowd, called the Artificial Neural Networks (ANN). ANN are inspired in the biology of human's brain, specifically the interconnections between the neurons. But, unlike a biological brain where any neuron can connect to any other neuron within a certain physical distance, these artificial neural networks have discrete layers, connections, and directions of data propagation [13]. It's a machine's learning sub-area which uses deep neural networks.

Techniques for automated medical imaging analysis started as soon as it was possible to scan and load into a computer the acquisitions. In the beginning, the 1970s to the 1990s period, medical imaging analysis was done with low-level pixel and mathematical modelling to construct compound rule-based systems that solved different and simple and low-complex tasks[12]. At the end of the 1990s, the supervised techniques started to gain attention in medical imaging analysis, being nowadays still a successful technique of many commercially available medical image analysis systems[12]. Besides that, in the old days, there were several limitations like overfitting, lack of computer power, it was impossible at that time to solve real medical imaging dilemmas. However nowadays, even with some limitations, the world of computer programming has changed drastically, and new algorithms have appeared with the GPU improvement: The DNN network algorithms. Since 2012, it appeared new and innovate DL approaches that have exhibit impressive results. At this time, the computer can solve visual problems just like a human [14].

1.4 DL in Medical Imaging

DL applied in medical imaging is getting popular, and many researchers believe that it can substitute human's diagnoses in fifteen years. Intelligent machines will achieve better accurate results compared to older non-intelligent algorithms. The first fields to be revolutionized will be the ophthalmology, pathology and cancer radiology [15]. Recent reviews show that DL have highlighted that DL was a successful approach in a wide range of medical image analysis tasks, e.g. segmentation, classification, detection, registration, image reconstruction, and others. These tasks are used cross an extensive range of anatomical fields, i.e. brain, lungs, abdomen, breast, prostate, musculature, and others.[5].

In neuroimaging, due to the extension variation from patient to patient, the traditional machine learning methods are not currently reliable. DL revolutionizes because it's the most successful and effective approach in the market. Its basic computational unit is the neuron, which is a concept inspired by the study of the human brain. It takes multiple signals as input, it combines them linearly using weights and passes the combined signals through non-linear operations generating nonlinear operations to generate output signals [16].

1.5 Context and Motivation

MRI in neuroimaging is currently the technique most utilized when monitoring neurological diseases and conditions since it provides high resolution and high contrast in soft tissues. Approaches using DL are gaining attention because of their ability to interpret and self-learn a big quantity of data images [17].

Quantitative analysis is presently a routine for many patients with neurological diseases, making segmentation and labelling an important component in their monitoring. The task of segmentation is the selection of a set of voxels that contour a certain structure of interest.

Organ's or structure's segmentation allows quantitative analysis and get parameters such as volume and shape. It allows to detect morphological changes within across different groups of subjects [18][19].

An alternative solution to the issue of segmenting uncommon brains is to require a human to do the manual segmentation of the brain. While this is still the gold standard in many institutions and would present good results compared to the common semi-automatic methods, it would still present some issues: it would be even more time consuming, especially in larger studies that sometimes have hundreds of subjects; it would still be prone to human error and bias; and it would require specialized researchers capable of doing such segmentation. In our view the true answer to this problem needs to be the full automation of this process which provides a good accuracy and consistency [17].

DL in medical imaging is currently one of the best new ways to get a more accurate and generalized results compared to traditional algorithms. Studies have shown that DL algorithms can be successful and even replace older non-DL based algorithms [12].

With all this in mind, the main objective of this work, is to create a DL algorithm that segments some target tissues, both in rats and humans since DL algorithms have been showing through the years really good results and a promising potential.

1.6 Related Work - Humans

There are already several segmentation approaches/tools optimized for the human brain such as Freesurfer [20], EMS (Expectation-Maximization Segmentation) [21] and FMRIB Software Library (FSL) bet [22] and FAST [23]. The most commonly used algorithms that don't use DL for brain MRI segmentation is Freesurfer. Freesurfer is a semi-automated algorithm that requires the user intervention for validation and correction of the classification and which can be very tricky sometimes, especially in elderly subjects. These manual corrections and verifications don't always lead to a good enough result and often lead to the expenditure of a lot of time of the medical image team.

To study some DL's approaches in medical imaging, there are already some surveys in this field. One of them is called: "A Survey on DL in Medical Imaging" and it explores over 300 contributions to the field, using DL for image classification, object detection, segmentation, registration and other tasks in areas such as neuro, retinal, pulmonary, digital pathology, breast, cardiac, abdominal musculoskeletal and others [12].

Another survey in DL applied in brain MRI gathered some of the most important findings in this field [24]. Many medical image segmentation challenges encourage the development of tools to segment lesions, e.g. ILES (Ischemic stroke lesion segmentation) challenge, and tool to segment tumours , e.g. BraTS (Brain Segmentation Challenge) in the brain. There are already some studies on brain's tissue segmentation applied in humans and most of them segment the WM (WM), GM (GM) and CSF (cerebrospinal fluid)[24] .

It is the example of a study [25] on infants' brain which uses four Convolution Neural Networks (CNN) to segment WM, GM and CSF combining T1 and T2 images using them as patches.

Another interesting case revealed a deep learning approach to segment brain tissue using three different CCNs [26] that merge together in the end. This article proposes a deep learning approach using patches of the original image to segment 9 different tissues using T1 T2, and FLAIR images [26].

The survey[24] gathered also some open resources, e.g. : BrainNets (https://github.com/kaspermarstal/BrainNet) [27] created using tensorflow that segments the WM and GM using Inception-v4 and Inception-resnet new architectures; LiviaNet (https://github.com/josedolz/LiviaNET) [30] which uses a 3D CNN for subcortical brain structure segmentation in MRI, using small kernels; Res_Net_mri_ani (https://github.com/neuro-ml/resnet_cnn_mri_adni) [29] exposes an architecture uses techniques of residual CNNs and transfer learning in MRI classification task; Mr Brain (https://github.com/lanpa/mrbrain) [30] is a regression deep learning model that predicts the human's age to usefully detect Alzheimer's disease. It uses VGG-NET[31] architecture.

U-NET based architectures are often successful in the field of medical imaging [12] published by Ronneberger in 2015 [32]. It's the case of a study[33] which used this three-dimensional architecture to segment damaged brain's areas and used a special loss function for highly unbalanced data. Another study based on U-NET architecture segmented the WM, GM and CSF using human's T1 MRI volumes reaching a dice ratio's average of 89,74%. This study uses merges some of the characteristics from U-NET and SEGNET, creating a new architecture called U-SEGNET.

U-NET architecture produces a fast and precise segmentation of images and has outperformed the prior best method in some online challenges for scientific community (a sliding-window convolutional network) [12], [32], [33].

1.7 Related Work - Rats

Even though studies of neuroimaging in rats are popular, there's a big lack of automated analysis of rat brains [19]. An open source software that can be used with some success for rodent segmentation is the statistical parametric mapping (SPM) [34] tools, which can provide an average template, mask and tissue segmentation of WM, GM and cerebrospinal fluid [34][35]. It allows an automatic tissue segmentation using templates and prior probability maps for different tissues types [18]. However, this software requires the research team to spend time preparing the inputs for each subject's brain: the original image, the location of anterior commissure in the brain and also requires the existence of appropriate tissue probability maps [34].

The approaches mentioned before for human brain, i.e. Freesurfer [20], EMS (Expectation-Maximization Segmentation) [21], FMRIB Software Library (FSL) *bet* [22] and *FAST* [23]can't be applied in rat's field. Most of this techniques rely on the intensity profiles and shape of the brain, which are very distinct from the rats brain [19].

One of the studies found that studied the segmentation in rats is "*RATS: Rapid Automatic Tissue Segmentation in Rat Brain MRI"* [19]. This study is interesting because it explores some of the tools that already exist in rat's MRI segmentation and reveals a new technique using mathematical morphology and LOGISM'OS-based graph segmentation [19].

No studies were found in the field of segmentation in rats using DL approach.

1.8 Research Questions

With all of this in mind, there are several research questions, which can be raised:

Is it possible to segment the brain from MRI image both in rat and human species using DL techniques?

Is it possible to segment the WM and GM from MRI image both in Rats and Humans using DL techniques?

Is the model able to generalize?

What is the difference between two-dimensional convolutions approach from three-dimensional convolutions approach?

What is the difference between Rats and Humans performance?

1.9 Document's Organization

The Introduction referred some of the history of DL and its evolution in history. It also explored what already exists in the medical field and the importance of using DL in the medical ground. Chapter 2 of technologies and concepts will introduce the functioning of MRI image, which is the type of medical imaging that this dissertation will lay on. It will also explore the DL concepts, and its functioning, introducing the basic unit of DL, the techniques to construct a DL architecture. This chapter will also explore the pre-processing changes in the MRI images before entering in the DL model to improve results and training, the evolution functions chosen to use in the testing phase, and some visualization of the training techniques used like TensorBoard and feature maps. There will also be a deepen study on the chosen final architecture. In the chapter 3- , rats experiment, the materials used for the rats specie will be presented, the BE (Brain Extraction) segmentation and the WGMCSF (White, Gray matter and CerebroSpinal Fluid) segmentation methods, and there will be a further explanation of the pipeline created to improve the final results. Chapter 4- will lay on the Humans Experiment. This chapter is more complex than the Rats because it explores not only the BE and WGM segmentation, but also the Subcortical tissues segmentation and studies the generalization strength of the model using outliers' cases, more specifically elderly's cases. Chapter 3 and 4 chapter will expose the performance of each model using accuracy, sensitivity, sensibility and DSC as evolution functions.

All the discussion of the results of the chapter 3 and chapterChapter 4- , related to rats and humans respectively, are exposed in the discussion and conclusion, chapter 5- . It will also interpret the results and its findings. The research questions posed at the beginning of the research will also be answered. In the end of this chapter, some of the final conclusions are re-state, exploring also the future work.

Chapter 2- Technologies and Concepts

This chapter will expose some of the important technologies and concepts useful to understand the some of the basic concepts of semantic segmentation using DL. Methods used will also be presented.

2.1 MRI

Magnetic Resonance Imaging (MRI) is a non-invasive imaging technology which produces three dimensional anatomical and structural images. It differs from computed tomography (CT) because it doesn't use damaging ionizing radiation of X-Ray, making it a modality of choice when a frequent imaging is required on diagnosis or therapy. Besides that, MRI can also detect more clearly tissues like brain, cord, nerves, muscles, ligaments than CT. In neuroimaging, MRI can differentiate the white from the GM and diagnose aneurysms and tumours. This technology uses strong magnetic fields that excites and detects the changes in the rotational axis of water's protons [36]. Figure 2.1 shows that when a radiofrequency current is pulsed, the protons are forced to change the direction of the rotational axis to achieve equilibrium. When the radiofrequency field is turned off, the MRI sensors are able to detect the energy released by the protons during their realignment with the magnetic field. The amount of energy detected will change depending on the nature of the chemical molecules. At the end, it is possible to be able to understand the difference between the different types of brain's tissues[37][36].

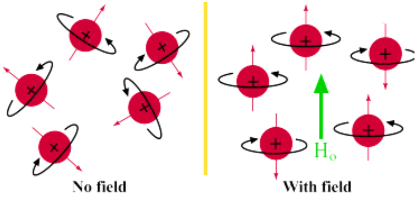


Figure 2.1 - Proton's alignment from natural state to direction of the magnetic field

After an acquisition the MRI is recreated and converted into a 3D volume matrix composed by voxels. A voxel is a volume element representing a value in the three dimensional space, corresponding to a pixel for a given slice thickness. Each voxel value represents the signal intensity in that certain point. Figure 2.2 shows an example of a rat's brain axial slice with a resolution of 64x64x64 and a human's brain axial slice with a resolution of 256x256x256:

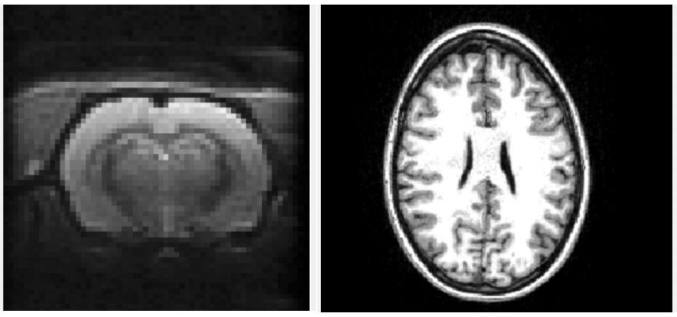


Figure 2.2 - Rat and human axial slice MRI

2.2 Deep Learning

DL is a class of ML (Machine Learning) that uses multiple layers of nonlinear processing units for feature extraction and transformation, it learns in supervised and unsupervised manners and uses artificial neural networks to do tasks such as classification, object detection, speech recognition, language translation and others [38]. DL networks are distinguished from the single-hidden-slice neural networks by their depth which represents the number of node slices that data passes through in a multistep process of pattern recognition [39].

Decades ago, the idea that the software can simulate the neocortex's large array of neurons in an artificial "neural network" emerged. It has led to as many disappointments as breakthroughs. But because of improvements in mathematical formulas and increasingly powerful computers, computer scientists can now model many more slices of virtual neurons than ever before [40].

2.2.1 Artificial Neural Networks

BRAIN'S NEURAL NETWORKS AND ARTIFICIAL NEURAL NETWORKS

DL consists in artificial neural networks that are modelled and inspired by the networks present on the Human's brain. The similarity lies in the way that both store and process the information. This information is stored in weights in artificial Neural networks the same way that the synapses change their signal's strength during learning [41].

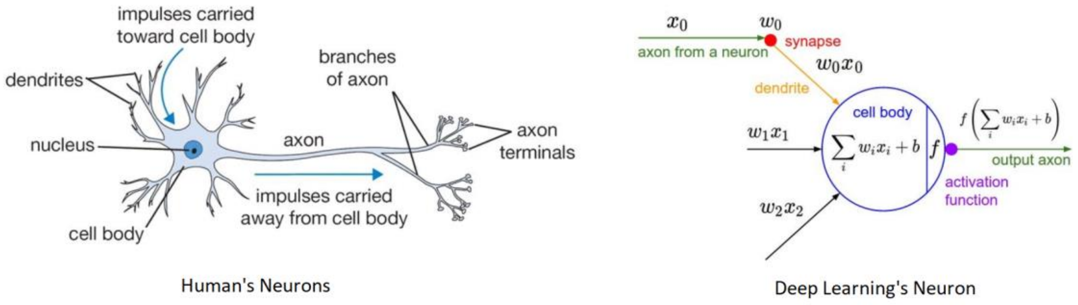
Another similar aspect, is the fact that all the errors and mistakes can improve the performance of the neuron, adjusting their weights to achieve a closer output to the desired one. There is no learning without mistakes [41].

One important characteristic of the artificial neural networks and human's neural networks lies in their ability to generalize to input patterns that were never been exposed while training. This ability happens because the algorithm can see relationships and understand patterns [41]. For example: A DL algorithm that can locate and segment a brain, when facing a different case that has never seen before must be able to generalize and correctly segment the brain, as a human would be able to do too.

Even though the artificial neural networks are similar to the brain function, there are still important differences that should be taken in consideration: The brain has 100 billion neurons and 100 trillion connections or synapses operating on 20 Watts, compared with the biggest neural network that has 10 million neurons and 1 billion connections on 16,000 CPU which is equivalent to 3 million Watts. Another difference is that the brain is limited to only 5 types of data from the 5 senses. [42].

THE BASIC UNIT - NEURON

The neural networks are constructed using the basic building block: The unit or the neuron. Each unit will be organized in a layer with more units. The information enters in input layers, goes through hidden layers and leaves through the output layer. The more layers the neural network has, the deeper is this structure. Each layer has a number of units, which refers to the width of the layer [43]. The Figure 2.3 presents the artificial neuron and the human's neuron.



Human's Neurons

Deep Learning's Neuron

Figure 2.3 - DL's neuron inspired by biological Human's neuron[44]

A single artificial neuron will perform a multiplication between the weights *W* and the input *X* and add the bias *b*. The result passes through an activation function [43] :

- The single artificial neuron will performance a multiplication between the weights *W* and the input *X* and add the bias *b*.
- Each layer has its own bias and weights (θ) and each f_{NN} is the overall function that represents the neural network.

ACTIVATION FUNCTIONS IN THE ARTIFICIAL NEURON NEURAL

In neural networks, the activation function implements the decision about the final output combined with the different weights. In human's biology, the activation of the neural network happens because a potential difference, as it shows in Figure 2.4. In DL's neural networks, this activation function will be activated if the value of the neuron is big enough to pass to the next neuron. If the value of entrance is sufficient, the neural unit will be activated(Figure 2.5) [43].

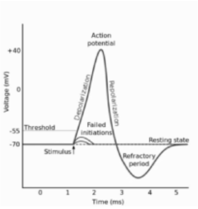


Figure 2.4 - Activation in Human's Neurons[45]

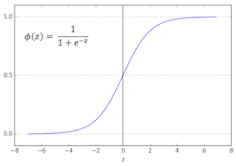


Figure 2.5 - Activation Function in DL's Neuron[46]

There are six types of activation functions: The linear, the sigmoid, the SoftMax, the Rectified Linear (ReLU) and the Hyperbolic tangent, explained in the Table 2.1. The linear is the fastest one in gradient-based learning. The sigmoid is ideal to binary classification problems. The SoftMax is used for multi-classification tasks and it turns the unit into a probability. Each unit receives the x_n and a weight w_n and a b (additional term). Each type of unit calculates differently the output through the following equations [43].

Table 2.1 - Activation Functions[43]

UNIT TYPE	***EQUATION***	***CHARACTERISTICS***
Linear	$y = w.x + b$	Makes gradient-based learning faster.
Sigmoid	$y = \frac{1}{1 + e^{-(wx+b)}}$	For Binary classification problems.
Softmax	$y = \frac{e^{x1}}{\sum_{c=1}^{n} e^{xc}}$	For Multi-Classification tasks. It turns values to probabilities. If the value is high, the probability will near 1.
ReLU	$y = \max(0, wx + b)$	Used as a hidden unit. Leads to large and consistent gradients.
Hyperbolic Tangent	$y = \tanh(wx + b)$	Also used as hidden unit

DEEP NEURAL NETWORKS

Multiple layer creates a more efficient neural network, with the ability to detect complex features: When an image is filtered the output is another image that contains the filter's response at each spatial location. Feature maps indicate where certain features were detected in the image (e.g. edges). However, in DL the feature maps are usually subjects to many other levels of filtering, because of the depth and number of layers. After many successive filtering, feature maps become more descriptive due to the organization of all spatial neurons. Edge-like filters and texture-like filters are frequently observed in early layers of DNN [47].

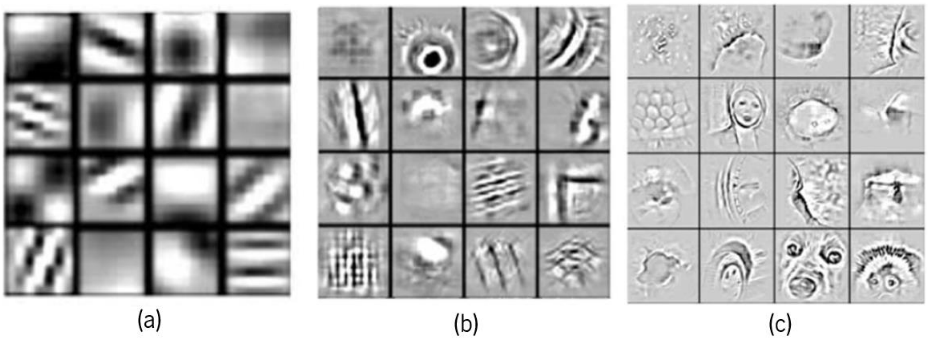


Figure 2.6 - Filters from different layers. (a) Filters from first layers. (b) Filters from middle layers. (c) Filters from higher layers[47].

Each layer in DNN involves filtering the feature map produced on the previous layer, as the DL layers continue training, the receptive field of neuron or feature become larger. Higher level layers detect larger features[47], as it can be seen in the Figure 2.6.

This is called the feature hierarchy and it makes deep-learning networks capable of handling big and high-dimensional and complex dataset. DL also performs automatic feature extraction without the human's intervention contrasting with machine learning algorithms.

When training unlabelled data, each node learns features automatically by trying to construct the final input, trying to minimize the difference between the network's guesses and the ground truth.

2.2.2 Model Optimization

DL models learn by updating parameters of neural network's weight and biases towards the direction of correct classification. The learning phase happens in the training where [43] :

- A loss function $l(\mathrm{y\_prediction}, \mathrm{y\_true})$ will compute the output of the neural network with its actual result. $l\left(f_{NN}(x, \theta), \mathrm{y\_true}\right)$
- A gradient of this loss is computed by: $\Delta l\left(f_{NN}(x, \theta), \mathrm{y}_{\mathrm{true}}\right)$
- The bias and weights (θ) are readjusted based in the gradient and loss function using an optimizer function

LOSS FUNCTIONS

Output's evaluation gives the DL model the ability to comprehend if the learning if proper or not. At the end of each epoch, the DL algorithm will compare the prediction with the ground truth value, using the loss function [43]. This loss value will be used after to update new values of weight and bias to improve loss-function. The choice of a correct loss-function is a huge factor for an accurate performance of the DL algorithm [42]. There are a several number of different loss functions depending on the task.

The loss function is metaphorically a "mountain", a visual example is presented in Figure 2.7. The main objective is to find the lowest place (global minimum)[42].

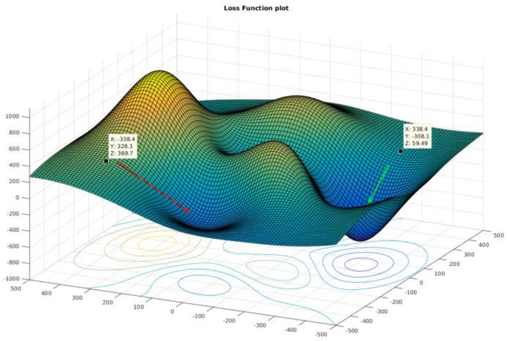


Figure 2.7 - Loss-function example graph [42]

There's more than one local minimum and depending on the weights initialization the model can find more easily the global minimum or get retained in a local minimum [42]. Some of the loss functions decided to use in this work are:

- **Binary Cross-Entropy in classification**

The binary cross entropy is appropriated for binary classification tasks, where the ground truth are label numbers ($y\_true \in \{0,1\}$) and the predictions represent the probability of being of some class $y\_prediction \in \mathbb{R}$. The following expression represents the binary cross entropy loss function[43]:

$$Binary\_Cross\ Entropy = -\sum_{i=1}^{n} y_1 \, log\, f(x_i, \theta) + (1 - y_i) log(1 - f(x_i, \theta)) \qquad (1)$$

- **Dice Similarity Coefficient loss function (DSC Loss) in binary segmentation**

This loss function measures the spatial overlap between two segmentations [48].

$$DSC_{LossFunction} = 1 - \frac{2|P \cap G|}{|P| + |G|} \qquad (2)$$

The DSC will be deeply explored in the chapter 2.4 - Evaluation metrics.

- **Asymmetric Similarity Loss Function in binary segmentation for unbalanced data**

When facing a case of unbalance data, it is important to adjust the weights of FPs and FNs (and achieve a better balance between precision and recall)[49]. The solution would be an asymmetric loss function such as:

$$F(P,G,\beta) = \frac{(1+\beta^2)|PG|}{(1+\beta^2)|PG| + \beta^2|G\setminus P| + |P\setminus G|} \qquad (3)$$

This asymmetric function [49] had the DSC loss function as the starting point. If β=1, the function turns into the DSC. Using a higher β the asymmetric loss function, it helps to shift the emphasis to decrease FNs and boost the recall.

OPTIMIZERS

The direction, speed and jumps in which the weights and biases will update will be dictate by the optimizer chosen. The backpropagation algorithm is significant for the DL to improve the results after analyzing the cost/loss function. This algorithm will calculate the errors gradient.

It was originally introduced in 1970s, but it was only recognized in 1986 in a paper called *"Learning representations by back-propagating errors"* [50]. The backpropagation uses an expression for the partial derivate of the cost function with weights or bias in the network that represents the speed that the cost changes when changing weights and bias.

There are several optimizers: SGD [51], Adam [52], Adadelta[53], RMSprop[54], Adagrad[55] and others. All of them have a different way to pick the direction, speed and jumps to achieve the global minimum.

2.2.3 CNN Building Blocks

Deep neural networks are compose computations performed by many levels[47]. There are already some golden rules for a good DL processing in the field of imaging. One of them is the use of CNN building block[43]. A CNN block is very efficient and effective to detect and extract important visual features. It has the ability to generalize as well across several tasks[56]. A CNN is constituted by:

- A detector as an activation function
- The convolution, detector and pooling operations are applied in sequence
- The output typically is passed to other layers (convolution or fully connected)
- This building block can be put in parallel consuming the same input and producing multiple outputs or feature maps.

The CNN will be composed by the block of convolutional layer, pooling layer and fully connected, which are explained in the appendix A.

2.2.4 DL Tasks

There are several tasks that the DL algorithms can efficiently do, as it is presented in the Figure 2.8. Some of them are: Classification task which is a single output for a full image; The localization task which returns a bounding box of the localization of the object that the model wants to detect; The localization and object recognition which returns all the bounding boxes of all located objects correctly classified; The binary and semantic segmentation which returns a map of the image's size with all the pixels classified as 1 or 0 (binary) or more class numbers (semantic), respectively.

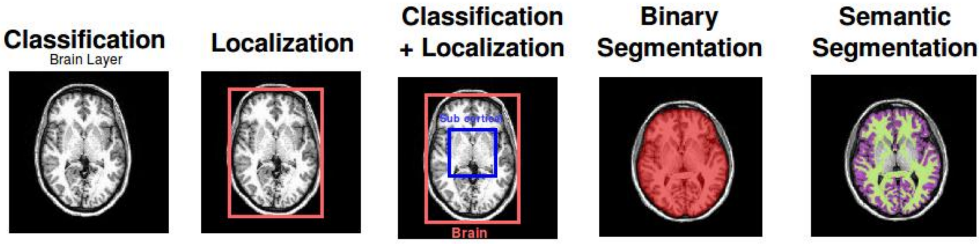


Figure 2.8 – Some examples of DL's tasks in medical imaging

For each task the architectures differ and already got some golden rules that should be taken in consideration. In this work, the architectures used in Classification and the Segmentation tasks will take a deeper study since they are the ones used further.

Some of the layers referred bellow are explored in the chapter Appendix A - DL layer types.

2.2.5 Architectures structure

CLASSIFICATION

Image classification is a simple task that assigns a unique label to an input image. It was one of the first areas in which deep learning made a major contribution to medical image analysis [12]. Even though it seems an easy problem to solve, there's a large variety of practical applications and it's one of the biggest and harder tasks in computer vision[57].

That difficulties can happen due to the image's variations such as luminosity conditions, scale changes, objects deformation, intra-class variation, the object is partially occulted and others.

For example, when trying to recognize a cat in a DL model, it becomes difficult when the data set can have all these types of variations. For example, the all the images of Figure 2.9 should be labeled as “cat”, but they show a big variation between them[57].

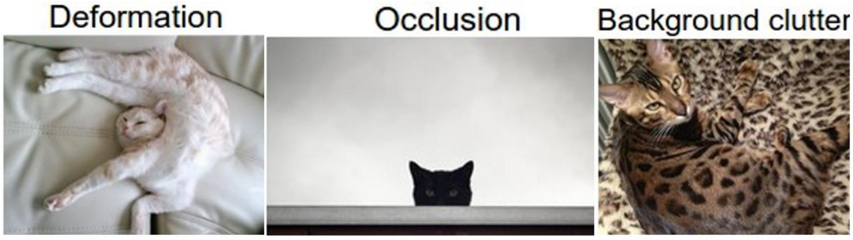


Figure 2.9 - Example of some difficulties in classification problem using common photos

Image classification was one of the first areas where DL made a major contribution to medical image analysis. For example, from a single exam it’s possible to detect and classify the image with the presence or absence of a disease[12].

In the case of brain’s MRI imaging, the images usually don’t have some of problems referred. Problems like occlusion, deformation and background clutter don’t happen in MRI since the acquisition task is very controlled and the images are quite similar to each other: The brain is often centered in a black background. The variations can happen in luminosity variation, depending on the acquisition parameters, scale variations (baby brain and adult brain) or intra-class variation. In the case of intra-class variation, if the main objective is to classify a slice with brain tissue from a slice without brain tissue, then the algorithm should be able to classify all the images of Figure 2.10 as brain, even though there’s a large variety of diseases.

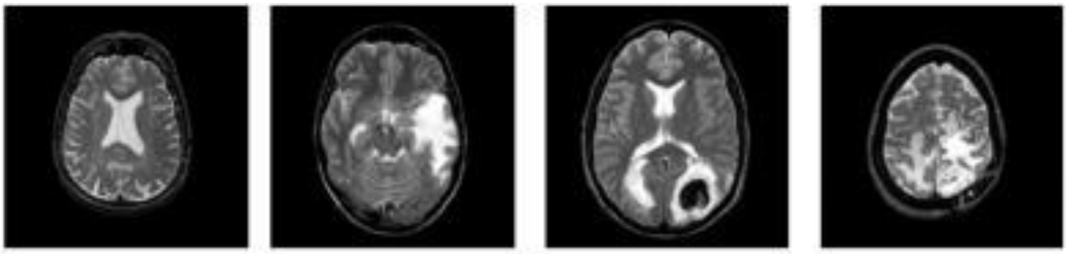


Figure 2.10 - Example of intra-class variation problem in brain slices

SEGMENTATION

The segmentation task is typically defined as identifying the set of voxels which belong to a certain substructure, tissue or other material of interest. Is the most popular and common task used in the DL papers applied in medical imaging [12]. In the last years, a technique called

FCN (fully convolutional networks). The main difference is in FCN the architecture is composed of convolutional layers without any fully-connected layers or MLP (Multilayer perceptron)[58].

A FCN tries to learn representations and make decisions based on local spatial input, while using fully connected layer enables the network to learn something using global information where the spatial arrangement of the input falls away and need not apply.

Looking at classification and localization problem, the simple *AlexNet* would work efficiently for the effect. Using fully connected layers helps for the DL model understands the global information and localization of the target [58]. When trying to perform a segmentation task using FCNs (only convolutional layers), the deep learning model fails, and can't segment the cat, as it is shown in Figure 2.11 [42].

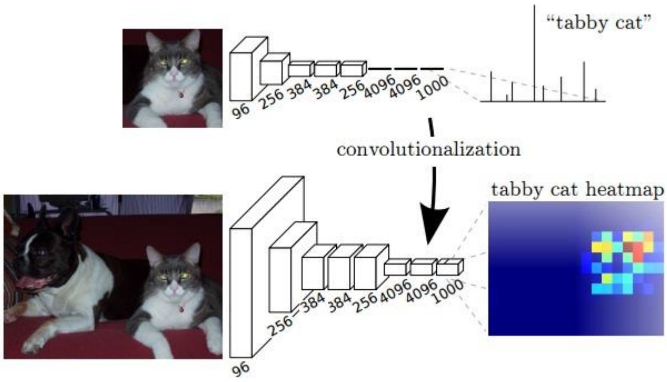


Figure 2.11 - Localization of a cat using a simple architecture using AlextNet [58]

One way to convert this problem into a segmentation problem would be to changing the final architecture adding up-sampling layers:

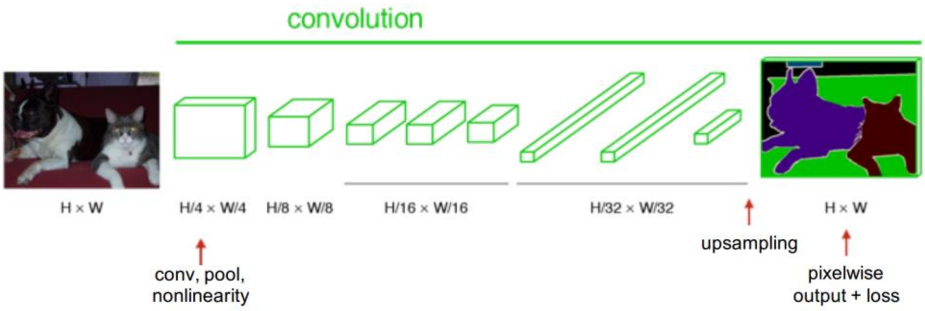


Figure 2.12 - The essential role of Upsampling in segmentation[58]

Using Convolutional and Up Sampling layers at the end of the architecture contributes to a final DL model that can execute a segmentation task, as it is seen in Figure 2.12 [58]. The only problem using this kind of approach is that resolution is often lost because of the use of several max-pooling layers [42]. A case of low resolution is shown in Figure 2.13:

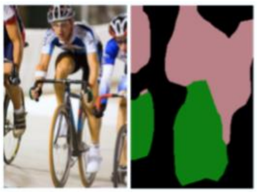


Figure 2.13 - AlexNet using an Upsampling layer to perform segmentation task[58]

One of the way to achieve better results would be to get some information from previous layers to keep a higher resolution. The Figure 2.14 shows a case of using the “skip-process” in a DL architecture:

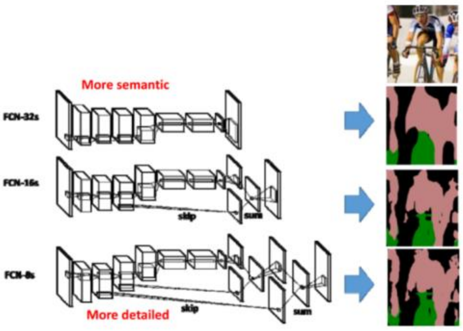


Figure 2.14 - Example of architectures using the technique called "skip process" [58]

Another method that can help to create a better segmentation is to use architectures like *Deconvnet* (Figure 2.15). The *Deconvnet* operates using a *VGG-NET* normal followed by a VGG-NET reversed, where Pooling layers transform into Upsampling layers [42]:

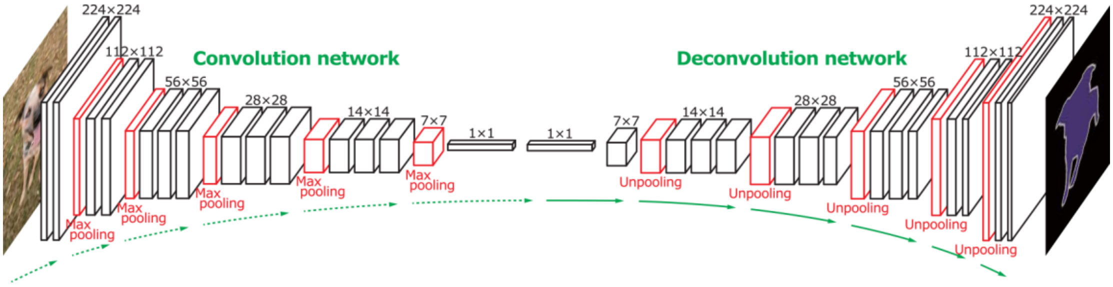


Figure 2.15 – DeconvNet architecture [58]

Even though the *Deconvnet* has a very interesting approach in the DL community to improve the resolution of the final images, there’s still the problem that most of the image’s information is brutally reduced in contracting phase, which can compromise the final result [42].

One FCN architecture used in medical imaging segmentation is U-NET [12] published by Ronneberger in 2015 [32]. One paper [33] used this 3D-U-NET architecture to segment damaged brain's areas and used a special loss function for highly unbalanced data. This architecture produces a fast and precise segmentation of images and it has outperformed the prior best method in some online challenges for scientific community [12], [32], [33]. The architecture is represented in the following image:

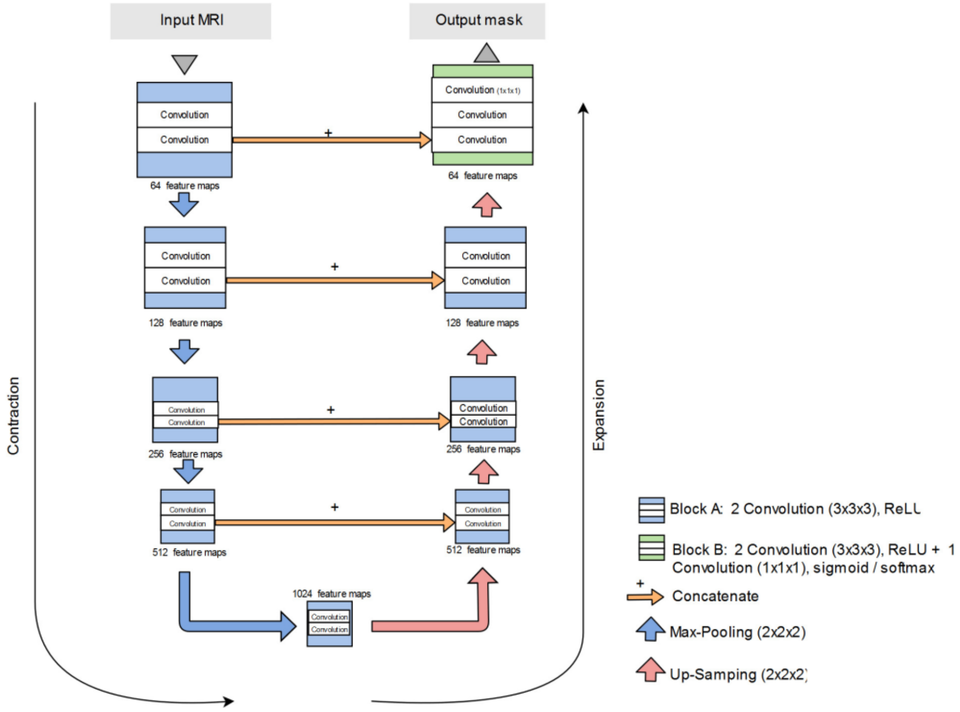


Figure 2.16 - U-NET architecture illustration

Figure 2.16 shows an architecture that uses both "skip process" and *Deconvnet* deconvolutional phase. The two main novelty of the architecture is the combination of an equal number of Upsampling and Max-Pooling layers and the use of information from previous layers to create feature maps with more resolution [12], [32]. At first, the layers are submitted to a spatial contraction and the details get increased in the feature channels. This first step has CNN's combined with Max-Pooling and have the main objective to understand the details and characteristics that the image has, having a condensed feature map as a result. Finally, after the contraction there's the expansion using CNN, Up-Convolution and Concatenate. This final step takes the image's feature maps from the previous layers and combines them, making a high-resolution feature map as the final result, which was one of the reasons that this kind of architecture had significantly better results than the DeconvNet [32]. This allows U-net to take into account the full context of the image, which can be an advantage in contrast to patch-based CNNs [12], [32].

2.3 Pre-processing methods

Pre-processing can be important in a DL model. The more similarity there is in the images, the simpler and faster would be for the DL model to learn features and segment correctly. So, in the beginning it was tested 2 possible types of pre-processing: normalization and histogram equalization.

2.3.1 Linear Normalization

Normalization has the main objective of place all image's values between a certain range. In this case, the normalization was between 0 and 1. The normalization formula is described equation 4:

$$x_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{4}$$

The process of normalization can be visualized in Figure 2.17:

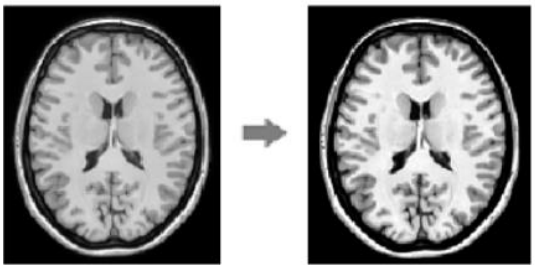


Figure 2.17 - Normalization process in MRI image

2.3.2 Histogram Equalization

The histogram is a representation of the voxel intensity in a distribution graph, as it is shown in Figure 2.18. Using fslview [59], it is possible to check the histogram graph [60]:

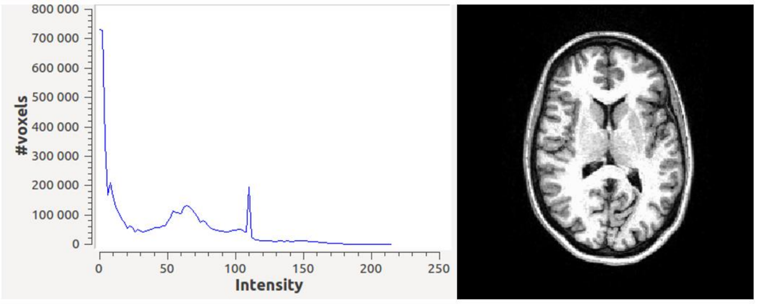
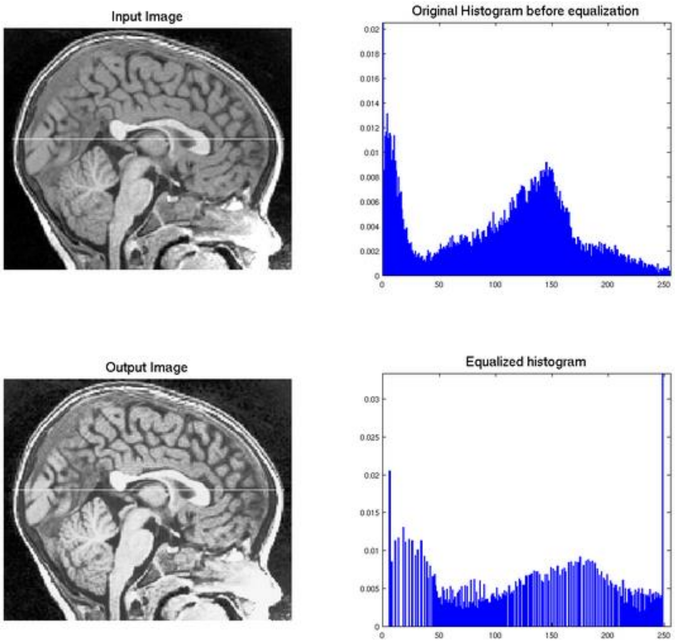


Figure 2.18 – Histogram of a MRI brain axial slice

The histogram equalization helps to improve the contrast in an image, increasing the stretch of the intensity range. In the image bellow, the voxels with lower value of intensity became darker, and the voxels with higher value of intensity became even more lighter, stretching the original histogram graph [60]. Figure 2.19 shows what happens to an MRI image after the process of histogram equalization.


Input Image

Original Histogram before equalization

Output Image

Equalized histogram

Figure 2.19 - Histogram equalization effects on brain image [61]

2.3.3 Select pre-processing tools

To understand the effects of the pre-processing normalization and histogram equalization, it was tested the accuracy changes after 10 epochs with some human's images in BE model, as it is described in Table 2.2.

Table 2.2 - Testing different types of pre-processing

Without any pre-processing	93.11 %
With normalization only	96.49 %
With histogram equalization only	88.92 %
With normalization and histogram equalization	90.77 %

After analysing the response of the DL model's accuracy, it was evident that some problems in the histogram equalization happened. The image bellow shows the MRI after a process of histogram equalization:

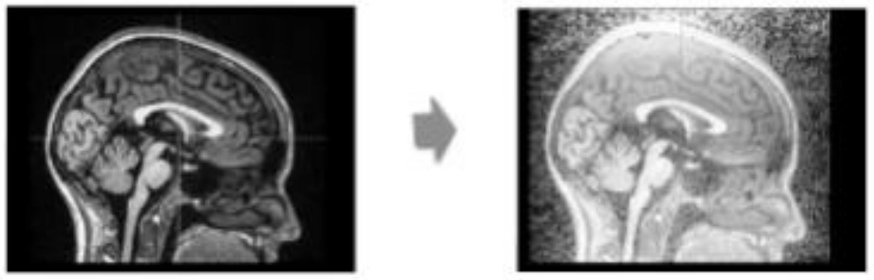


Figure 2.20- Histogram equalization result using the current dataset

The histogram equalization didn't work for this MRI images because it has noisy pixels across all the image in the background (Figure 2.20), where the image should be in solid black. After many attempts trying to use different approach of histogram equalization to improve this result, it was decided that this type of pre-processing would be put aside.

At the end, normalization was the pre-processing method chosen. Even though there are more pre-processing tools, they were not explored since the models had good performance with only the normalization pre-processing step.

2.4 Evaluation metrics

This section will explore the assessment methods to evaluate the performance of the final DL model.

Accuracy metric in Keras can be differently calculated depending on the type of problem to solve. If it's a case of binary classification or segmentation, the value of accuracy can be calculated as [62]:

$$Acc = K.mean\left(K.equal(y_{true}, K.round(y_{pred})\right) \tag{5}$$

It suggests that 0.5 is the threshold to distinguish between classes in the predictions. At the accuracy of a binary classification reveals the mean of the number of times that the rounded prediction is equal to the real classification.

Another three metrics were added add are sensitivity, specificity and dice coefficient similarity.

SENSITIVITY

Is the ability of the model to predict correctly a segmented brain. It returns the probability of being correctly classified when the voxel is the tissue to segment.

$$Sens = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{6}$$

SPECIFICITY

The ability of the model to predict correctly when a voxel has no tissue to segment.

$$Spec = \frac{True\ Negatives}{True\ Negatives + False\ Positives} \tag{7}$$

DICE SIMILARITY COEFFICIENT

This metric measures the spatial overlap between two segmentations (P – Predicted segmentation mask; G- Ground truth segmentation mask). [48]

$$DSC = \frac{2|P \cap G|}{|P| + |G|} \tag{8}$$

Which can be calculated as:

$$DSC = \frac{2 \times True\ Positives}{2 \times True\ Positives + False\ Positives + False\ Negatives} \tag{9}$$

The Figure 2.21 exhibits how the DSC function works: The more percentage of the overlap segmentation between the Ground Truth and the Predict mask, the better results will have.

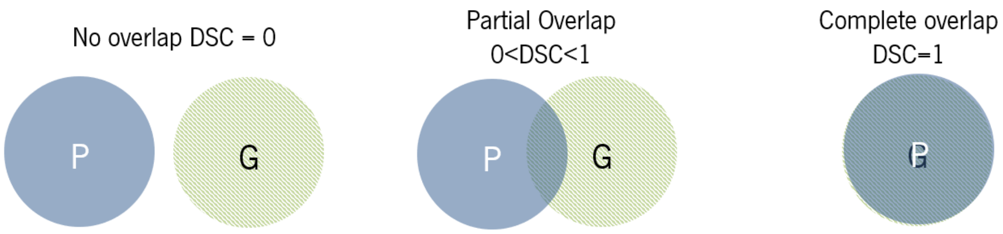


Figure 2.21 - DSC function

In multi-tissue segmentation, the *argmax* function from Keras selects the biggest classification value of each voxel and then the mean of all classification maps. The Figure 2.22 shows the color maps after *argmax* function. Each map represents a different tissue segmentation with values of 0 or 1:

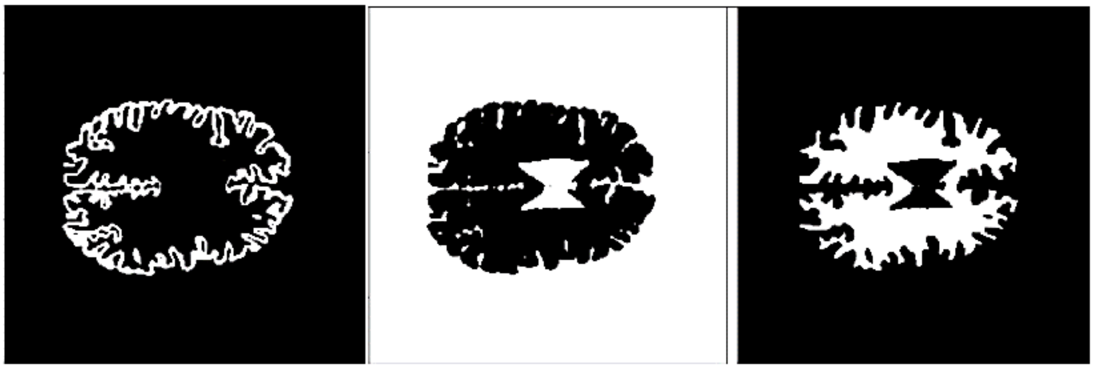


Figure 2.22 - Example of the labels created using argmax

The multi-tissue segmentation evaluation is calculated in the same way as binary classification. All the False-Negatives (FN), True-Negatives(TN), False-Positives(FP) and True-Positives(TP) of all classification maps were sum and then calculated the overall sensitivity, specificity and dice similarity coefficient.

2.5 Visualizing the Learning Process

This chapter will present the how the visualization of the learning process can be carried out. The observation of feature maps to understand if the model was able to understand the images and create accurate feature maps, and the use of TensorBoard for analyzing the values of loss function during training.

2.5.1 Feature Maps

Saving the feature maps was an extra element that helped to understand if the machine was making good choices when creating the feature maps (preserving important information and diminishing noise and background). Figure 2.23 shows an example of the feature maps created during a DL training.

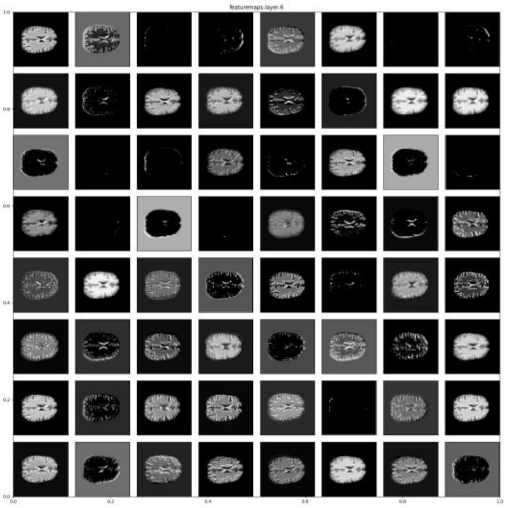


Figure 2.23 - Example of feature maps created from DL algorithm after 15 epochs

2.5.2 Tensorboard

Tensorboard was also a useful tool that TensorFlow community created[63]. TensorBoard operates by reading TensorFlow events files, which contain summary data during training, allowing a faster and easier way to understand the accuracy and other metrics levels, all represented in graphs, as it is shown in Figure 2.24. Each graph contains information of the training set metric value as well the respective validation set metric value, which can be important to understand the behaviour of the training.

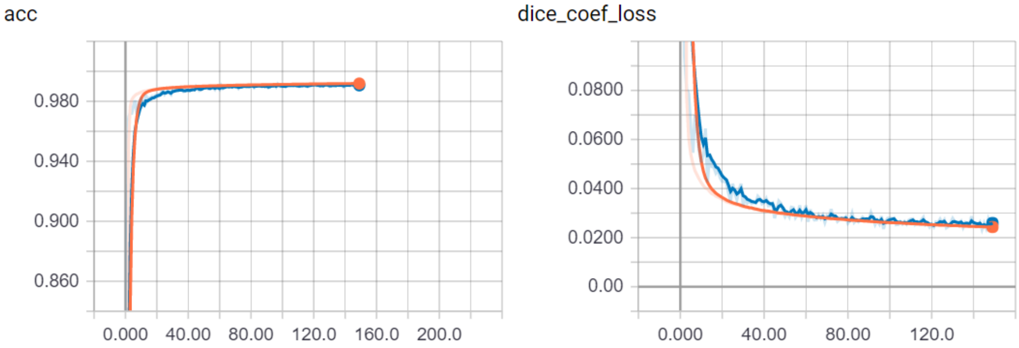


Figure 2.24 - TensorBoard view of the training's performance

2.5.3 Computer specifications

- Ubuntu 14.04 LTS (64 bit) as the operative system
- CPU – Intel Xeon E5-1650
- RAM – 64Gb
- Secondary memory:
    - 2 Disks of 2TB and 1 disk of 512Gb
    - GPU – NVIDIA P6000
    - Cuda Parallel-Processing Colors 3840
    - GPU 24 GB GDDR5X
    - FP32 Performance 12 TFLOPS

Chapter 3- Rats Experiment

It was important to start with rat MRI images. Since brain segmentation is a similar problem both in rats and humans, it would make sense to start with rats since the images are significantly smaller (rats studies have 40 axial slices of 64x64 voxel images and humans have 256 axial slices of 256x256 voxel images) which leads to a faster training. They are also more controlled images on the acquisition phase since they were all acquired for research e.g. all the 2D axial slices had brain tissue, so there was no need to remove some of the slices, as it will be necessary in the human's experiment. Since the data has the same characteristics, the results from the study with Rats is a good starting point to resolve the similar problem in Humans. All the results are discussed in chapter 5.1.

3.1 Rats Dataset

A total of 80 Wistar rats between 14 and 18 weeks of age were used in this study and imaged at different time-points for a total of 174 valid scanning sessions. Scanning was done on a Bruker Biospec 11.7T pre-clinical scanner, using a 2x2 surface coil and running Paravision 6. SE-EPI diffusion sensitive acquisitions were used, with Time to Repetition=5s, Time to Echo=20ms, in-plane resolution of 0.375x0.375 mm, slice thickness of 0.5 mm over 40 slices and a Field-of-View of 24 mm. 10 volumes where b=0 s/mm^2 were acquired and averaged for each scanning session. SPMMouse (ref: SPMMouse: a new toolbox for SPM in the animal brain) was used to locate the center of the FoV over the anterior commissure and SPM *segment* (https://www.fil.ion.ucl.ac.uk/spm/) to create gray and white mater and cerebrospinal masks using the priors created in the SIGMA project [64]. The rat MRI data-set used was acquired as part of the SIGMA project, co-financed by *Fundação para a Ciência e Técnologia* and *Agence National de Recherche* (FCT-ANR/NEU-OSD/0258/2012)[64].

The final purpose of using rats MRI is to create a DL model able to detect and segment three different brain structures using DL techniques: The gray matter, the white matter and the cerebrospinal fluid (WGMCSF Mask). Figure 3.1 shows an example of an input MRI's axial slice (Figure 3.1.a), its respective label of brain tissue (Figure 3.1.b) and WGMCSF mask (Figure 3.1.c).

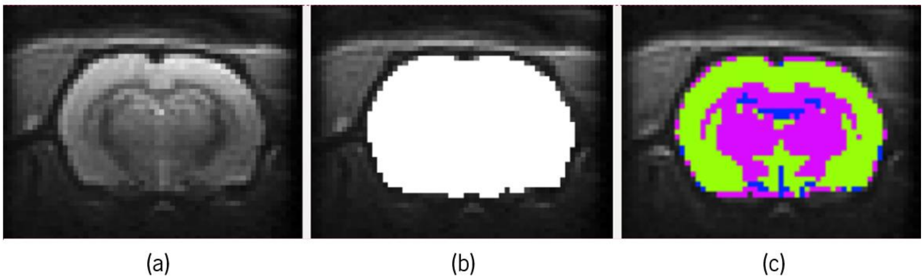


Figure 3.1 – Dataset used for each Rats case. (a) A Rat axial slice brain. (b) Label mask for brain tissue. (c) Semantic segmentation WGMCSF mask (In green is the WM, in purple the GM and in blue the CSF)

In the rats BE (brain extraction) segmentation, the number of images available in total are 174 images with brain segmentation. It results in a tensor with a shape of (174, 40, 64, 64, 1) considering the colour channel.

For the semantic segmentation of 3 tissues (WGMCSF), only 138 of the 174 images were used, because 36 of the images did not have the corresponding label. It results in a tensor with a shape of (138, 40, 64, 64, 1) considering the colour channel. The following figure shows the distributing of each volume: (110 in training, 88 in training, 22 in validation and 28 in testing).

Figure 3.2 presents the number of cases for training, validation and testing for each segmentation: BE segmentation and WGMCSF segmentation.

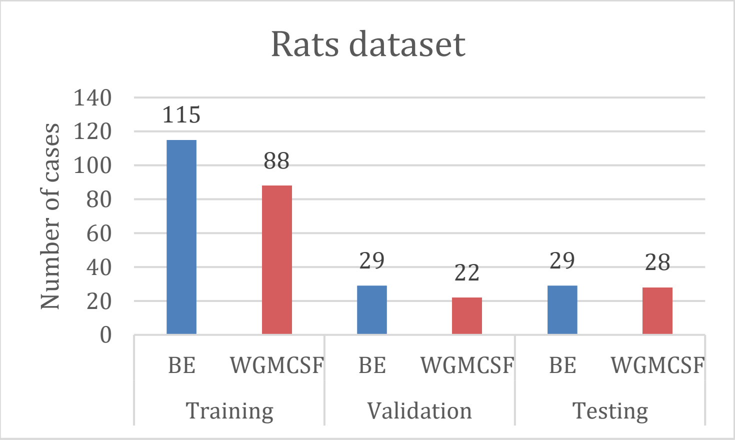


Figure 3.2 - Number of Rats cases for each set

3.2 Rat Brain Semantic Segmentation – First Approach

The main objective is to create a semantic mask that will segment the white matter, gray matter and cerebrospinal fluid. In the first approach a DL model was created that would receive the full MRI original image and return the semantic WGMCSF mask, as it is shown in Figure 3.3.

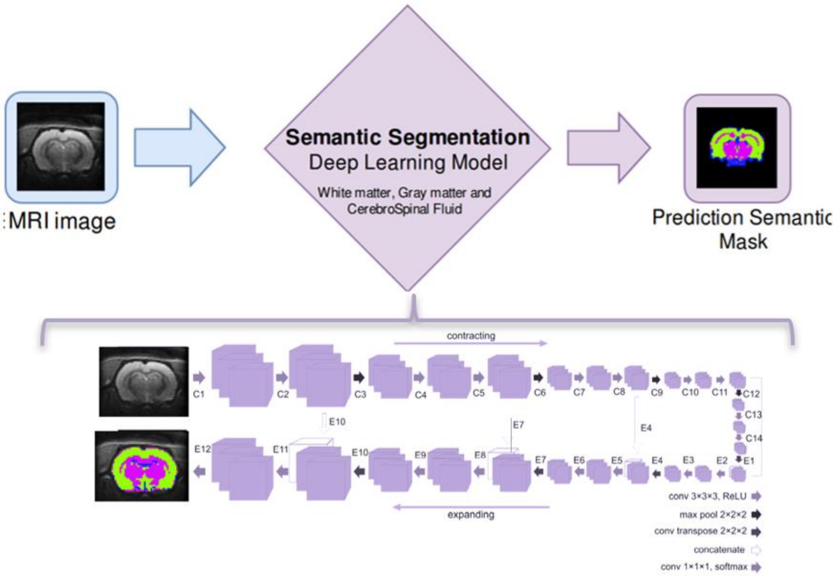


Figure 3.3 - Semantic Segmentation in Rat's MRI brain

The neural network structure chosen to try this first approach was based on the FCN U-NET. For further details on the U-NET architecture check Figure 2.16. In the first tests the model was clearly overfitting. Figure 3.4 represents the testing of one image with the best DSC value model, that was saved in the middle of training and its respective ground truth mask.

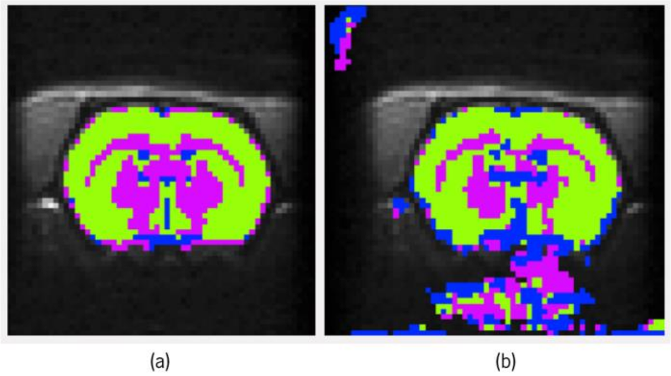

(a)
(b)

Figure 3.4 - Semantic Segmentation using full MRI image. (a) Ground truth. (b) Generated mask.

This image presents some segmentation errors, particularly the voxels segmented in non-brain area.

3.3 Rat Semantic Segmentation – Second approach

One way to overcome the problems reported in the previous section, it would be to create a DL model that segments the brain and turn all the non-brain voxels into 0 in the MRI images. Use in the end the ''only-brain' volumes to create a second DL model that segments the 3 different types of tissues (Figure 3.5). This approach flow will be used both with two-dimensional convolutions and three-dimensional convolutions approaches.

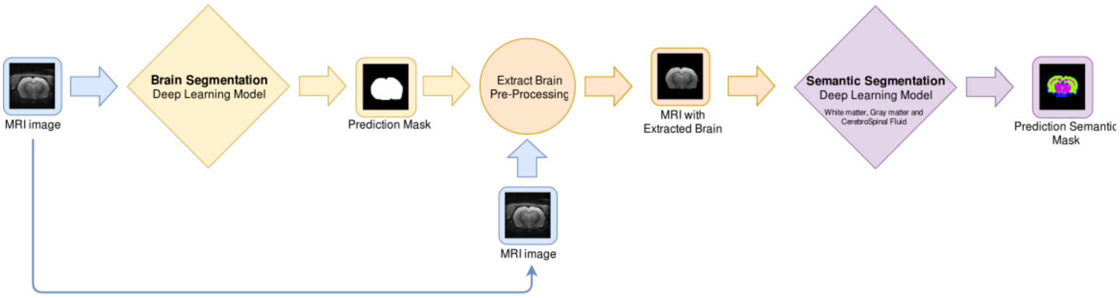


Figure 3.5 - DL pipeline for semantic segmentation with brain extraction

In the following sections, each of the two phases will be presented separately i.e. Rat Brain Extraction and Rat Semantic Segmentation after BE.

3.3.1 Rat Brain Extraction

The brain extraction happens in two phases: First the DL model will create a mask from the original MRI image of all voxel considered brain tissue classified as one (white) and the remaining voxels as zero (black). Using the BE mask created, the original MRI is processed and transformed into an MRI with only brain voxels, as it is shown in Figure 3.6.

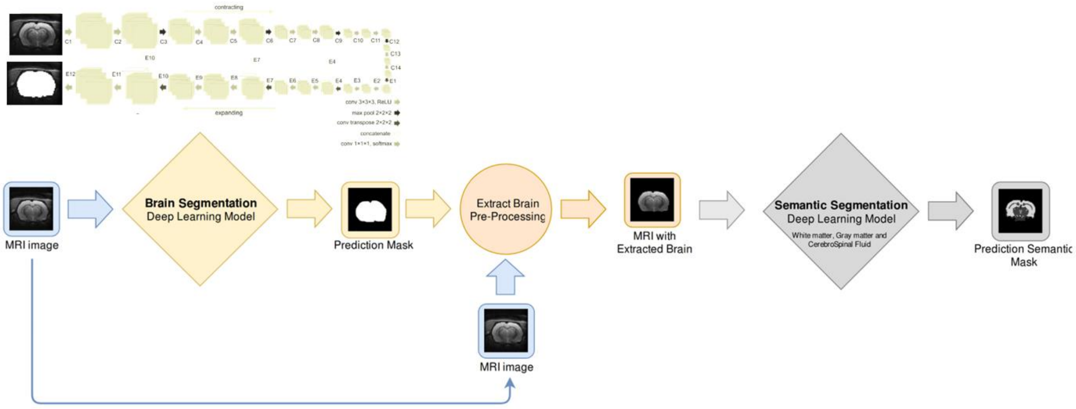


Figure 3.6 - Processing Pipeline for semantic segmentation, highlighting the BE segmentation

The architecture both in slice-by-slice (two-dimensional convolutional approach) and 3D volume (three-dimensional convolutional approach) were inspired by U-NET (U-NET is deeply explored in Figure 2.16). Most of the fine tuning happened using dropout and batch normalization slices in order to improve the results and reduce overfitting.

The DL model with the lowest value of loss in the validation test used SGD as optimizer function and DSC as loss function. Details about the architecture is presented in the section of Appendix C - Architectures. The results obtained of 96% and 98% for the DSC are accurate (Table 3.1 – Final results of BE segmentation in testing set of Rats).

Table 3.1 – Final results of BE segmentation in testing set of Rats

	2D convolutions	3D convolutions
Accuracy	98.59 %	99.63 %
Sensibility	94.89 %	98.59 %
Specificity	99.47 %	99.77 %
DSC	**96.18 %**	**98.42 %**

3.3.2 Rat Semantic Segmentation after BE

The architecture was inspired again by U-NET, more deeply explored in Figure 2.16. Again, some of the fine tuning happened when adding dropout slices and batch normalization slices in order to reduce the overfitting. The best DL model chosen was the one with the lowest value of loss in the validation test using SGD as optimizer function and dice coefficient loss as loss function. The architectures are in the section of Appendix C - Architectures. Figure 3.7 shows the final phase of the pipeline: The semantic segmentation after BE segmentation.

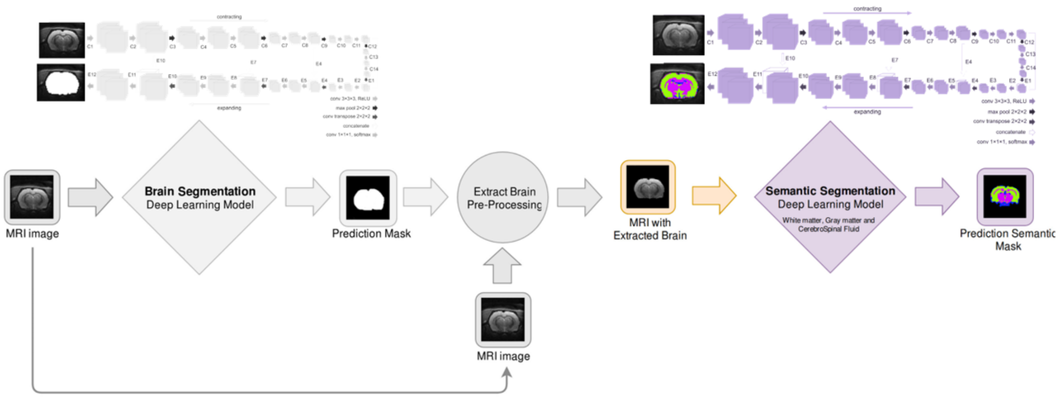


Figure 3.7 - Processing pipeline for semantic segmentation, highlighting the WGMCSF segmentation

Since in this phase is not important to understand if the DL model can detect background, it was decided that the next results only calculate the important classification tissue maps (WM, GM and Cerebrospinal liquid). The background classification map was put aside in this phase because it would influence the results in a deceiving way. The results are shown in the Table 3.2.

Table 3.2 - Final results of WGMCSF segmentation in testing set of rats

	2D convolutions	3D convolutions
Accuracy	99.96 %	99.29 %
Sensibility	92.33 %	89.61 %
Specificity	99.97 %	99.64 %
DSC	**90.85 %**	**89.69 %**

Dice similarity coefficient was also calculated for the different tissues individually: The results for a two-dimensional convolutions approach using DSC were 94.65 % for WM, 91.03% for GM and 76.89 % for cerebrospinal fluid. Using the three-dimensional approach, the results using DSC found are 93.81 % for WM, 89.69 % for GM and 74.68 % for cerebrospinal fluid.

3.4 Voxels' quantification using Deep learning regression

There are several ways to count the number of voxels for each tissue of the brain. One of them can be through the segmentation method explored in this dissertation, using a non-DL quantification to predict the number of voxels after a segmentation DL model (Figure 3.8, upper line). Another way would be to create a DL model which predicts directly from the image the number of voxels that belong to a certain tissue (Figure 3.8, lower line).

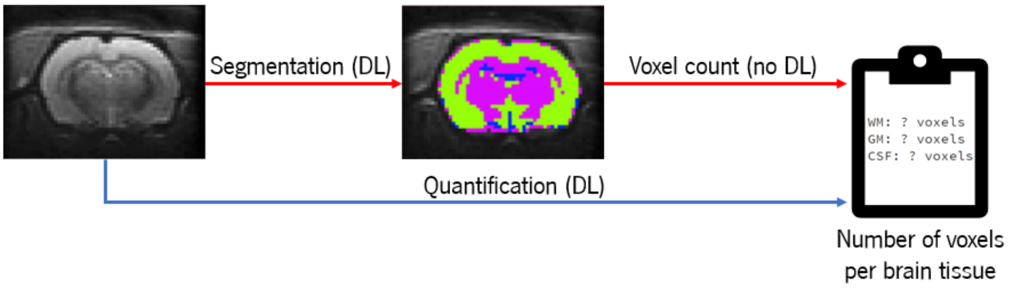


Figure 3.8 - Two different approach of voxels quantification

This section will compare these two approaches in a collaboration with another study "A deep learning approach for quantification of brain characteristics using MRI"(2018), the master's dissertation of João Fernandes. This study created a CNN new approach to find the number of voxels using regression method.

Both studies used the same rat's dataset to quantify the number of voxels. This dissertation used the segmentation method and João's study used a quantification deep learning model. The two studies were compared using 2D (slice-by-slice) convolutions and 3D (volume) convolutions. The results were compared using the root-mean-square error as it is shown in table 3.3**Erro! A origem da referência não foi encontrada.**.

Table 3.3 – Values of RMSE for each tissue. Comparison between segmentation and quantification using deep learning.

	2D convolutions			**3D convolutions**		
	WM	GM	CSF	WM	GM	CSF
Segmentation	2,4	2,0	7,9	4,3	1,6	9.1
Quantification	5,7	2,0	10,4	6,7	3,0	10,4

These results were expected since in segmentation method each voxel has a classification method and deals with probabilities. Each voxel is associated with a certain probability of belonging to each tissue which means that a voxel can be considered as classified correctly even though there's a small probability of not belonging to the same tissue. In quantification, the regression method directly predicts a continuous number instead of 3 classes. Therefore, all this uncertainty will be translated to the continuous number.

It is important to notice that the results do not explicitly conclude which method is better. Even though RMSE it is the best possible metric for a comparison between segmentation and quantification, it is not the most accurate metric to understand the segmentation method. For example, a segmentation can have the very similar number of voxels as the ground truth for a certain tissue but, this segmentation may be dislocated (i.e. miss some voxels where the tissue is present and segment voxels from other tissues).

Chapter 4- Humans Experiment

In the sequence of results obtained with the rats dataset, a similar experiment was tested with humans MRI brain images. It was unviable to use a volume view (3D convolutional approach) due to the memory limitations of the computer used for training processing equipment limitations. The system would have to handle images 16 times larger than rats' images and need to create an architecture with much more learning parameters. This chapter describes the human's dataset, trains a DL model for brain extraction as it happened in rats and a DL model for semantic segmentation.

4.1 Humans Dataset

Human dataset was acquired on a clinically approved 1.5T Siemens Magneton Avanto (Siemens, Erlangen, Germany) in Hospital de Braga. All protocols and sessions of data acquisition were done under projects [65] approved by Ethics Committee of Hospital de Braga (Portugal). A T1 scan MPRAGE acquisition was used with TR=2730 ms, TE=3.48ms, FA=7°, an isometric resolution of 1mm was used, with 176 slices and a Field-of-View of 256mm.

The brain extraction (BE segmentation) followed by the white and gray matter segmentation (WGM segmentation) was the main objective. Also, one interesting study carried out was the segmentation of WM, GM, sub-cortical and cerebellum areas (WGMSC). It was an interesting and challenging task to test, especially because the subcortical tissues are difficult tissues to segment with traditional algorithms. Figure 4.1 shows a visual example of the final labels and its respective axial slice.

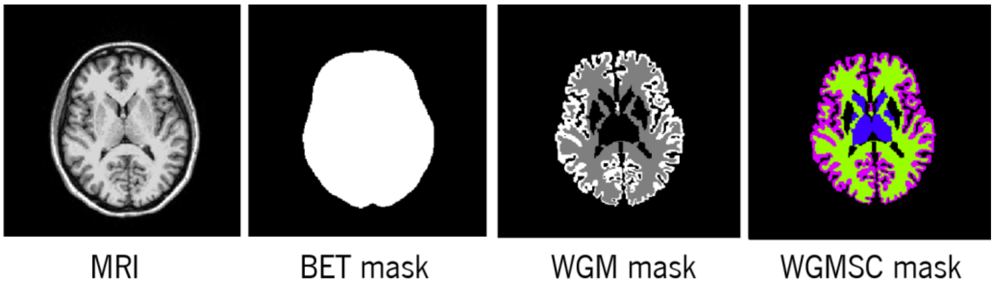


Figure 4.1 - Example of a case of Human's Dataset. An MRI axial slice and its respective labels.

Humans dataset for training phase had 54 images with their corresponding BE (Brain Extraction Mask) and ASEG corresponding Mask conceived by *freesurfer* (Automatic subcortical segmentation of a brain volume). This mask is based on the existence of an atlas containing probabilistic information on the location of brain's structures [66].

In the Human BE, WGM and WGMSC segmentation, the number of images available in total are 97. It results in a 2D tensor with a shape of (97 x 256, 256, 256, 1) considering the colour channel. Figure 4.2 shows the distributing of each volume (44 in training, 10 in validation and 43 in testing).

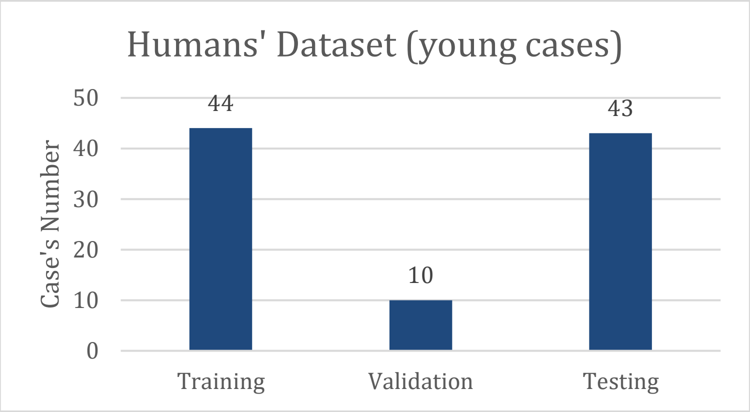


Figure 4.2 - Number of human's young cases MRI in each phase

An experiment with elderly cases was also tested. These images were acquired with the same parameters as the previous dataset [65]. The total number of elderly cases images available in total are 92. It results in a 2D tensor with a shape of (92 x 256, 256, 256, 1) considering the colour channel. Figure 4.3 presents the distributing of each volume (25 in training, 6 in validation and 61 in testing).

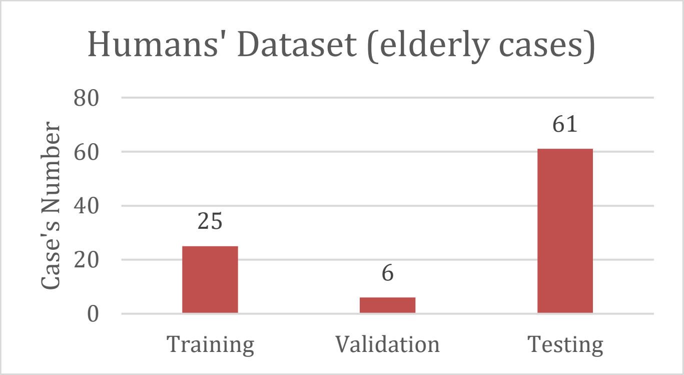
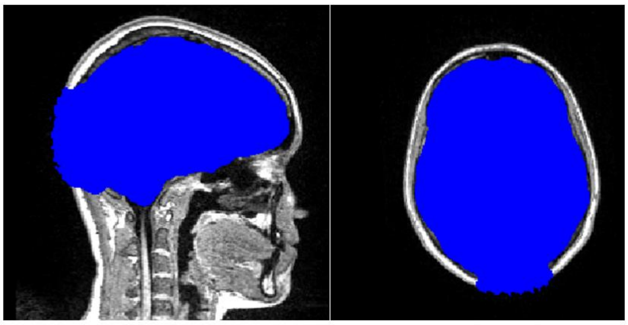


Figure 4.3 - Number of humans' elderly cases MRI in each phase

MASKS PRE-PROCESSING

Some of the first tests of BE segmentation began with a great capacity of learning, however the accuracy didn't reach values of 90% of accuracy. Analysing the images and respective ground truth masks, it was noticeable that some of the brain's masks created using *FSL BET* [59] had segmentation errors. This can lead to a less accurate training phase and consequently have imperfect results. In Figure 4.4 it is notable an example of this incorrect classified mask:



SAGITTAL SLICE AXIAL SLICE

Figure 4.4 - Segmentation errors of a mask generated using FSL BE

Automatic subcortical SEGmentation (ASEG)'s masks [66] were available and had a more precise segmentation. Each voxel is assigned of about 40 labels, including all the tissue's brain. These masks are produced by *freesurfer* [66] New BE labels were created using the ASEG's masks applying some transformations. Figure 4.5 illustrates the pre-processing of these masks: ASEG Mask was transformed in a binary mask and applied a Gaussian smoothing using fslmaths. Finally a threshold was established (The brightest voxels over 40% of luminosity became 1 and the rest of the image turn to zero).

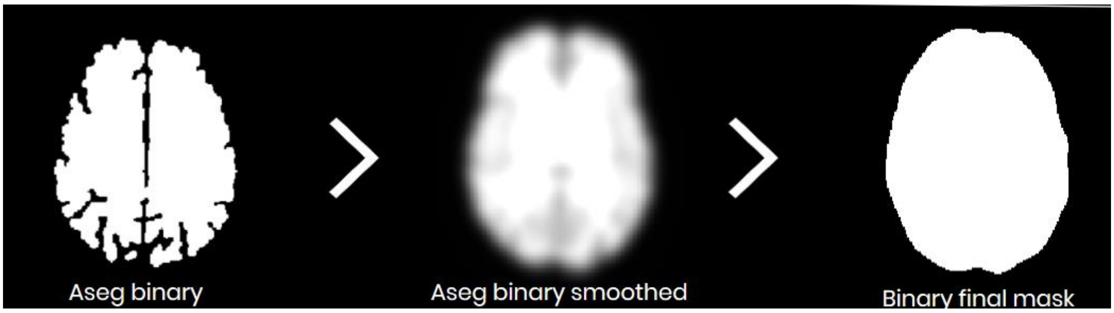


Figure 4.5 - ASEG Masks creating new and more precise BE masks

While training the DL model, it was visible that the validation accuracy before the BE masks substitution never reached 90%, and with these new BE masks created using ASEG mask the validation accuracy reached rapidly levels of 98% in accuracy. Accurate labels matter and it interfere in the success of the DL model training.

The WGM and WGMSC masks, shown in Figure 4.1, were created using the ASEG corresponding Mask obtained from by *freesurfer.*

4.2 Human Brain Extraction – First approach

To begin the study with humans, for a 2D convolutions approach, it was decided to use the axial layers to train in BE segmentation. The input would be the MRI axial slices and the output would be the label of the tissue's brain for each slice. The Figure 4.6 illustrates this pipeline. In this phase the U-NET was used, more deeply studied in Figure 2.16.

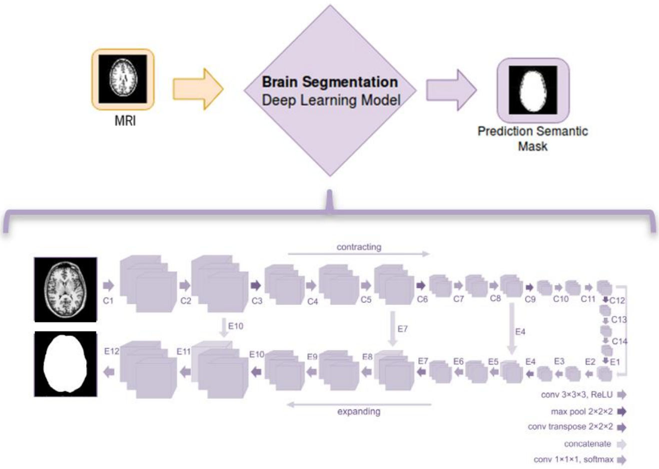


Figure 4.6 – BE segmentation using a single DL model

It was noticible that there was some difficulties in the first trainings. When looking at the dataset it was clear that the DL model was facing a unbalanced dataset when using axial slices. As it can be seen in Figure 4.7, it is possible to notice that relatively half of the slices don't present brain tissue:

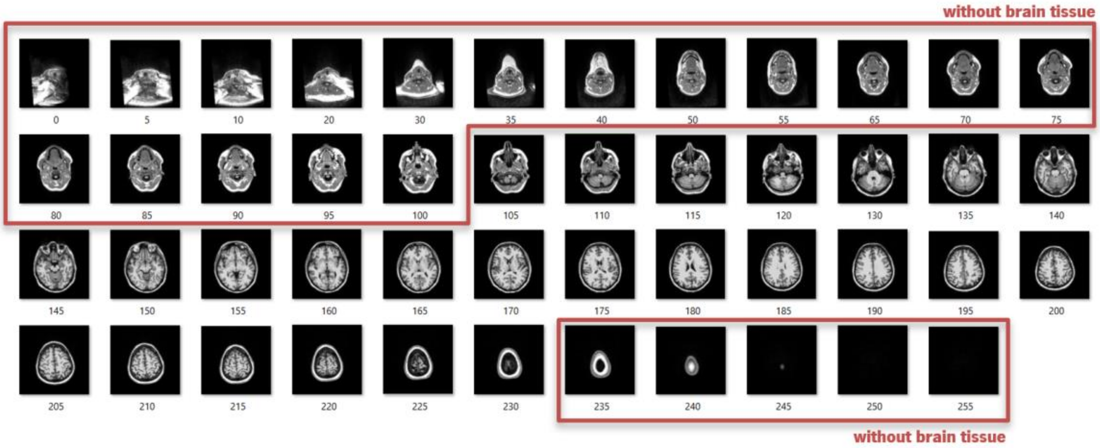


Figure 4.7 – Unbalanced data issue found using axial slices of MRI

The best DL model found using all the axial slices was using an U-NET based Architecture(Figure 2.16) using the Tversky loss function for unbalanced data [33]. This training was undoubtedly a time-consuming task, and the values of accuracy were under 85%. Figure 4.8 represents one of the test images and corresponding mask generated by the best DL model:

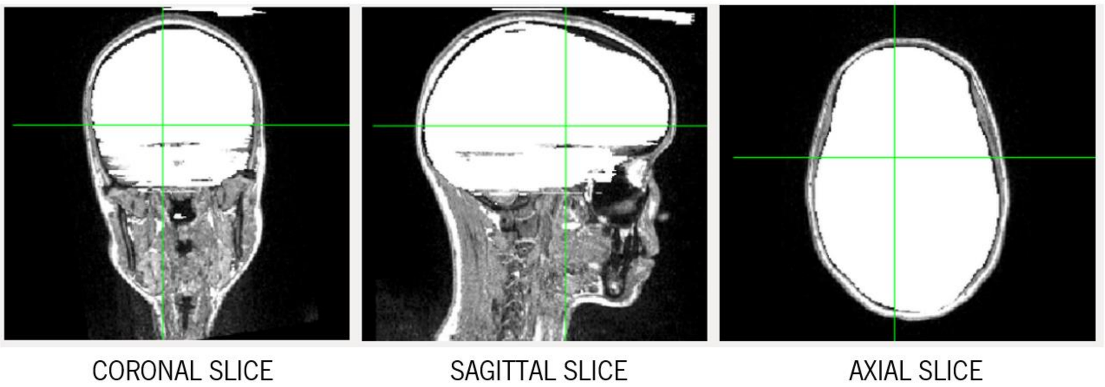


Figure 4.8 - Brain segmentation result in Humans. Problems related with unbalanced data.

There are some errors in the brain's segmentation especially in the area above the eye, where some slices are not recognized as brain voxels. Using the axial views in Humans can be quite difficult to train the neural network, considering that nearly half of the slices to segment the brain do not exist, making the DL model forced to distinguish an axial slice with brain tissue from an axial slice without brain tissue.

4.3 Human Brain Extraction – Second approach

To prevent consuming more time and relieve the computer's resources, a DL flow was created to prevent the segmentation DL segmentation model to distinguish from slice with brain from slice without brain. Figure 4.9 proposes a new pipeline.

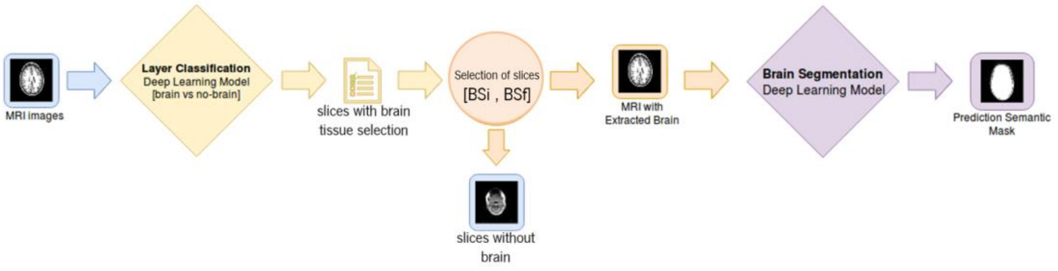


Figure 4.9 - DL pipeline to extract slices without brain

4.3.1 Classification DL Model Brain and No-Brain Axial Slice

Using this new pipeline, the focus will be in the first DL Classification slice in this current section (Figure 4.10):

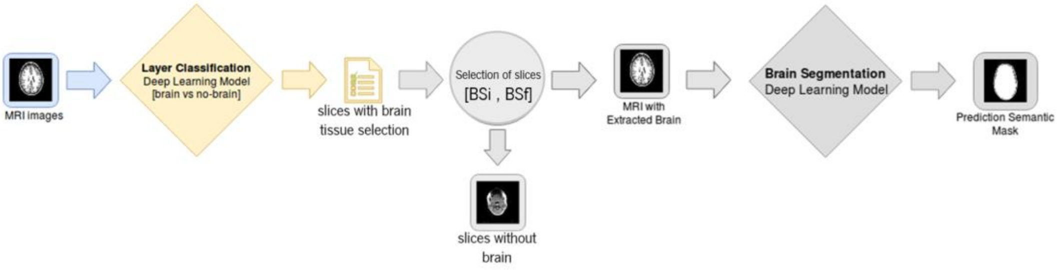


Figure 4.10 - Pipeline to extract the brain slices, highlighting the slice classification DL step

In a classification task, U-NET architecture was not appropriate since it was conceived to segmentation problems as a FCN. In this case it was only essential an CNN architecture using fully connected layers. At the end, a single number will classify a single input image.

A CNN approach for classification tasks is VGG-NET 16, which is inspired in *Keras* pre-trained DL model was enough to have successful results. This architecture contract and it ends with fully connected network layers.

The version created was simpler, containing less layers and feature maps and with some batch normalization and dropouts to prevent overfitting. While testing was possible to reach to the conclusion that the fully connected layers didn't need to be 4096 feature maps. The number of 512 feature maps was enough to get good results and the training was faster.

SELECTION OF THE BRAIN TISSUE IMAGES

It was necessary to ensure that no brain slice is left behind during the classification process, since the DL model could still have some classification errors. A simple algorithm was created to pre-process the output the classification slices' list (Figure 4.11). This pipeline's phase happens in the "selection of slices" step in the flow:

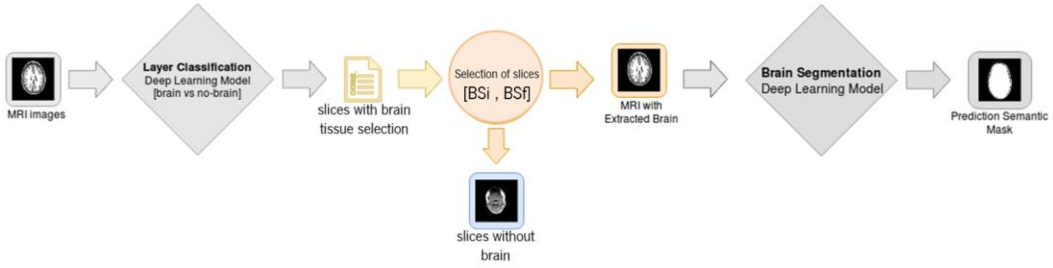


Figure 4.11 - Pipeline to extract the brain slices, highlighting the slices selection, after the DL classification

Axial brain slices are numbered sequentially, and the brain tissue is in a range of numbers. It is necessary to have that range of slices with brain tissue and ensure that in that range all the slices are used. It was also important to eliminate the isolated slices that were outside of the range of the slices with brain tissue. This selection will try to find the first and last brain slice of the range (BSi and BSf, respectively):

$$[BSi \, , BSf]$$

Figure 4.12 illustrates the effect of this selection. Figure on the left shows an example of an image and its respective classification, with some usual errors (colourful bar on the right) after going through a DL model that classifies as slice with brain tissue (green) or slice without brain tissue (red). After this pre-processing step, the isolated slices were eliminated, and the minimum (BSi) and maximum (BSf) axial number of slices with brain was selected.

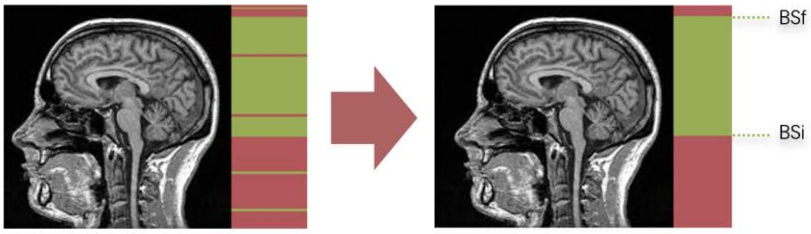


Figure 4.12 - Selection of the axial slices with brain tissue BE segmentation.

Figure 4.13 presents the pseudocode to find the BSi and BSf. The idea behind is create a function that takes the sequence of numbers classified with brain tissue, deletes the isolated slices and find the minimum and maximum of the remaining slices. An isolated slice is eliminated when there is no slice classified as with brain tissue in a space of 2 slices above and under.

```
def slices_selection(array):

        “““input: array with slices’ number previously classified as slices with brain tissue by DL model)
        output: integers BSi, BSf (first and last slice number where the brain is inserted) ”””
        slice_alone = [ ]  #isolated slices list
        for i in range(0, len(array)):
                if i<=2 and array[i + 1]!=array[i]+1 and array[i + 2]!=array[i]+2:     # if it is first two slices
                        slice_alone.append(i)
                elif i>2 and i<len(array)-2 and array[i + 1]!=array[i]+1 and array[i + 2]!=array[i]+2 and array[i - 1]!=array[i]-1
                and array[i - 2]!=array[i]-2:
                        slice_alone.append(i)
                elif i>=len(array)-2 and array[i - 1]!=array[i]-1 and array[i - 2]!=array[i]-2:     # if it is last two slices
                        slice_alone.append(i)
        array=np.delete(array, slice_alone, axis=0)   # remove isolated slices
        BSi = array[0]       # first slice of the range
        BSf = array[len(array)-1]       # last slice of the range

        return BSi, BSf
```

Figure 4.13 – Selection of slices Python code

4.3.2 Brain Extraction Segmentation using slices with brain tissue

After removing the slices without brain tissue, FCN U-NET architecture, also used in rats, was chosen (Figure 4.14). This DL model will only use brain's slices in training phase.

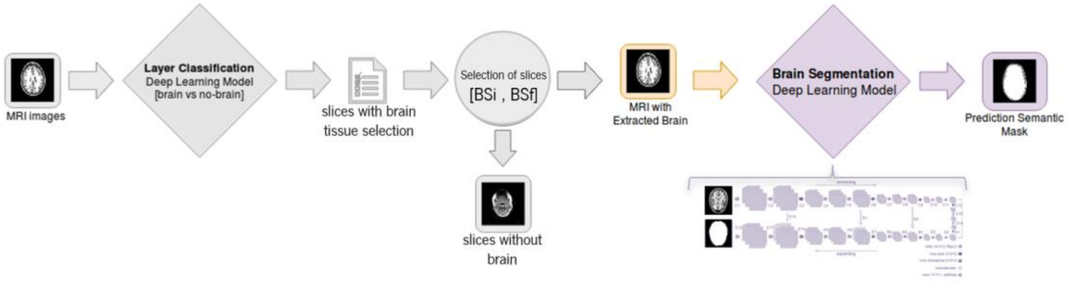


Figure 4.14 - Pipeline to extract the brain slices, highlighting the Brain Segmentation DL step using slices with brain tissue

The best architecture used is in appendix C. Table 4.1 presents the results of BE segmentation using the human dataset.

Table 4.1 - Final results of BE segmentation in Humans using two-dimensional convolutions approach

Accuracy	98.53 %
Sensibility	99.17 %
Specificity	97.44 %
DSC	**98.83 %**

4.4 Human Brain Semantic Segmentation

The brain semantic segmentation in humans started by using the pipeline with brain extraction explained earlier. Since in rats the semantic segmentation worked better using the pipeline that segmented first the brain followed by the semantic segmentation, it was decided to skip training the whole image and start with the image with the skull already extracted, to keep the same methodology in humans. This section describes the two different semantic segmentation human brain approaches (WGM segmentation and the WGMSC segmentation) and the experience with outliers.

4.4.1 WGM segmentation

The semantic segmentation of WM and GM followed the BE segmentation, shown in figure 4.15.

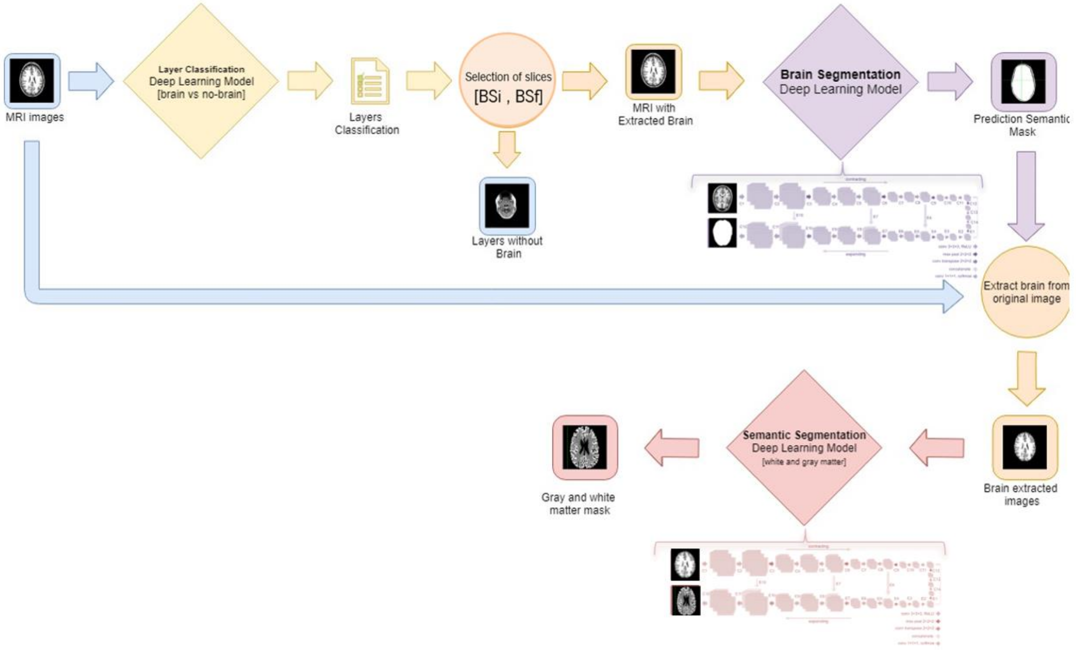


Figure 4.15 - Processing Pipeline for semantic segmentation of WM and GM

The best DL model is in Appendix C. The Table 4.2 represents the final results of the segmentation WGM in Humans using U-NET and two-dimensional approach.

Table 4.2 - Final results of WGM segmentation in Humans using two-dimensional convolutions approach

Accuracy	99.35 %
Sensibility	89.77 %
Specificity	99.62 %
DSC	**88.16 %**

Dice similarity coefficient was also calculated for the different tissues individually: The results for a two-dimensional convolutions approach using DSC were 91.59% for WM and 84,58% for GM.

4.4.2 WGMSC segmentation

The objective of the WGMSC segmentation was to segment 4 different tissues the WM, GM, subcortical tissues and cerebellum. The same pipeline was used, where the brain is extracted before the semantic segmentation. Figure 4.16 illustrates the pipeline used for WGMSC segmentation.

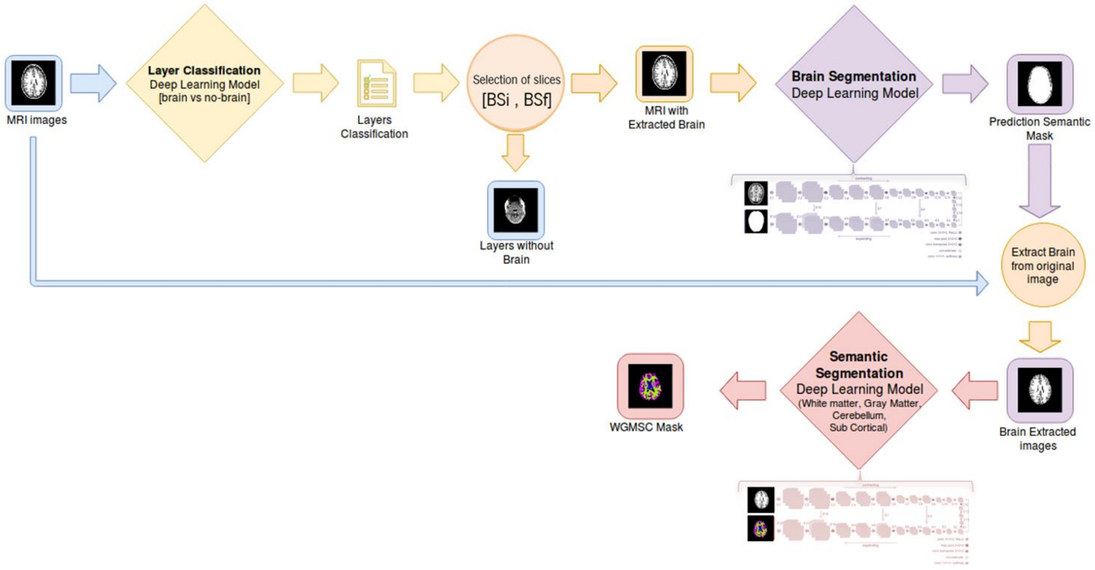


Figure 4.17 - Processing pipeline for semantic segmentation of WM and GM, subcortical tissues and cerebellum

The FCN U-NET based architecture and approach used in the previous segmentation of WM and GM was used in this four tissues' segmentation, but the results were not satisfiable. While training, looking at the values of sensitivity, it is possible to notice that the WM and GM are being trained correctly, while the subcortical and cerebellum areas are not recognized as a tissue to segment by the DL model. While the values of sensitivity of WM and GM were always increasing, the subcortical and cerebellum areas presented 0% of sensitivity. Since cerebellum and subcortical tissues are significantly smaller than the other two tissues (WM and GM), it creates a situation of unbalanced data. An analysis was carried out to understand the number of voxels in each of the areas. The Table 4.3 presents the average number of voxels for each area.

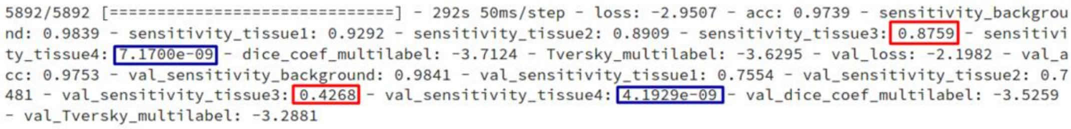
Table 4.3 - Mean of the number of voxels in each area of the training and validation set

Area	**Average number of voxels**
Background	15 734 539
WM	457 770
GM	446 000
Cerebellum	112 307
Subcortical tissues	26 600

The subcortical area is approximately 20 times smaller than gray or white matter. The cerebellum is 4 times smaller than white or gray matter. Also, the subcortical areas are only visible in 27% of slices that present brain tissue and the cerebellum nearly 47%.

The dice similarity coefficient loss revelled itself inefficient with this unbalance data situation. Because we are working with an unbalanced data set between the tissues, a new loss function is used, i.e. the asymmetric loss function [49], for each tissue mask which was introduced in the section 2.2.2 , equation number 4.

Using this loss function instead of the dice coefficient loss, the results relative to the cerebellum tissue improved. Figure 4.18 presents the values of training set and validation set during training of the DNN. Looking at the values, it is noticeable that the DL model was now able to detect and segment the cerebellum (tissue 3 in red) but it still failed when segmenting the subcortical tissues (tissue 4 in blue).



```
5892/5892 [==============================] - 292s 50ms/step - loss: -2.9507 - acc: 0.9739 - sensitivity_backgrou
nd: 0.9839 - sensitivity_tissue1: 0.9292 - sensitivity_tissue2: 0.8909 - sensitivity_tissue3: 0.8759 - sensitivi
ty_tissue4: 7.1700e-09 - dice_coef_multilabel: -3.7124 - Tversky_multilabel: -3.6295 - val_loss: -2.1982 - val_a
cc: 0.9753 - val_sensitivity_background: 0.9841 - val_sensitivity_tissue1: 0.7554 - val_sensitivity_tissue2: 0.7
481 - val_sensitivity_tissue3: 0.4268 - val_sensitivity_tissue4: 4.1929e-09 - val_dice_coef_multilabel: -3.5259
- val_Tversky_multilabel: -3.2881
```

Figure 4.18– Keras output after changing dice loss function to asymmetric loss function

The loss function used is giving the same amount of importance to all tissue masks when summing them all, distributing the weights equally to all the tissue masks. Not giving higher importance to the segmentation of the subcortical tissues, it is possibly an issue at this point.

All the tissues mask had the same importance when calculating the loss function:

- 100% to background
- 100% WM
- 100% GM
- 100% Cerebellum
- 100% Subcortical

The equation 10 describes the loss function for semantic segmentation.

$$Loss_{total} = \sum_{i=0}^{i=Number\ of\ tissue\ masks} Loss_{tissue\ number} \tag{10}$$

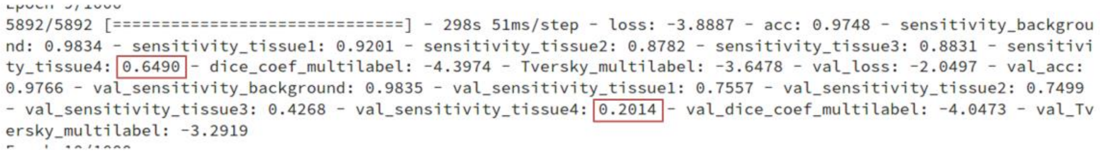
The asymmetric loss function was changed in a way that the tissue masks were given different importance. Since the background is the easiest task to perform, this tissue mask will receive the lowest weight. The best results were obtained were with the following weights:

- 4% Background
- 8% WM
- 8% GM
- 35% Cerebellum
- 45% Subcortical

The equation 11 represents the changes in the weights of the loss function.

$$Loss_{total} = 0.04 \times Loss_{background} + 0.08 \times Loss_{White\ matter} + 0.08 \times Loss_{Grey\ matter} + 0.35 \times Loss_{Cerebellum} + 0.45 \times Loss_{Subcortical\ tissues} \tag{11}$$

Although the subcortical tissue's sensitivity (Figure 4.19 inside the rectangles) increased greatly after the weights' changes in the loss function, it did not reach 65% with the training set and 20% in the validation set.



```
5892/5892 [==============================] - 298s 51ms/step - loss: -3.8887 - acc: 0.9748 - sensitivity_backgrou
nd: 0.9834 - sensitivity_tissue1: 0.9201 - sensitivity_tissue2: 0.8782 - sensitivity_tissue3: 0.8831 - sensitivi
ty_tissue4: 0.6490 - dice_coef_multilabel: -4.3974 - Tversky_multilabel: -3.6478 - val_loss: -2.0497 - val_acc:
0.9766 - val_sensitivity_background: 0.9835 - val_sensitivity_tissue1: 0.7557 - val_sensitivity_tissue2: 0.7499
- val_sensitivity_tissue3: 0.4268 - val_sensitivity_tissue4: 0.2014 - val_dice_coef_multilabel: -4.0473 - val_Tv
ersky_multilabel: -3.2919
```

Figure 4.19 - Training values visualization after changing weights in loss function

Table 4.4 presents the results of WGMSC segmentation using asymmetric loss function and weights' changes in the tissue maps. The results were not satisfiable.

Table 4.4 - Final results of WGMSC segmentation in Humans using two-dimensional convolutions approach

Accuracy	96.33 %
Sensibility	53.44 %
Specificity	99.02 %
DSC	**49.48 %**

4.5 Outlier influence in the results

In this section it is studied the reaction of the best DL models when are facing outliers' images. Outlier's images are significantly different cases from the images used for training the DNN. These images can be: e.g. cases of brain with cancer tissues, cases of older patient's cases, cases of brain with degenerated tissue, cases of child's brain. The generalization and segmentation accurately capability of the best DL model obtained until now will be evaluated adding outliers to the training dataset.

Aging causes changes in the brain's size, vascular and cognition. The natural degeneration of the brain makes it "shrinks" due to the changes at all levels from molecules to morphology [67]. The incidence of stroke, WM lesions also increase with age. There are also degenerative diseases that tend to affect the brain at an old age such as dementia which shrinks even more the brain [67]. In the medical imaging aspect, the brain's transformation can be quite extreme in Figure 4.20.

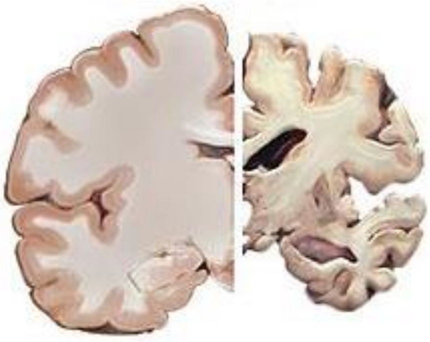


Figure 4.20 – Comparison between young brain and extreme alteration[68]

Working with elderly patient's images is nowadays a fundamental task, since this population group is increasing their ratio and number of elderly age group. Segmentation of an elderly brain can be a challenging task for traditional algorithms without AI due to their morphology changes comparing to an adult's case.

The best DL model of Brain extraction followed by WM and GM segmentation was tested in 61 elderly patient's cases and the results are exposed in Table 4.5.

Table 4.5 - Final results of BE and WGM segmentation in Humans using outliers

	BE	**WGM**
Accuracy	96.01 %	97.28 %
Sensibility	75.70 %	41.91 %
Specificity	97.87 %	98.56 %
DSC	**75.88 %**	**40.93 %**

Figure 4.21 shows what is happening visually in the brain's semantic segmentation with these new older cases. The following case shows an atrophy on the right side of the brain and ventricles bigger than the younger cases of the dataset. The algorithm fails specially in the BE segmentation. This DL model is not ready to face these cases and succeed. It failed due to big differences between younger patients' brain and elderly patients' brain. When having a poor result in BE phase, it consequently affects the WM and GM segmentation negatively:

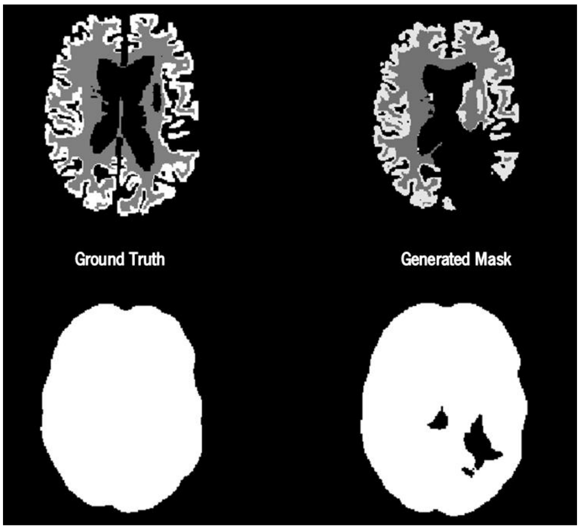


Figure 4.21 - Ground truth and the respective result of an older case with the best DL model

The model can generalize when it comes to younger cases, due to their similarity with the training images. But when it comes to elderly patient's brains the DL model is not ready to segment it accurately.

A new experiment was proceeded to understand if the DL model is capable to segment properly older brains when using these cases in the training phase. Some of these cases were added to the original training dataset, and a new DL model was created both in BE and WGM segmentation phases using these new cases. Using a transfer learning method, the weights saved before when creating the best DL model were to save time and get better results. Figure

4.22 illustrates the number of cases used in each phase, where the blue colour is the original dataset and the red colour is the new added outliers (elderly cases).

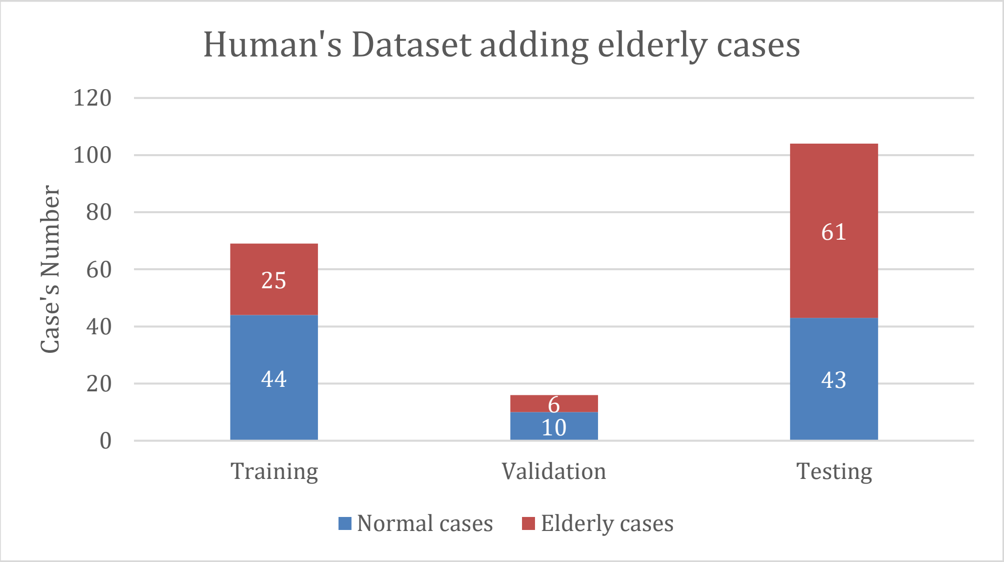


Figure 4.22 - New dataset to train DL model again adding elderly cases

When building the DL model with the best architecture and same parameters used before, the DL model couldn't train and detect the GM after adding the 25 elderly cases in the training set. This happens because the GM in elderly cases is normally a shrunk tissue, making it harder to segment. The weights' changes of the DSC in each segmentation area to overcome this obstacle are:

- 20% to background
- 40% WM
- 40% GM

Table 4.6 - Before and after testing images with new DL model trained with elderly patients, using 61 elderly for testing

	BE before	**BE after**	**WGM before**	**WGM after**
Accuracy	96.01 %	98.38 %	97.28 %	99.41 %
Sensibility	75.70 %	99.18 %	41.91 %	89.45 %
Specificity	97.87 %	97.72 %	98.56 %	99.64 %
DSC	**75.88 %**	**98.66 %**	**40.93 %**	**87.29 %**

Table 4.6 presents the results found in previous tables (before training with outlier cases) and the new results with the new trained model (after training with outlier cases). Table 4.7

presents the results of the young cases testing data set before and after training with outlier cases, to understand the impact of training with outliers in the testing of young cases:

Table 4.7 - Younger testing set results before and after using elderly patients in the training phase, using 43 young cases for testing

EVALUATION	**BE before**	**BE after**	**WGM before**	**WGM after**
Accuracy	98.53 %	98.48 %	99.36 %	99.43 %
Sensibility	99.18 %	99.18 %	89.77 %	90.90 %
Specificity	97.44 %	97.29 %	99.62 %	99.66 %
DSC	**98.88 %**	**98.81 %**	**88.16 %**	**89.26 %**

A detail discussion of these results is presented in the next chapter.

Chapter 5- Discussion and conclusion

Each of the experiments was carried out using similar procedures: They all start with a base solid architecture from accurate studies (CNN with fully connected VGG-NET for slice classification and FCN U-NET for segmentation). During training, loss graphs are analysed, and changes are made in underfitting or overfitting situations. In most of the trials a situation of overfitting (especially in the segmentation problems) was verified. In those cases, Dropout and batch normalization layers were used.

In some cases, there was also cases of underfitting, especially in multi-segmentation since it was a complex problem to solve with complex images as well. In those cases, it was added the pipelines explained in the last two chapters both in Rats and Humans that would segment first the brain and continue to a semantic segmentation. Figure 5.1 presents the process in a generalized way:

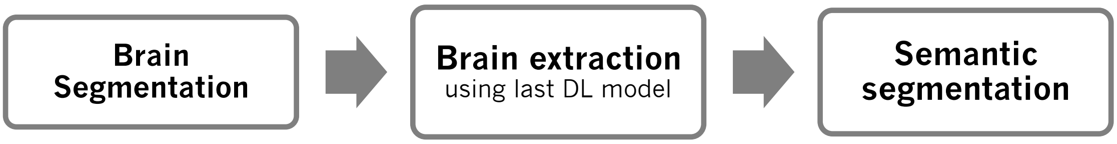


Figure 5.1 - Approach to prevent underfitting used in Rats and Humans semantic segmentation

All DNN used SGD algorithm with customize parameters such as a learning rate of 3×10^{-4}, a decay of 1.5×10^{-6} and a momentum of 0.9. The DNN used also the dice coefficient as loss function and a 5 case batch size for train. The Table 5.1 presents an example of five tests for Rat's BE segmentation. All the tests also entered in over-fitting after some training.

Table 5.1 - Different parameters used in different testing phases

	Best Loss-Function Validation set	**Best Loss- function training set**	**Changes in the architecture**
Test 1	0,0584	0,0180	Original U-NET
Test 2	0,0505	0,0066	Test 1 with batch Normalization
Test 3	0,0462	0,0230	Test 1 with batch Normalization and Dropouts of 0.1 after each CNN block
Test 4	0,0453	0,0268	Test 3 with stronger dropouts (0.2)

Test 5	0,0387	0,0262	Test 4 with even stronger dropouts (0.5 and 0.2)

The different tests were created to prevent overfitting. Even though in the loss-value gets worst in the training cases with the dropout slices added, it improved the values of validation set. This shows that with simple slice changes, better results can be achieved. The test 2,3,4 and 5 used weights from the first test, to make a faster training. The Figure 5.2 and Figure 5.3 represent the validation set loss values and the training set's loss values:

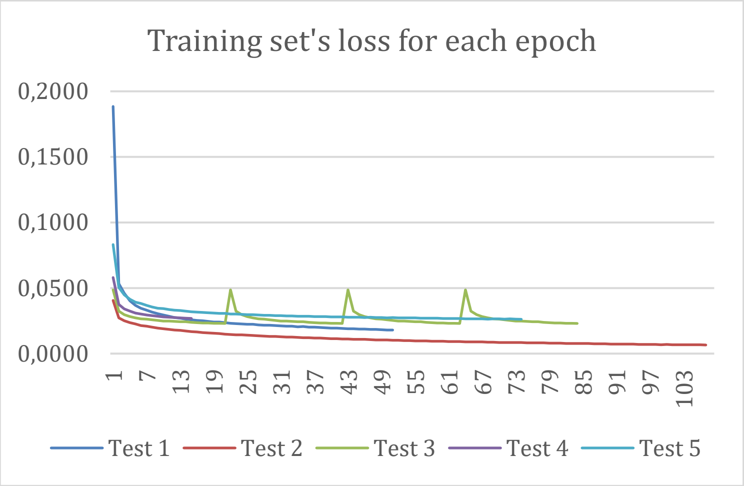


Figure 5.2 - Training set's loss function for each epoch

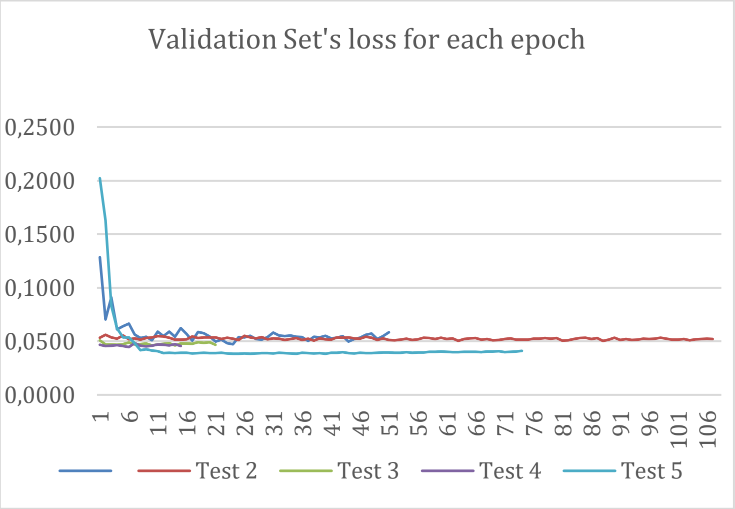


Figure 5.3 - Validation set's loss function for each epoch

According to the previous figures, in all the tests the validation values were always higher the training values. Also the training values stagnated as the epochs increased. Even though the best test in the training values was test 2, the same didn't happen for the validation set line.

The test that had better values in validation set was test 5, with batch normalization and higher dropouts.

5.1 Discussion on Rat's experiment

The Rats' results will be explored in the following sections, comparing all the DSC values.

5.1.1 BE and WGMCSF analysis

While training rats, it was possible to try a slice-by-slice (2D convolutions) and a volume (3D convolutions) DL approaches. The following figure show a chart with the values of DSC in the Rat's segmentation:

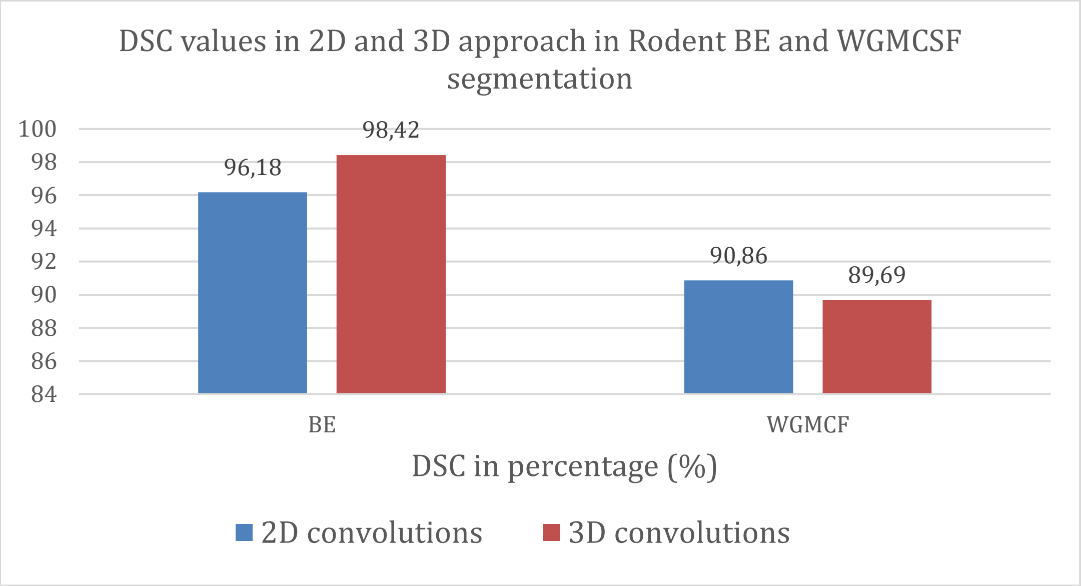


Figure 5.4 - Values of DSC to compare BE and WGMCF phase using 2D or 3D DL approach in Rats

Looking at the BE phase, the value of DSC in 3D approach was considerably better than 2D, which was expectable because the shape of the brain is continuous between slices meaning that the network would consider more information coming from the neighbor slices. When using semantic segmentation of WGMCSF, the values of the 3D convolutions approach were a slightly worst, but not significantly different to make the results that different (less than 1.5%). This result can be explained because of the less effort spent in tuning the 3D convolutions approach architecture.

5.2 Discussion on Human's experiment

The following graph shows the results of DSC in all the phases of segmentation: BE, WGM and WGMSC, presented previously in the Chapter 4- Humans Experiment.

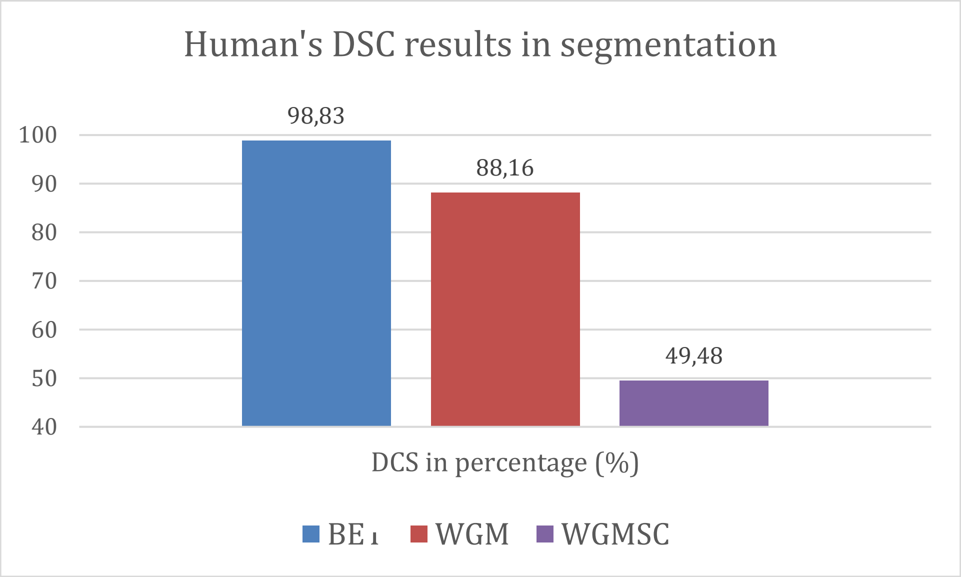


Figure 5.5 - Values of DSC to compare BE, WGM and WGMSC segmentation of DL in Humans

5.2.1 BE and WGM analysis

Looking at the values of DSC in BE and WGM phase, the results were accurate, and it had the same performance as rat's DL model. Even though most of the images were correctly segmented, some cases of the testing images had a low value of DSC than what it would be expected. It is the case of a young adult with abnormal larger ventricles, which is considered an outlier. The case of the Figure 5.6 had a DSC value of 86% on BE's phase, instead of the common 98% DSC.

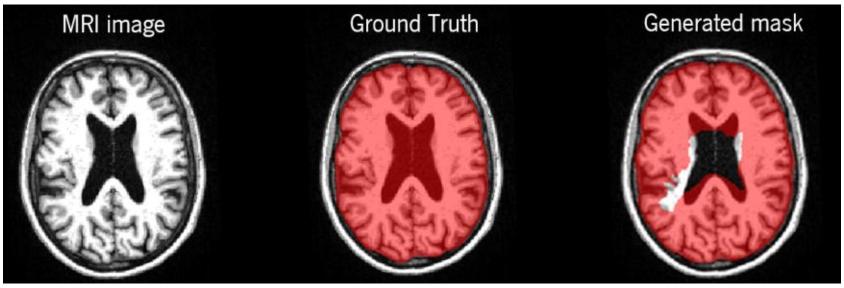


Figure 5.6 – Test case of abnormal big ventricles in a young case (~25 years old)

5.2.1.1 WGMSC analysis

Looking at the WGMSC results, even though the results didn't seem promising because of the value of sensibility in the subcortical areas when training (20% of validation sensibility in subcortical tissues), after looking at the final testing images, the results were visually better than expected. Segmentation errors found in the sensitivity value of the subcortical areas can be happening because the ground truth mask still has issues and the result mask can actually segment better than the ground truth. Since the subcortical area is a small area, it's natural that it's value of sensitivity was so low: a low number of wrongly classified voxels can be translated in a massive change of the sensitivity value. The Figure 5.7 shows an example of the subcortical areas (in blue) ground truth and generated mask problems:

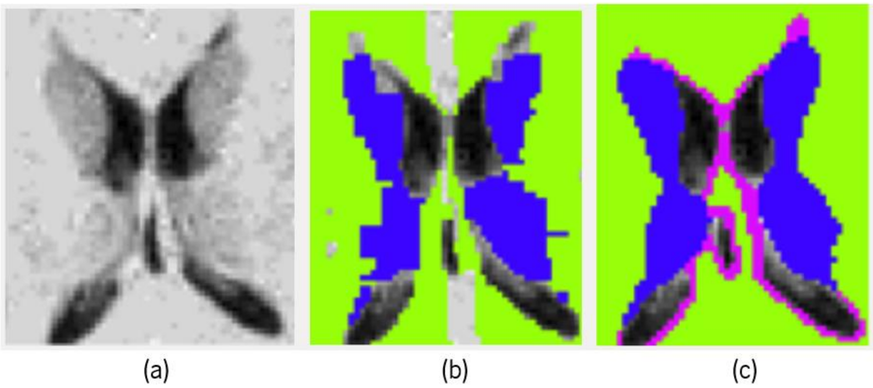


Figure 5.7 – Case of bad labelled data. (a) Subcortical tissues MRI. (b) Wrongly segmented ground truth. (c) Generated mask.

Even though there are still some issues in the WM and GM segmentation, it is possible to check visually that the subcortical tissues segmentation (in dark blue) had a better segmentation compared to the ground truth mask, proving the capability of the DL model to generalize.

The training was also not successful because it increases the difficulty of the segmentation task when their respective ground truth labels are not accurate enough.

5.2.2 Outliers' influence analysis

The following figure represents the values of DCS found previous in section 4.5.

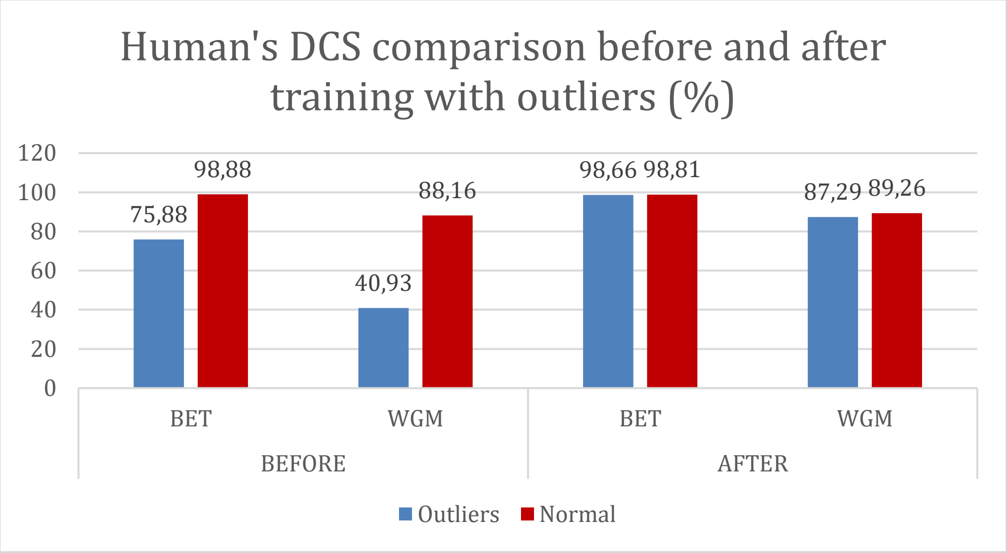


Figure 5.8 - Values of DSC to compare BE and WGM in outliers and normal cases before and after the new training

After training the brain binary segmentation and the WGM semantic segmentation phase, the remaining 61 images were tested once more to understand if the DL model would perform better after training with similar cases.

When adding 25 elderly images in the training phase, made a big difference in the final results. Looking at the DSC of elderly testing cases, after training with an additional set of 25 outliers' images the DL model improved from 75,88 % to 98,66% in the BE's segmentation phase and from 40,93 % to 87,29 % in the WGM segmentation phase. This is the proof that giving similar cases in the training phase, the DL model can learn them and generalize them with the same accurate results as the younger cases. The following figure shows the final generated masks before and after training to compare with the ground truth:

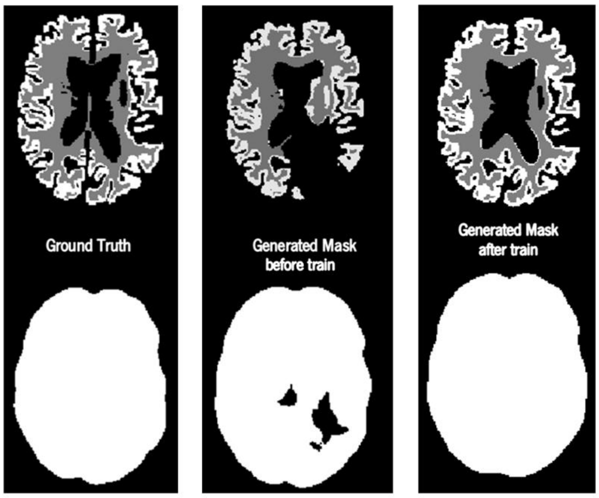


Figure 5.9 - Comparison between ground truth, generated mask before training with outliers and generated mask after training with outliers

It was also interesting to check if the DL model could maintain a good performance in the younger testing set, after the training with outliers' images i.e., could the use of elderly people images in the training phase make the DL model less accurate when facing younger MRI brains?

When adding 25 elderly cases not only the DL model was able to generalize with the older patients dataset, but also improve the performance of BE and WGM segmentation for young cases.

The young cases results improved after training with elderly cases. One of the cases referred before of younger brains, was not correctly segmented due to their overly large ventricles. Figure 5.10 presents the BE of this case before and after training with elderly cases.

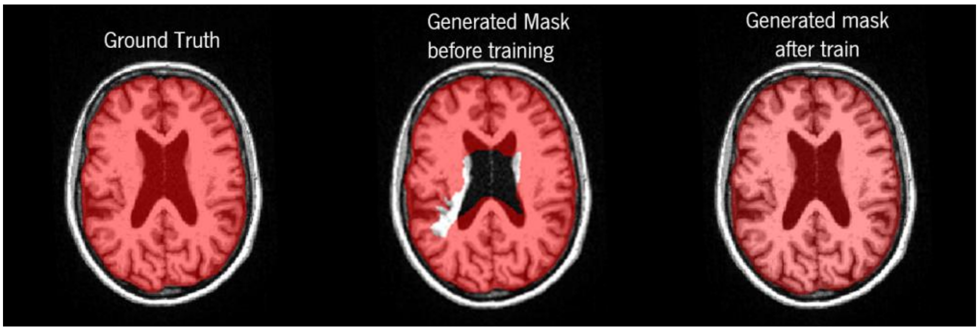


Figure 5.10 - BE of a younger brain with large ventricles before and after training with outliers

Since the elderly cases have larger ventricles due to their morphologic changes of aging process, it is logic that after training with these outliers, the DL model was able to generalize on these young cases, take this particular characteristic from elderly brains (larger ventricles) and segment correctly this time.

5.3 Humans and rats DL model performance

Segmenting a tissue using magnetic resonance images can be more complex in some species than in others. Since this study laid in exploring two distinctly different species: humans and rats, it is possible to conclude which one was more complex and caused more struggles when training and segmenting. The following table shows some of the best results found both in Rats and Humans, already presented in the section 5.1 and 5.2:

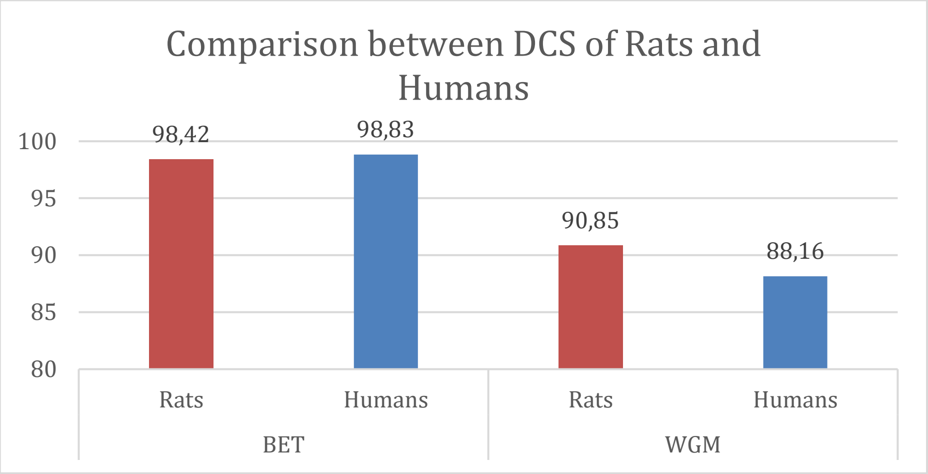


Figure 5.11 - Comparison between DSC values of Rats and Humans experiment

Looking at Figure 5.11 it is possible to understand that the specie of rats outperformed the specie of Humans in the final main objective: the semantic segmentation. That could happen due to: Rat's dataset was bigger than the human's dataset, which could explain why the model performed better. Larger amount of data in training phase produces better models. The images also had smaller resolution and looked less complex than humans. Humans had to go through an additional DL model to remove non-brain slices, possibly making the DL model less accurate than rats.

5.4 Analyzing the effect of U-NET architecture changes

To achieve better results some modifications of the U-NET architecture were tested. Inspecting the initial results and feature maps, it was visible that some of the feature maps had very low resolution and information. It was hypothesized that the feature maps were losing important information when contracting after successive max-pooling slices. Figure 5.12 shows the feature maps in the slice 20 (where they had a resolution of 4 x 4).

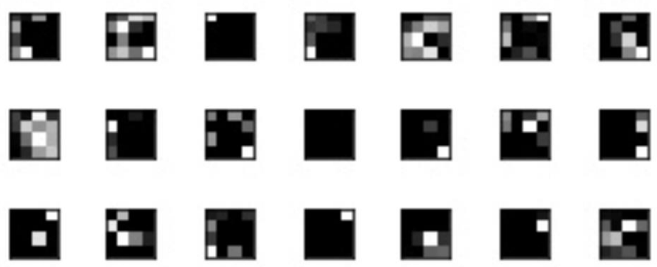


Figure 5.12 - Feature maps in slice number 20 of the U-NET architecture (4 x 4)

Considering this, it was created an alternative architecture where it was removed one contracting block and one expanding block to prevent the feature maps significantly reduction of resolution, as it is shown in figure 5.13. The removed blocks are in gray:

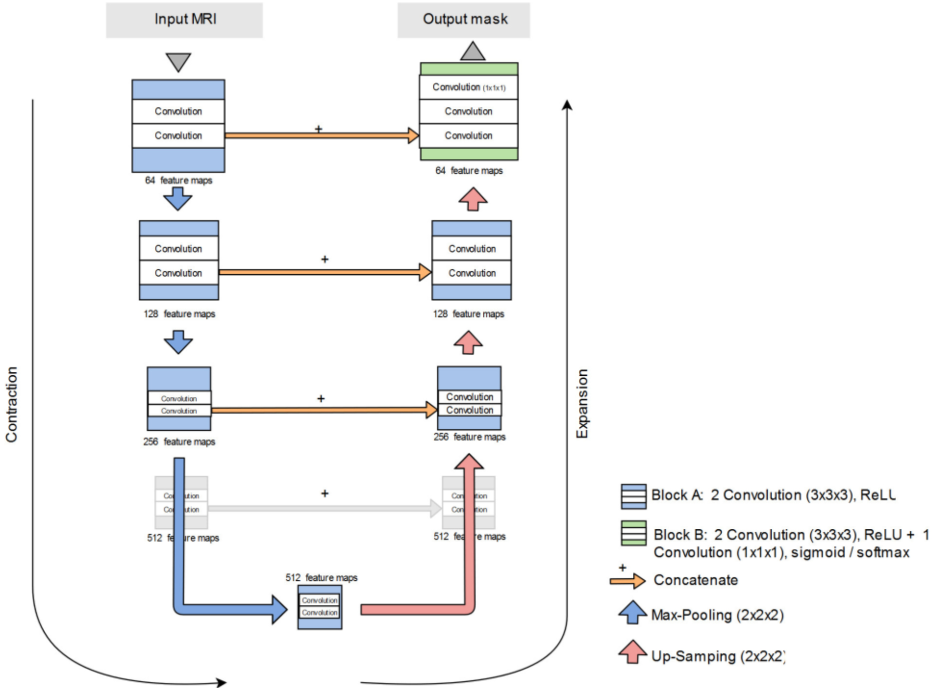


Figure 5.13 - Changed U-NET for a simpler approach

After testing with this new architecture, it was noticeable that the results were not as good, explaining that sometimes even if the feature maps seem to have low resolution, it doesn't necessarily mean that it could interfere in a negative way in the final result.

Another noticeable feature is that in the center of the U-NET the number of feature maps reaches 1024. This could also be preventing better results in training phase where the DL model could be losing some capacity to understand some important features due to the large amount of information. One possibility to improve this was to try to reduce the number of feature maps in each layer to half. The feature maps were changed from (64→128→256→512→1024) to (32→64→128→256→512).

This approach provided a faster training, but the results didn't reach better results than before. With that in mind, having a big amount of feature maps can help to improve the results.

5.5 Feature maps analysis

Visualizing the feature maps during training can be a way to understand if the deep learning models are learning correctly the important features from the MRI images during training. The feature maps shown in this section are created by the best deep learning model of semantic segmentation both in rats and humans in different layers of the architecture U-NET.

CONTRACTING PHASE

These three images represent the first created feature maps of the U-NET. It shows what happens after 2 convolutions of a single MRI images, followed by a Max-pooling. The convolutional layers make several different images using different filters, while the max pooling layer reduces the resolution, but keeps the important information.

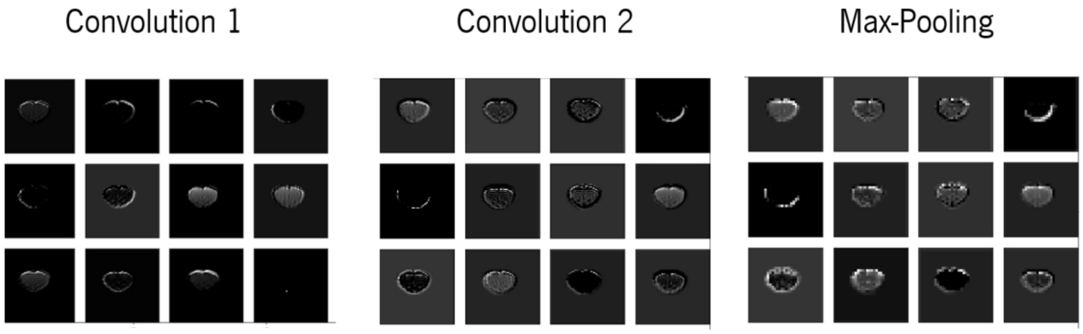


Figure 5.14 -Example of final feature maps of the first convolutional and max pooling layers using Rat's MRI

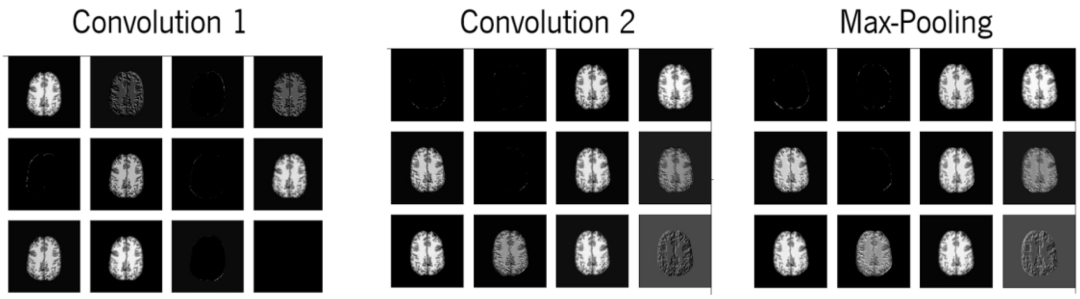


Figure 5.15 - Example of final feature maps of the first convolutional and max pooling layers using human's MRI

The previous feature maps were created in the phase of the DL U-NET:

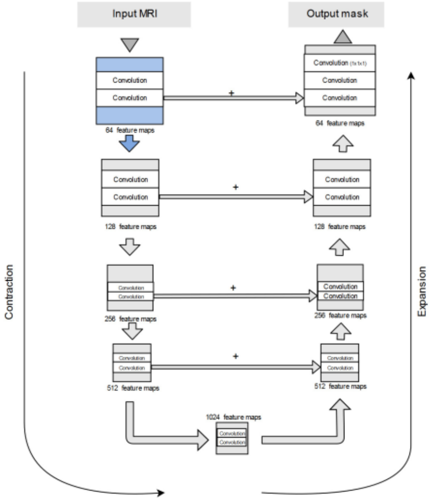


Figure 5.16 - Location in the U-NET of the feature maps created in the first CNN block

After some convolutional layers and max-pooling, the feature maps are starting to show lower resolution and compact results. Figure 5.17, Figure 5.18, Figure 5.19 shows the result of the feature map's second, third and fourth CNN block (after max-pooling layer):

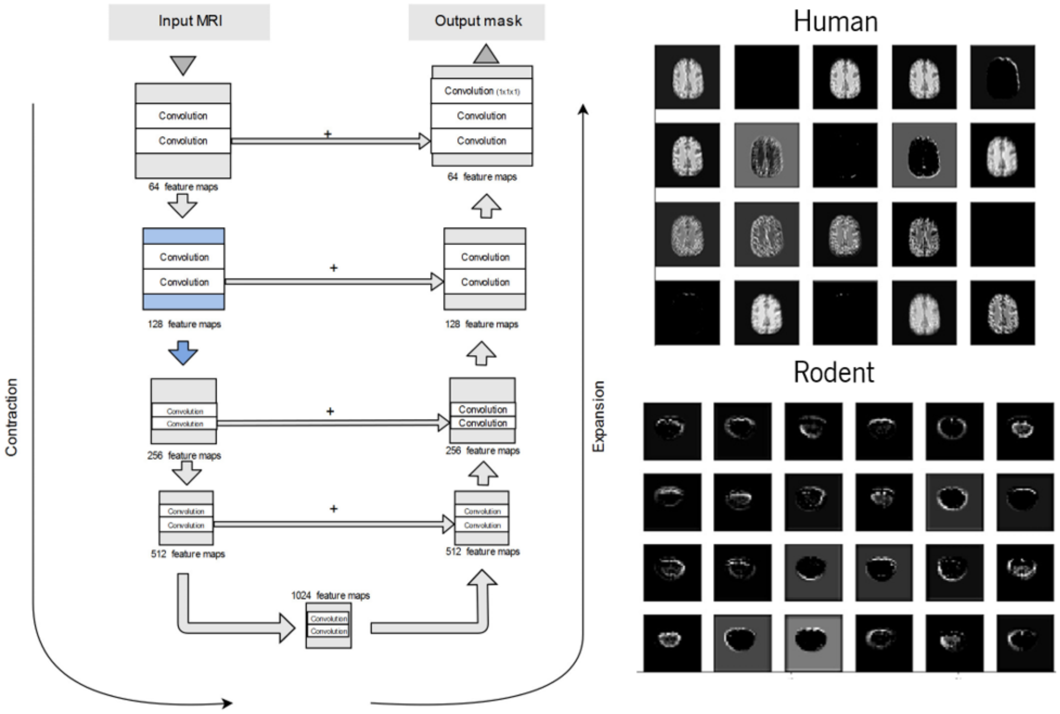


Figure 5.17 – Feature maps from second CNN block from U-NET

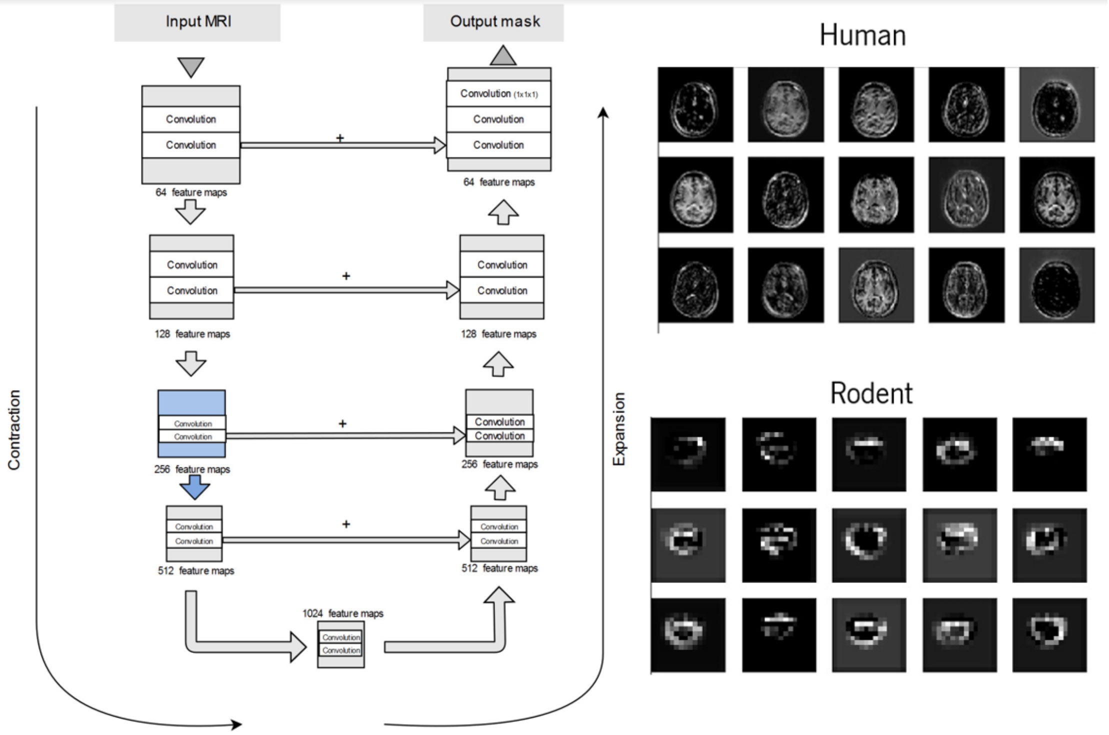


Figure 5.18 - Feature maps from third CNN block from U-NET

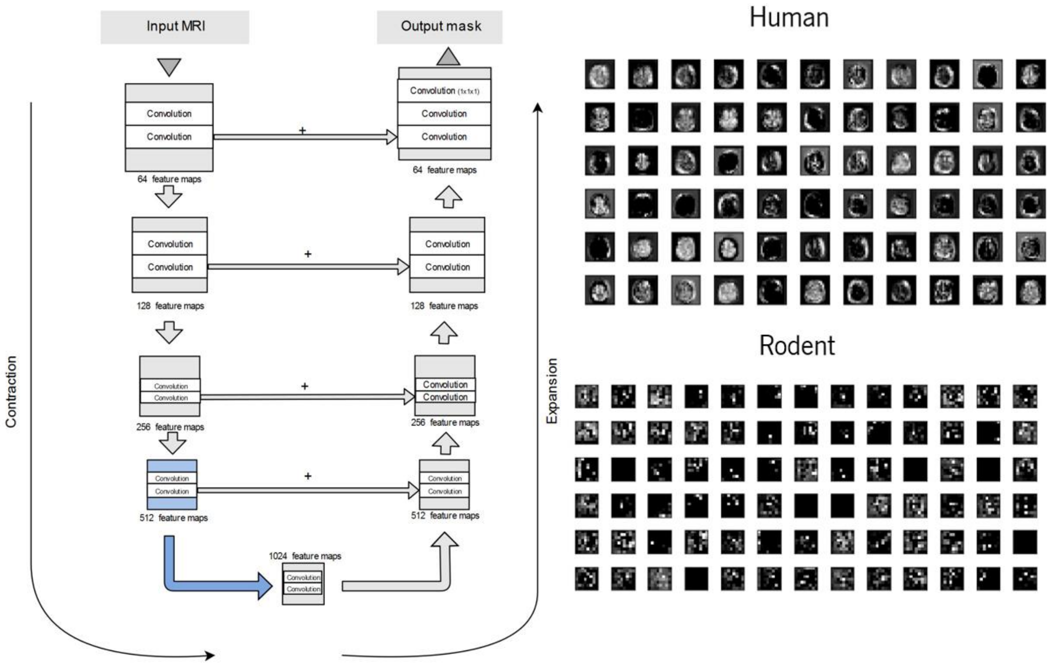


Figure 5.19 - Feature maps of fourth block from U-NET CNN both in humans and rats

It is possible to understand from the previous images that the deeper we go through the CNN layers, the less resolution the images have, the more compact is the information and that the more difference there is between the feature maps.

CENTER OF THE U-NET

The center of the U-NET is when the network reached the final phase of the contracting phase and it’s preparing for the expansion phase. The results of the feature maps reveal small and low-resolution image.

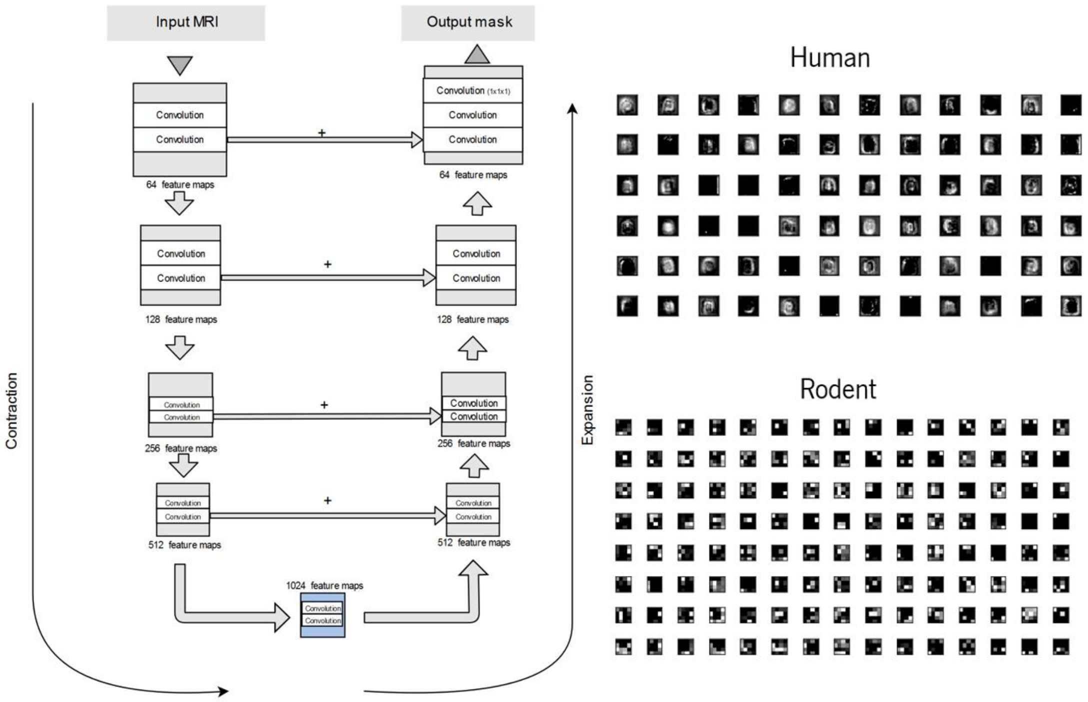


Figure 5.20 - U-NET center feature maps of both humans and rats

EXPANSION PHASE

In the expansion phase, there is the merge block: Some of the previous images are added to the resulting feature maps, a convolutional process is done and an Upsampling happens. The following images show visually what happens some of the merge blocks:

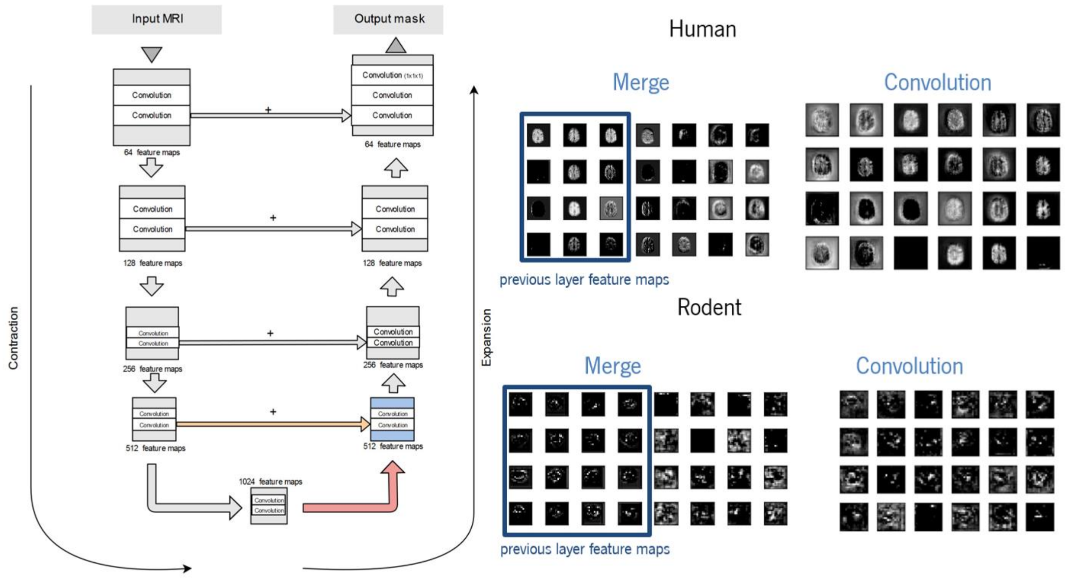


Figure 5.21 - Feature maps from the first merge block in Rats and Humans

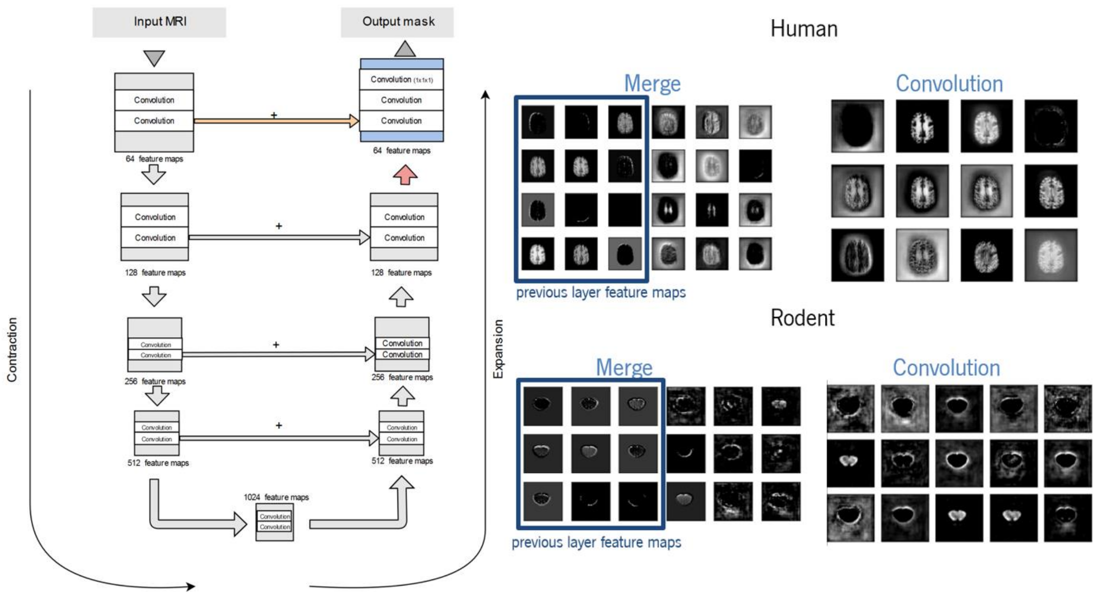


Figure 5.22 - Feature maps from final merge block in Rats and Humans

The convolutional layers in the merge block create a mixture of the previous feature maps and the current feature maps. This combination allows a bigger resolution for the final masks with all important feature remaining. At the end, the DL algorithm will create the final generated masks:

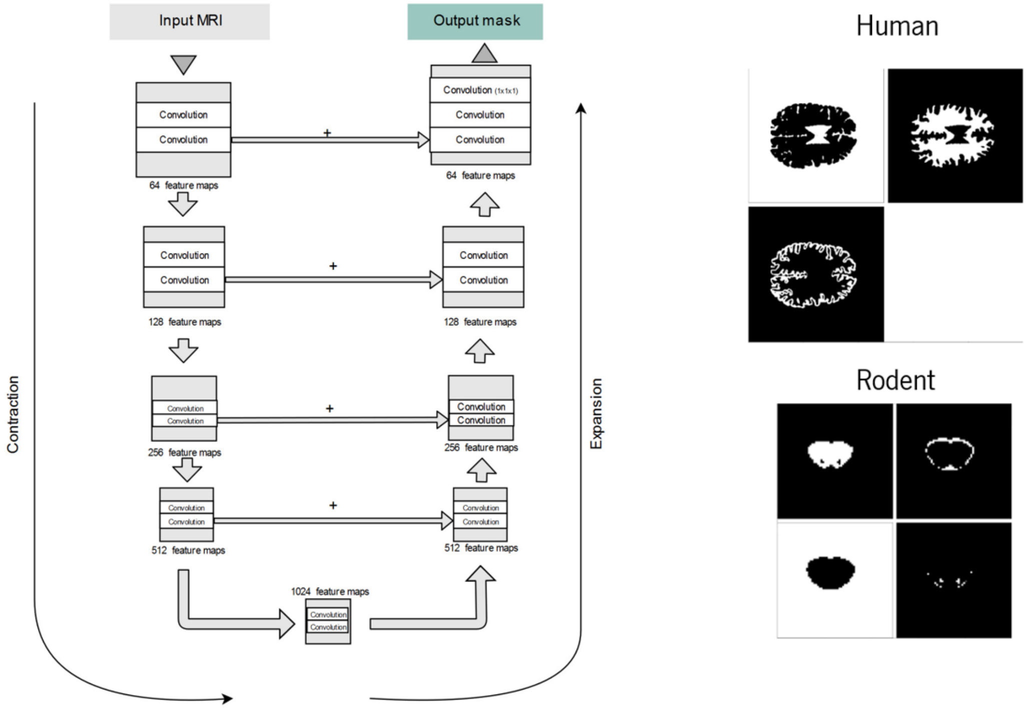


Figure 5.23 - Final generated masks in Humans and Rats

5.6 Conclusions

This work aimed at evaluating the viability of applying deep learning in the semantic segmentation of brain tissues in rat MRI and developing a new tool to assist in the processing of this kind of data, a type of resource currently missing in the field.

Overall the FCN (U-NET based) architecture presented good results for the main objective of this study: Brain Semantic segmentation in rats and also in humans MRI datasets. These results are comparable to similar studies [69] that applied this architecture in human MRI datasets. Importantly some of these works [25][26] used images with different contrasts, however we were able to achieve similar results using only T1 weighted images. This finding raises the question whether better results could have been achieved if more information was used, or if a single T1 weighted image is enough.

Some changes in the original U-NET architecture were tested in this study. Reducing the number of layers to prevent the feature maps to contract extremely didn't improve the results. Allowing us to conclude that even if the feature maps seem to have low resolution and have lost important information, it doesn't necessarily mean that it could interfere in a negative way in the final result. Another change in the architecture was reducing the number of feature maps to half the size. It allowed faster training of the network, but the results were worst suggesting that having a larger amount of feature maps can help improve the results. Of course, it is not known if the same conclusion can be extended to further doubling the number of feature maps to 2048 as the resources to run such an experiment are not available to us.

Before starting training any DL model, it is essential to have accurate and well segmented data. In this dissertation some of the first problems faced was due to the lack of correct masks. Accurate labels matter and interfere in the success of the training process, as it was proved when trying to segment the subcortical areas where the results showed better generated masks than the ground truth mask.

The transition from applying the architecture to a rodent dataset to a human dataset was straightforward, meaning that if we have a successful deep learning model for brain segmentation in humans' MRI, by transfer learning, it is possible that it also will have good results for similar applications in rodents' MRI. This opens new avenues of research in adapting existing pipelines for human datasets to rodents using deep learning, which may greatly accelerate the development of the field. The deep learning model presented better performance and results when using rat's dataset, compared to the human's dataset, likely because the images are smaller in size and the features are less complex. We also must consider that the rodent dataset was acquired in an ultra-high field scanner (11.7T) while the human dataset was on a low field (1.5T), which translates into a better signal-to-noise ratio in the human dataset.

In the rodent dataset it was possible to apply the U-NET with 2D convolutions approach (slice by slice) and a 3D convolutions approach (volume). Looking at the brain extraction segmentation, the value of DSC in 3D approach was considerably better than 2D, which was expectable because the shape of the brain is continuous between slices meaning that the network would consider more information coming from the neighbor slices. When using semantic segmentation of WGMCSF, the values of the 3D convolutions approach were a slightly worst, but not significantly different to make the results that different (less than 1.5%). This result can be explained because of the less amount of time spent in tuning the architecture compared to the 2D convolutions approach.

Changing the weights of the tissue maps of each cerebral structure in the loss function for a more accurate semantic segmentation was also an essential process to have accurate results. Smallest sized tissues are often harder to segment in semantic segmentation. Giving more weight to these tissues in the loss function, allowed the model in the learning process to segment them more easily and correctly, since they were more controlled than usual.

It was also possible to understand that adding outliers to the training data set helps the DL model not only to generalize with elderly brains, but to perform better with younger subject's brain. After training with these outliers, the model was able to correctly segment the large ventricles' cases, since it generalized with the elderly brain's cases which have larger ventricles due to morphologic changes from aging. With more variability in the training dataset, we obtain better results and generalization proprieties.

At the end, the results found using U-NET in rats' MRI for a 2D convolutions approach using DSC were 94.65 % for WM, 91.03% for GM and 76.89 % for cerebrospinal fluid. Using the 3D convolutions approach, the results using DSC found are 93.81 % for WM, 89.69 % for GM and

74.68 % for cerebrospinal fluid. The results using humans' MRI using DSC were 91.59% for WM and 84,58% for GM.

Normally when using traditional algorithms for brain segmentation, several processing iterations of interactions from the researcher to obtain satisfactory results may be necessary. It is interesting to note that the majority (~95%) of the generated masks were correct on the first try. Some generated masks also seemed visually to have a better segmentation than ground truth masks. In conclusion, it seems that the model behaves better than the traditional algorithms both in Rats and Humans.

FUTURE WORK

To improve the generalization ability of the model, one interesting approach to test would be to put dropout layers in the first layer of the model. Regularizations by noise is a common technique to improve generalization performance [70].

In the future it will be also interesting to understand if applying the same approach to a 3T human data-set can yield better results, since the images have more resolution than the dataset used.

Some thoughts on how to improve the WGMSC segmentation was: The more classes we need to segment, the more difficult it is for the algorithm to get it right, the smaller the areas, the harder it is for the DL model to segment in a semantic segmentation task. For future work it would be interesting to separate each tissue and run a binary segmentation DL algorithm of the subcortical tissues, for example to see if the DL model could get more accurate results even with a poor ground truth mask. The ground truth masks for WGMSC weren't also accurate enough and the generated mask predicted better than the ground truth. For better results, the ground truth would have to be reviewed and manually corrected.

Working with 2D slices can sometimes be a problem when the tissue does not exist in several slices, as it was the case of the subcortical tissues. If the algorithm located first the tissue, it would be easier to train and get better results. It could be also possible to use a 3D convolutions approach instead of slice by slice, after the localization of the tissue. The images would be smaller, and less computer resources would be needed for a 3D volume approach.

Another interesting future project that was not explored in this dissertation was the segmentation of three different tissues from the subcortical area. The subcortical area is a section that can be very tricky to segment and dubious process. One of the biggest problems is that the three target tissues to segment were considerably smaller than the WM and GM, so the DL model created didn't have the capacity to segment them and entered in underfitting. To overcome this issue, it would be necessary to create a pipeline that would first create a bounding box around the target area, and then segment the 3 tissues:

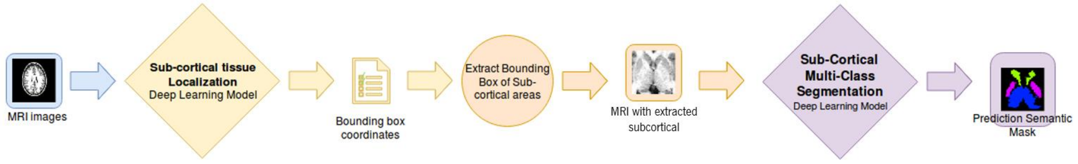


Figure 5.24 - Pipeline for subcortical tissues segmentation

References

[1] R. C. K. Elliot J. Gindis, "Spotlight on: Biomedical Engineering," *Up Run. with AutoCAD 2018*, pp. 445–447, 2018.

[2] J. Lucas, "What is Biomedical Engineering?," 2014. [Online]. Available: https://www.livescience.com/48001-biomedical-engineering.html. [Accessed: 10-Jan-2018].

[3] R. Haux, "Medical informatics: Past, present, future," *Int. J. Med. Inform.*, vol. 79, no. 9, pp. 599–610, 2010.

[4] W. M. Wells, "Medical Image Analysis – past, present, and future," *Med. Image Anal.*, vol. 33, pp. 1339–1351, 2016.

[5] E. Gibson *et al.*, "NiftyNet: a deep-learning platform for medical imaging," *Comput. Methods Programs Biomed.*, vol. 158, pp. 113–122, 2017.

[6] "Medical News Today - MRI." [Online]. Available: https://www.medicalnewstoday.com/articles/146309.php. [Accessed: 10-Aug-2018].

[7] J. M. Soares *et al.*, "Stress Impact on Resting State Brain Networks," *PLoS One*, vol. 8, no. 6, pp. 1–9, 2013.

[8] R. Magalhães *et al.*, "White matter changes in microstructure associated with a maladaptive response to stress in rats," *Transl. Psychiatry*, vol. 7, no. 1, p. e1009, 2017.

[9] ScienceDaily, "Computer Vision." [Online]. Available: https://www.sciencedaily.com/terms/computer_vision.htm. [Accessed: 08-Jan-2018].

[10] Techopedia, "Computer Vision," 2017. [Online]. Available: https://www.techopedia.com/definition/32309/computer-vision. [Accessed: 10-Jan-2018].

[11] M. Leo, G. Medioni, M. Trivedi, T. Kanade, and G. M. Farinella, "Computer vision for assistive technologies," *Comput. Vis. Image Underst.*, vol. 154, pp. 1–15, 2017.

[12] G. Litjens *et al.*, "A Survey on Deep Learning in Medical Image Analysis," no. 1995,

2017.

[13] "What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?" [Online]. Available: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/. [Accessed: 09-Dec-2017].

[14] J.-G. Lee *et al.*, "Deep Learning in Medical Imaging: General Overview," *Korean J. Radiol.*, vol. 18, no. 4, p. 570, 2017.

[15] M. I. Razzak, S. Naz, and A. Zaib, "Deep Learning for Medical Image Processing: Overview, Challenges and Future," *CoRR*, vol. 1704.06825, pp. 1–30, 2017.

[16] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, no. December 2012, pp. 60–88, 2017.

[17] Z. Akkus, A. Galimzianova, A. Hoogi, D. L. Rubin, and B. J. Erickson, "Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions," *J. Digit. Imaging*, vol. 30, no. 4, pp. 449–459, 2017.

[18] L. Zheng, S. Biedermann, J. Hesser, a Sartorius, and W. Weber-Fahr, "A rat brain template of class distribution maps based on diffeomorphic image registration," *Proc. 17th Sci. Meet. Int. Soc. Magn. Reson. Med.*, vol. 32, no. 2, p. 2897, 2009.

[19] I. Oguz, H. Zhang, A. Rumple, and M. Sonka, "RATS: Rapid Automatic Tissue Segmentation in rodent brain MRI," *J. Neurosci. Methods*, vol. 221, pp. 175–182, 2014.

[20] "FreeSurfer." [Online]. Available: https://surfer.nmr.mgh.harvard.edu/. [Accessed: 15-Jul-1BC].

[21] K. Van Leemput, F. Maes, D. Vandermeulen, and P. Suetens, "Automated model-based bias field correction of MR images of the brain Automated Model-Based Bias Field Correction of MR Images of the Brain," vol. 18, no. January, pp. 885–896, 1999.

[22] S. M. Smith, "Fast robust automated brain extraction," *Hum. Brain Mapp.*, vol. 17, no. 3, pp. 143–155, 2002.

[23] Y. Zhang, M. Brady, and S. Smith, "Segmentation of brain MR images through a hidden

Markov random field model and the expectation-maximization algorithm," *IEEE Trans. Med. Imaging*, vol. 20, no. 1, pp. 45–57, 2001.

[24] J. Liu *et al.*, "Applications of deep learning to MRI images: A survey," *Big Data Min. Anal.*, vol. 1, no. 1, pp. 1–18, 2018.

[25] W. Zhang *et al.*, "Deep convolutional neural networks for multi-modality isointense infant brain image segmentation," *Neuroimage*, vol. 108, pp. 214–224, 2015.

[26] P. Moeskops *et al.*, "Evaluation of a deep learning approach for the segmentation of brain tissues and white matter hyperintensities of presumed vascular origin in MRI," *NeuroImage Clin.*, vol. 17, no. October 2017, pp. 251–262, 2018.

[27] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," 2016.

[28] J. Dolz, C. Desrosiers, and I. Ben Ayed, "3D fully convolutional networks for subcortical segmentation in MRI: A large-scale study," *Neuroimage*, vol. 170, pp. 456–470, 2018.

[29] S. Korolev, A. Safiullin, M. Belyaev, and Y. Dodonova, "Residual and plain convolutional neural networks for 3D brain MRI classification," *Proc. - Int. Symp. Biomed. Imaging*, pp. 835–838, 2017.

[30] T. Huang *et al.*, "AGE ESTIMATION FROM BRAIN MRI IMAGES USING DEEP LEARNING Department of Computer Science , National Tsing-Hua University , Taiwan Graduate School of Information Science , Tohoku University , Japan South China University of Technology , China Institute of D," *Conf. 2017 IEEE 14th Int. Symp. Biomed.*, vol. 2, no. 1, pp. 849–852, 2017.

[31] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," pp. 1–14, 2014.

[32] O. Ronneberger, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 2017. [Online]. Available: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/. [Accessed: 11-Dec-2017].

[33] A. Gholipour and S. Salehi, "Tversky Loss Function for Image Segmentation using 3D Fully Convolutional Deep Networks," vol. 10541, no. September, 2017.

[34] K. FRISTON, J. ASHBURNER, S. KIEBEL, T. NICHOLS, and W. PENNY, *Statistical Parametric Mapping - The Analysis of Functional Brain Images*. 2007.

[35] P. A. Valdés-Hernández *et al.*, "An in vivo MRI Template Set for Morphometry, Tissue Segmentation, and fMRI Localization in Rats," *Front. Neuroinform.*, vol. 5, p. 26, Nov. 2011.

[36] A. Berger, "Magnetic resonance imaging," *Magn. Vol Ii Mater. Appl.*, vol. 324, no. January, pp. 355–386, 2002.

[37] "Magnetic Resonance Imaging (MRI)," *National Institute of Biomedical Imaging and Bioengineering*, 2017. [Online]. Available: https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri. [Accessed: 04-Feb-2018].

[38] L. Deng and D. Yu, "Deep Learning: Methods and Applications," NOW Publishers, 2014.

[39] Skymind, "Introduction to Deep Neural Networks," 2017. [Online]. Available: https://deeplearning4j.org/neuralnet-overview. [Accessed: 01-Jul-2018].

[40] NVIDIA, "Deep Learning," 2018. [Online]. Available: https://developer.nvidia.com/deep-learning. [Accessed: 25-Jul-2018].

[41] S. Of and T. H. E. Brain, "SCIENCE OF THE BRAIN Neuroscience : the Science of the Brain."

[42] L. Araujo, "Artificial Intelligence." [Online]. Available: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/neural_networks.html. [Accessed: 02-Jul-1BC].

[43] N. Ketkar, *Deep Learning with Python*. Apress, 2017.

[44] "CS231 Convolutional Neural Networks for Visual Recognition." [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed: 20-May-2018].

[45] O. WetWare, "Action Potentials," 2018. [Online]. Available: https://openwetware.org/wiki/BIO254:AP. [Accessed: 30-Oct-2018].

[46] S. Sharma, "Activation Functions: Neural Networks." [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed: 20-Oct-2018].

[47] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*. 2017.

[48] K. H. Zou *et al.*, "Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index," *Acad. Radiol.*, vol. 11, no. 2, pp. 178–189, 2004.

[49] S. R. Hashemi, S. S. M. Salehi, D. Erdogmus, S. P. Prabhu, S. K. Warfield, and A. Gholipour, "Asymmetric Similarity Loss Function to Balance Precision and Recall in Highly Unbalanced Deep Medical Image Segmentation," no. June, 2018.

[50] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, p. 533, Oct. 1986.

[51] H. Robbins and S. Monro, "A Stochastic Approximation Method," *Ann. Math. Stat.*, vol. 22, pp. 400–407, 1951.

[52] D. P. Kingma and J. Ba, "Adam: {A} Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014.

[53] M. D. Zeiler, "{ADADELTA:} An Adaptive Learning Rate Method," *CoRR*, vol. abs/1212.5701, 2012.

[54] Y. N. Dauphin, H. Vries, J. Chung, and Y. Bengio, "RMSProp and equilibrated adaptive learning rates for non-convex optimization," *arXiv*, vol. 35, 2015.

[55] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.

[56] M. Havaei *et al.*, "Brain tumor segmentation with Deep Neural Networks," *Med. Image Anal.*, vol. 35, pp. 18–31, Jan. 2017.

[57] S. Edu, "Classification in Deep Learning." [Online]. Available: http://cs231n.github.io/classification/. [Accessed: 11-Jul-2018].

[58] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation."

[59] and P. M. M. S.M. Smith, M. Jenkinson, M.W. Woolrich, C.F. Beckmann, T.E.J. Behrens, H. Johansen-Berg, P.R. Bannister, M. De Luca, I. Drobnjak, D.E. Flitney, R. Niazy, J. Saunders, J. Vickers, Y. Zhang, N. De Stefano, J.M. Brady, "FSL," 2016.

[Online]. Available: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/. [Accessed: 20-Sep-2018].

[60] OpenCV, “Histogram Equalization.” [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html. [Accessed: 01-Jul-2018].

[61] A. Coste, “MRI Histogram.” [Online]. Available: http://www.sci.utah.edu/~acoste/uou/Image/project1/Arthur_COSTE_Project_1_report.html. [Accessed: 30-Oct-2018].

[62] Keras, “Keras Documentation Metrics.” [Online]. Available: https://github.com/keras-team/keras/blob/c2e36f369b411ad1d0a40ac096fe35f73b9dffd3/keras/metrics.py#L13. [Accessed: 04-May-2018].

[63] Tensorflow, “Tensorboard: Visualizing Learning,” 2018. [Online]. Available: https://www.tensorflow.org/guide/summaries_and_tensorboard. [Accessed: 25-Jul-2018].

[64] R. Magalhães *et al.*, “The dynamics of stress: a longitudinal MRI study of rat brain structure and connectome,” *Mol. Psychiatry*, Dec. 2017.

[65] P. Marques, J. M. Soares, R. Magalhães, N. C. Santos, and N. Sousa, “The Bounds of Education in the Human Brain Connectome,” *Sci. Rep.*, vol. 5, pp. 1–8, 2015.

[66] B. Fischl *et al.*, “Whole Brain Segmentation: Neurotechnique Automated Labeling of NeuroanatomicalStructures in the Human Brain,” *Neuron*, vol. 33, no. 3, pp. 341–355, 2002.

[67] M. M. Esiri, “Ageing and the brain,” *J. Pathol.*, vol. 211, no. 2, pp. 181–187, 2007.

[68] “Alzheimer disease,” 2018. [Online]. Available: https://ghr.nlm.nih.gov/condition/alzheimer-disease. [Accessed: 20-Sep-2018].

[69] P. Kumar, P. Nagar, C. Arora, and A. Gupta, “U-SEGNET: Fully Convolutional Neural Network based Automated Brain tissue segmentation Tool,” in *ICIP*, 2018.

[70] H. Noh, T. You, J. Mun, and B. Han, “Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization,” no. Nips, 2017.

[71] P. Veličković, “Deep learning for complete beginners: convolutional neural networks

with keras," 2017. .

[72] M. Nielsen, "Deep Learning -Chapter 6," 2017. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap6.html. [Accessed: 01-Feb-2018].

[73] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[74] S. Sharma, "Epoch vs Batch Size vs Iterations," 2017. [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9. [Accessed: 27-Jan-2018].

Appendix

Appendix A - DL layer types

CONVOLUTION LAYER

This layer has the purpose of finding features of interest in an image and it's the first layer in CNN blocks. The original image is multiplied with a filter called kernel to detect edges and limits different times. The outcome of a multiplication of the original image with a kernel is a new feature map. The size of each kernel is a parameter specified by the user [56]. Figure A.0.1 shows the effects in an image applying a convolution layer. At the end, the feature map will retain information about the image's edges and lines [71].

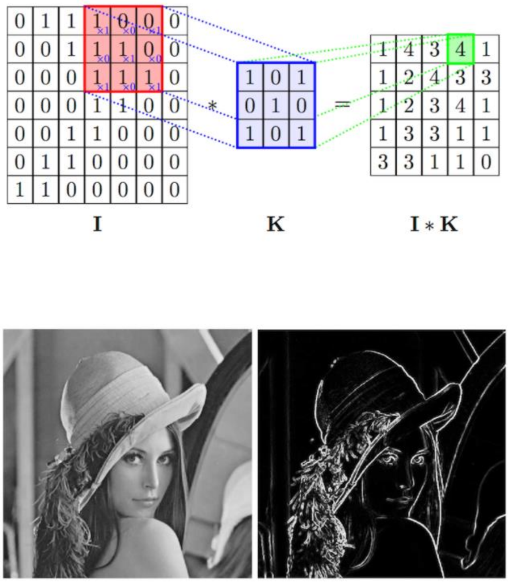


Figure A.0.1 - Convolution layer applied in an image and a resulting feature map [71]

The convolution layer can be imagined has a flashlight, sliding across all the image. The flashlight is the filter and the region it shines is the receptive field.

POOLING LAYER

The pooling operation is frequently used in CNNs in conjunction with convolution. It is a down sampling layer with the main objective of reducing the dimensionality of each feature map and retaining the important information. At the end there's a more condensed and smaller feature map. that may help to locate a certain feature in an image. It helps to reduce the computational complexity of the network. The final output will be smaller always than the layer's input[72].

The following equation shows a Max-Pooling operation reducing the final output to half of the size. The parameters used in the stride value is 2, and the filter size is 2x2:

$$\begin{bmatrix} 0.9 & 0.2 & 0.4 & 0.1 \\ 0.7 & 0.2 & 0.3 & 0.1 \\ 0.2 & 0.6 & 0.9 & 0.2 \\ 0.1 & 0.7 & 0.1 & 0.2 \end{bmatrix} \xrightarrow{Max-Pooling\ function} \begin{bmatrix} 0.9 & 0.4 \\ 0.7 & 0.9 \end{bmatrix}$$

Figure A.0.2 illustrates the pooling transformation. The feature maps reduce their size, keeping the important information:

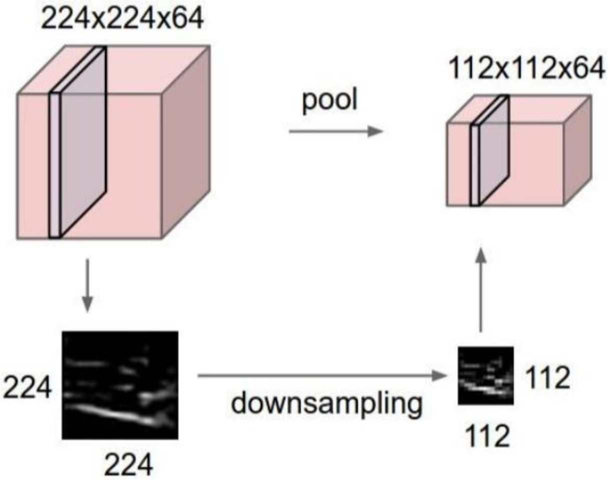


Figure A.0.2 – Max-pooling transformation

DROPOUT LAYER

Dropout randomly sets a certain percentage of their units to zero. The network most of the times can provide a correct classification even though some of the unit's weights are null. This layer is used mainly to prevent overfitting, reducing the size and amount of data on the feature maps, making a less heavy and complex training phase and contributing also for more independent layers. The dropout layers also encourages the neural network to learn and keep features that are useful on their own [56]. Figure A.0.3 represents what happens to the whole network when a dropout layer is insert [73].

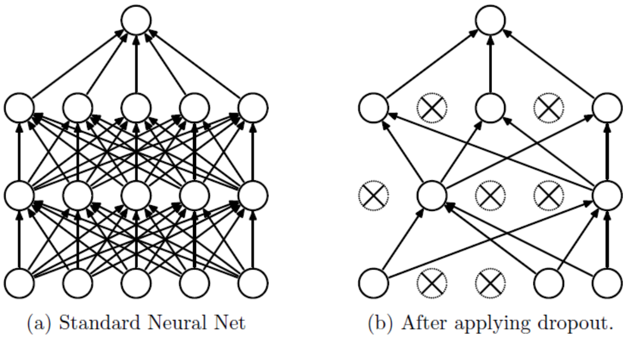


Figure A.0.3 - The effects of a dropout slice in a training network [73]

Normally in popular DL models these layers are used in Fully-Connected, but is also possible to use dropout slice after Max-Pooling layers [42].

FULLY CONNECTED (DENSE) LAYER

The fully connected layer (Figure A.0.4) is usually used in the final phase of DL's layers. Fully connected layers arranging into layers allows to use vectorize operations. Multiple neurons are computed at the same time. The more hidden layers, the more the computer can compute and resolve a complex problem.

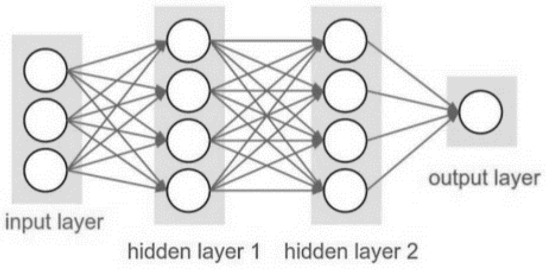


Figure A.0.4 - Example of a dense layer

BATCH NORMALIZATION LAYER

The batch normalization normalizes all the activation values of every Fully Connected or Convolution layer during training. The batch normalization improves gradient flow, allows higher learning rates, reduces the dependency on initialization.

Usually the batch normalization is used after Convolution layers and Fully connected layers. It is used also to prevent overfitting.

Appendix B – Some important concepts during DL training

EPOCH

An epoch is a single pass through the full training set. Using only one epoch is not a viable option because gradient descent only changes the weights by a small amount in the direction of improvement, so backpropagation can't get a good accuracy result if the training examples go through the DL layers just once. So, to optimize the learning using a limited dataset, there should be multiple epochs, to converge on a combination of weights with an acceptable level of accuracy.[74]

There's no correct number of epochs. It depends on the diversity of dataset and the main objective of the work. For example, a DL with the main objective to identify cats in images is less complex than identifying living beings in an image, so in this last example the training should be more efficient and with more epochs[74].

As the number of epochs increases, the more number of times the weights are changed in neural networks and the learning curve can go from underfitting to optimal to overfitting[74].

BATCH SIZE

Batch size defines number of samples that will be propagated through the network at the same time. This is an important parameter because inserting the full data in the computer's memory at once overcharges the system, so one solution is to divide it in several smaller batches[74].

If the batch size is N, and the number of samples is M, the algorithm takes the first N samples (from 1st to Nth) from the training dataset and trains network. Next it takes the next N samples (from N+1st to 2Nth) and train network again. This procedure continues until all samples propagates the networks. The batch number N should be divisible with M, otherwise the last train will be with less than N images.

TRAINING PHASE DIFFICULTIES

To understand the importance of achieving the most accurate number of epochs, there are 3 different situations in the training's phase: The underfitting, Optimum and overfitting (Figure B.0.5).

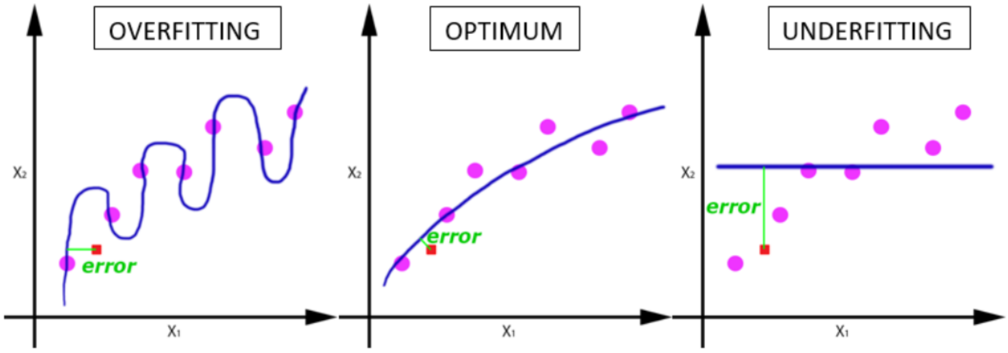


Figure B.0.5 –The three different types of training difficulties [74].

Underfitting occurs when a model is too simple, uses too few features. It is model inflexible in learning from the dataset.

An overfitting situation happens when the weights memorized all the training samples, and don't have the ability to perform well when facing new examples[73]. A model that is overfitting, has great accuracy in the training phase (99%) but lower accuracy in the test set (55%).

One way to detect overfitting during training is visualizing the behaviour of val_acc and acc in each epoch. When we are facing a case of overfitting, the line of acc and val_acc begin to separate. The following graph is one training example produced in this work and it shows exactly a case of Overfitting: the training set is always learning and achieving better accuracy at each epoch, reaching 98% at 61 epoch, and the validation set doesn't get any better than 88% of accuracy and at the end of the training it gets worse and does not improve. Figure B.0.6 presents a overfitting situation.

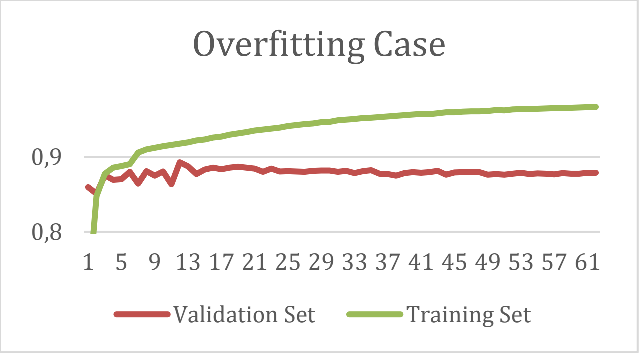


Figure B.0.6 - Overfitting case , A training graph of TensorBoard

Appendix C - Architectures

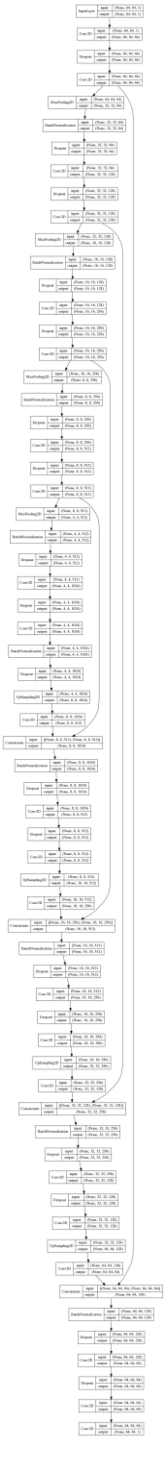


Figure C.0.7 - Architecture U-NET used for 2D convolutions approach in Rats

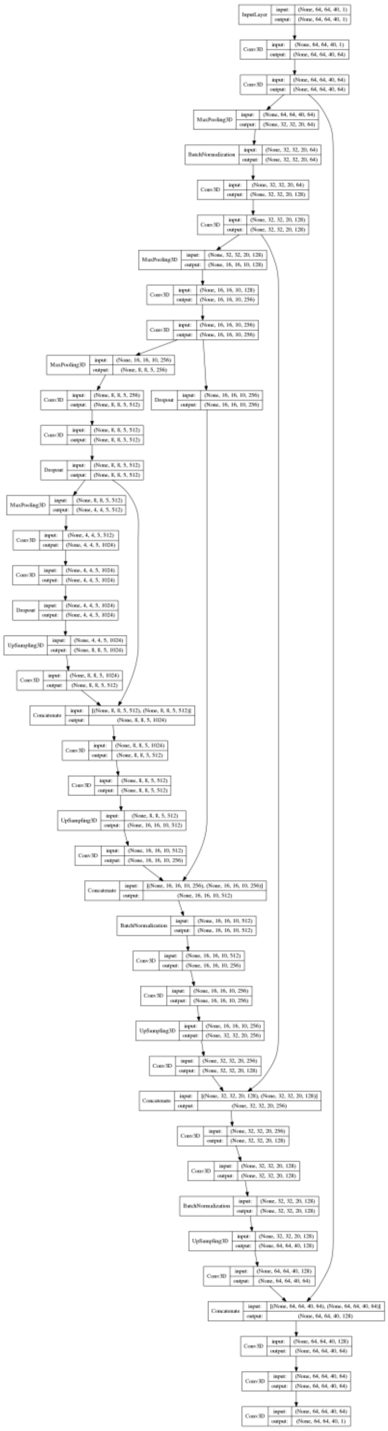

InputLayer
input: (None, 64, 64, 40, 1)
output: (None, 64, 64, 40, 1)
Conv3D
input: (None, 64, 64, 40, 1)
output: (None, 64, 64, 40, 64)
Conv3D
input: (None, 64, 64, 40, 64)
output: (None, 64, 64, 40, 64)
MaxPooling3D
input: (None, 64, 64, 40, 64)
output: (None, 32, 32, 20, 64)
BatchNormalization
input: (None, 32, 32, 20, 64)
output: (None, 32, 32, 20, 64)
Conv3D
input: (None, 32, 32, 20, 64)
output: (None, 32, 32, 20, 128)
Conv3D
input: (None, 32, 32, 20, 128)
output: (None, 32, 32, 20, 128)
MaxPooling3D
input: (None, 32, 32, 20, 128)
output: (None, 16, 16, 10, 128)
Conv3D
input: (None, 16, 16, 10, 128)
output: (None, 16, 16, 10, 256)
Conv3D
input: (None, 16, 16, 10, 256)
output: (None, 16, 16, 10, 256)
MaxPooling3D
input: (None, 16, 16, 10, 256)
output: (None, 8, 8, 5, 256)
Conv3D
input: (None, 8, 8, 5, 256)
output: (None, 8, 8, 5, 512)
Dropout
input: (None, 16, 16, 10, 256)
output: (None, 16, 16, 10, 256)
Conv3D
input: (None, 8, 8, 5, 512)
output: (None, 8, 8, 5, 512)
Dropout
input: (None, 8, 8, 5, 512)
output: (None, 8, 8, 5, 512)
MaxPooling3D
input: (None, 8, 8, 5, 512)
output: (None, 4, 4, 5, 512)
Conv3D
input: (None, 4, 4, 5, 512)
output: (None, 4, 4, 5, 1024)
Conv3D
input: (None, 4, 4, 5, 1024)
output: (None, 4, 4, 5, 1024)
Dropout
input: (None, 4, 4, 5, 1024)
output: (None, 4, 4, 5, 1024)
UpSampling3D
input: (None, 4, 4, 5, 1024)
output: (None, 8, 8, 5, 1024)
Conv3D
input: (None, 8, 8, 5, 1024)
output: (None, 8, 8, 5, 512)
Concatenate
input: [(None, 8, 8, 5, 512), (None, 8, 8, 5, 512)]
output: (None, 8, 8, 5, 1024)
Conv3D
input: (None, 8, 8, 5, 1024)
output: (None, 8, 8, 5, 512)
Conv3D
input: (None, 8, 8, 5, 512)
output: (None, 8, 8, 5, 512)
UpSampling3D
input: (None, 8, 8, 5, 512)
output: (None, 16, 16, 10, 512)
Conv3D
input: (None, 16, 16, 10, 512)
output: (None, 16, 16, 10, 256)
Concatenate
input: [(None, 16, 16, 10, 256), (None, 16, 16, 10, 256)]
output: (None, 16, 16, 10, 512)
BatchNormalization
input: (None, 16, 16, 10, 512)
output: (None, 16, 16, 10, 512)
Conv3D
input: (None, 16, 16, 10, 512)
output: (None, 16, 16, 10, 256)
Conv3D
input: (None, 16, 16, 10, 256)
output: (None, 16, 16, 10, 256)
UpSampling3D
input: (None, 16, 16, 10, 256)
output: (None, 32, 32, 20, 256)
Conv3D
input: (None, 32, 32, 20, 256)
output: (None, 32, 32, 20, 128)
Concatenate
input: [(None, 32, 32, 20, 128), (None, 32, 32, 20, 128)]
output: (None, 32, 32, 20, 256)
Conv3D
input: (None, 32, 32, 20, 256)
output: (None, 32, 32, 20, 128)
Conv3D
input: (None, 32, 32, 20, 128)
output: (None, 32, 32, 20, 128)
BatchNormalization
input: (None, 32, 32, 20, 128)
output: (None, 32, 32, 20, 128)
UpSampling3D
input: (None, 32, 32, 20, 128)
output: (None, 64, 64, 40, 128)
Conv3D
input: (None, 64, 64, 40, 128)
output: (None, 64, 64, 40, 64)
Concatenate
input: [(None, 64, 64, 40, 64), (None, 64, 64, 40, 64)]
output: (None, 64, 64, 40, 128)
Conv3D
input: (None, 64, 64, 40, 128)
output: (None, 64, 64, 40, 64)
Conv3D
input: (None, 64, 64, 40, 64)
output: (None, 64, 64, 40, 64)
Conv3D
input: (None, 64, 64, 40, 64)
output: (None, 64, 64, 40, 1)

Figure C.0.8 - U-NET architecture used in 3D convolutions approach in Rats

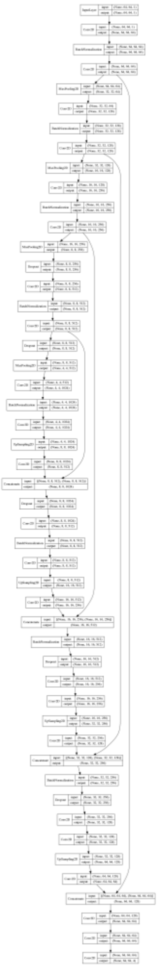


Figure C.0.9 - U-NET architecture 2D approach for semantic segmentation for Rats

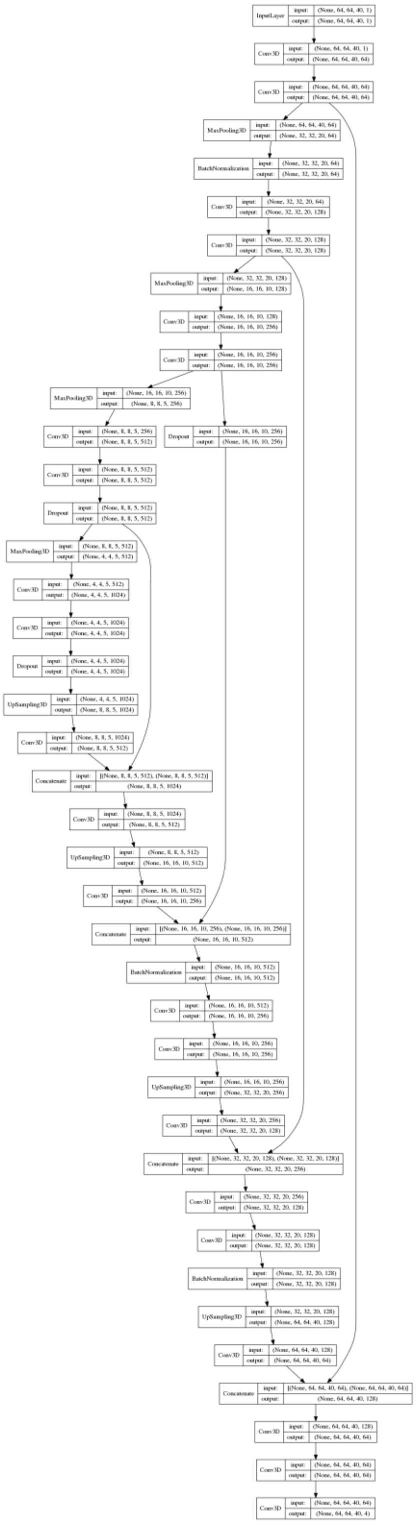

InputLayer input: (None, 64, 64, 40, 1) output: (None, 64, 64, 40, 1)
Conv3D input: (None, 64, 64, 40, 1) output: (None, 64, 64, 40, 64)
Conv3D input: (None, 64, 64, 40, 64) output: (None, 64, 64, 40, 64)
MaxPooling3D input: (None, 64, 64, 40, 64) output: (None, 32, 32, 20, 64)
BatchNormalization input: (None, 32, 32, 20, 64) output: (None, 32, 32, 20, 64)
Conv3D input: (None, 32, 32, 20, 64) output: (None, 32, 32, 20, 128)
Conv3D input: (None, 32, 32, 20, 128) output: (None, 32, 32, 20, 128)
MaxPooling3D input: (None, 32, 32, 20, 128) output: (None, 16, 16, 10, 128)
Conv3D input: (None, 16, 16, 10, 128) output: (None, 16, 16, 10, 256)
Conv3D input: (None, 16, 16, 10, 256) output: (None, 16, 16, 10, 256)
MaxPooling3D input: (None, 16, 16, 10, 256) output: (None, 8, 8, 5, 256)
Conv3D input: (None, 8, 8, 5, 256) output: (None, 8, 8, 5, 512)
Dropout input: (None, 16, 16, 10, 256) output: (None, 16, 16, 10, 256)
Conv3D input: (None, 8, 8, 5, 512) output: (None, 8, 8, 5, 512)
Dropout input: (None, 8, 8, 5, 512) output: (None, 8, 8, 5, 512)
MaxPooling3D input: (None, 8, 8, 5, 512) output: (None, 4, 4, 5, 512)
Conv3D input: (None, 4, 4, 5, 512) output: (None, 4, 4, 5, 1024)
Conv3D input: (None, 4, 4, 5, 1024) output: (None, 4, 4, 5, 1024)
Dropout input: (None, 4, 4, 5, 1024) output: (None, 4, 4, 5, 1024)
UpSampling3D input: (None, 4, 4, 5, 1024) output: (None, 8, 8, 5, 1024)
Conv3D input: (None, 8, 8, 5, 1024) output: (None, 8, 8, 5, 512)
Concatenate input: [(None, 8, 8, 5, 512), (None, 8, 8, 5, 512)] output: (None, 8, 8, 5, 1024)
Conv3D input: (None, 8, 8, 5, 1024) output: (None, 8, 8, 5, 512)
UpSampling3D input: (None, 8, 8, 5, 512) output: (None, 16, 16, 10, 512)
Conv3D input: (None, 16, 16, 10, 512) output: (None, 16, 16, 10, 256)
Concatenate input: [(None, 16, 16, 10, 256), (None, 16, 16, 10, 256)] output: (None, 16, 16, 10, 512)
BatchNormalization input: (None, 16, 16, 10, 512) output: (None, 16, 16, 10, 512)
Conv3D input: (None, 16, 16, 10, 512) output: (None, 16, 16, 10, 256)
Conv3D input: (None, 16, 16, 10, 256) output: (None, 16, 16, 10, 256)
UpSampling3D input: (None, 16, 16, 10, 256) output: (None, 32, 32, 20, 256)
Conv3D input: (None, 32, 32, 20, 256) output: (None, 32, 32, 20, 128)
Concatenate input: [(None, 32, 32, 20, 128), (None, 32, 32, 20, 128)] output: (None, 32, 32, 20, 256)
Conv3D input: (None, 32, 32, 20, 256) output: (None, 32, 32, 20, 128)
Conv3D input: (None, 32, 32, 20, 128) output: (None, 32, 32, 20, 128)
BatchNormalization input: (None, 32, 32, 20, 128) output: (None, 32, 32, 20, 128)
UpSampling3D input: (None, 32, 32, 20, 128) output: (None, 64, 64, 40, 128)
Conv3D input: (None, 64, 64, 40, 128) output: (None, 64, 64, 40, 64)
Concatenate input: [(None, 64, 64, 40, 64), (None, 64, 64, 40, 64)] output: (None, 64, 64, 40, 128)
Conv3D input: (None, 64, 64, 40, 128) output: (None, 64, 64, 40, 64)
Conv3D input: (None, 64, 64, 40, 64) output: (None, 64, 64, 40, 64)
Conv3D input: (None, 64, 64, 40, 64) output: (None, 64, 64, 40, 4)

Figure C.0.10 - U-NET architecture 3D convolutions approach for semantic segmentation for Rats

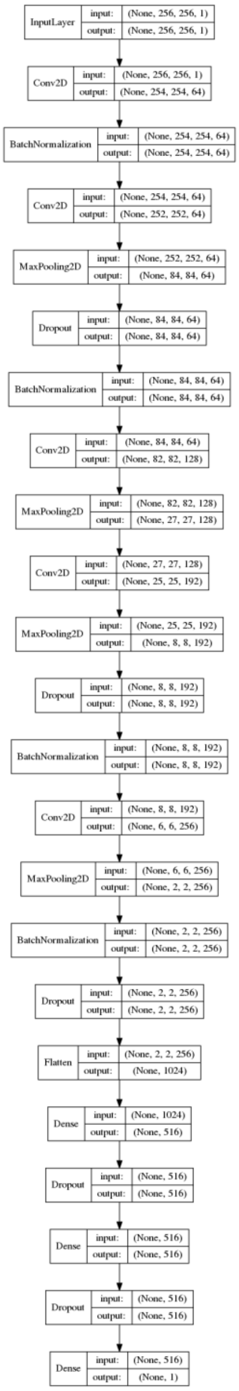

InputLayer
input: (None, 256, 256, 1)
output: (None, 256, 256, 1)
Conv2D
input: (None, 256, 256, 1)
output: (None, 254, 254, 64)
BatchNormalization
input: (None, 254, 254, 64)
output: (None, 254, 254, 64)
Conv2D
input: (None, 254, 254, 64)
output: (None, 252, 252, 64)
MaxPooling2D
input: (None, 252, 252, 64)
output: (None, 84, 84, 64)
Dropout
input: (None, 84, 84, 64)
output: (None, 84, 84, 64)
BatchNormalization
input: (None, 84, 84, 64)
output: (None, 84, 84, 64)
Conv2D
input: (None, 84, 84, 64)
output: (None, 82, 82, 128)
MaxPooling2D
input: (None, 82, 82, 128)
output: (None, 27, 27, 128)
Conv2D
input: (None, 27, 27, 128)
output: (None, 25, 25, 192)
MaxPooling2D
input: (None, 25, 25, 192)
output: (None, 8, 8, 192)
Dropout
input: (None, 8, 8, 192)
output: (None, 8, 8, 192)
BatchNormalization
input: (None, 8, 8, 192)
output: (None, 8, 8, 192)
Conv2D
input: (None, 8, 8, 192)
output: (None, 6, 6, 256)
MaxPooling2D
input: (None, 6, 6, 256)
output: (None, 2, 2, 256)
BatchNormalization
input: (None, 2, 2, 256)
output: (None, 2, 2, 256)
Dropout
input: (None, 2, 2, 256)
output: (None, 2, 2, 256)
Flatten
input: (None, 2, 2, 256)
output: (None, 1024)
Dense
input: (None, 1024)
output: (None, 516)
Dropout
input: (None, 516)
output: (None, 516)
Dense
input: (None, 516)
output: (None, 516)
Dropout
input: (None, 516)
output: (None, 516)
Dense
input: (None, 516)
output: (None, 1)

Figure C.0.11 - Classification NET used in classification of slices with brain and slice without brain

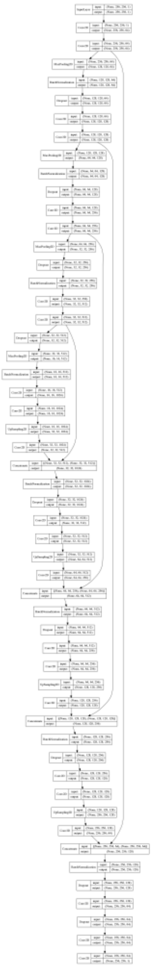


Figure C.0.12 - U-NET architecture used for brain segmentation in Humans (2D convolutions approach)

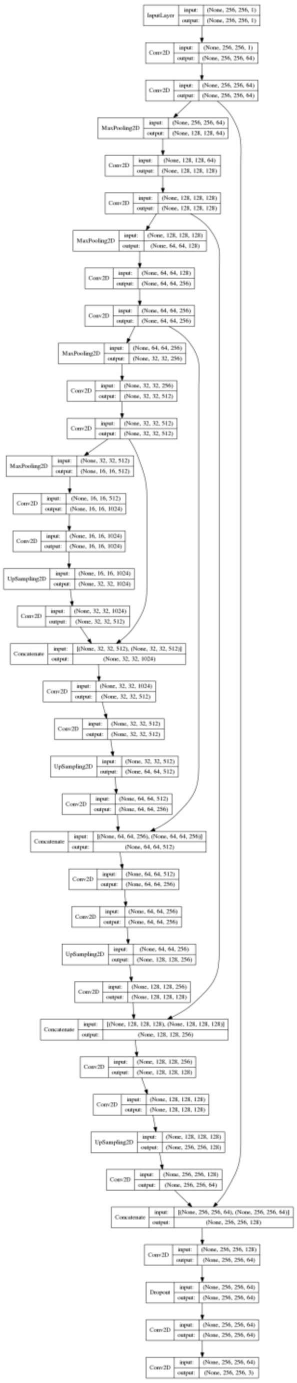

InputLayer
input: (None, 256, 256, 1)
output: (None, 256, 256, 1)
Conv2D
input: (None, 256, 256, 1)
output: (None, 256, 256, 64)
Conv2D
input: (None, 256, 256, 64)
output: (None, 256, 256, 64)
MaxPooling2D
input: (None, 256, 256, 64)
output: (None, 128, 128, 64)
Conv2D
input: (None, 128, 128, 64)
output: (None, 128, 128, 128)
Conv2D
input: (None, 128, 128, 128)
output: (None, 128, 128, 128)
MaxPooling2D
input: (None, 128, 128, 128)
output: (None, 64, 64, 128)
Conv2D
input: (None, 64, 64, 128)
output: (None, 64, 64, 256)
Conv2D
input: (None, 64, 64, 256)
output: (None, 64, 64, 256)
MaxPooling2D
input: (None, 64, 64, 256)
output: (None, 32, 32, 256)
Conv2D
input: (None, 32, 32, 256)
output: (None, 32, 32, 512)
Conv2D
input: (None, 32, 32, 512)
output: (None, 32, 32, 512)
MaxPooling2D
input: (None, 32, 32, 512)
output: (None, 16, 16, 512)
Conv2D
input: (None, 16, 16, 512)
output: (None, 16, 16, 1024)
Conv2D
input: (None, 16, 16, 1024)
output: (None, 16, 16, 1024)
UpSampling2D
input: (None, 16, 16, 1024)
output: (None, 32, 32, 1024)
Conv2D
input: (None, 32, 32, 1024)
output: (None, 32, 32, 512)
Concatenate
input: [(None, 32, 32, 512), (None, 32, 32, 512)]
output: (None, 32, 32, 1024)
Conv2D
input: (None, 32, 32, 1024)
output: (None, 32, 32, 512)
Conv2D
input: (None, 32, 32, 512)
output: (None, 32, 32, 512)
UpSampling2D
input: (None, 32, 32, 512)
output: (None, 64, 64, 512)
Conv2D
input: (None, 64, 64, 512)
output: (None, 64, 64, 256)
Concatenate
input: [(None, 64, 64, 256), (None, 64, 64, 256)]
output: (None, 64, 64, 512)
Conv2D
input: (None, 64, 64, 512)
output: (None, 64, 64, 256)
Conv2D
input: (None, 64, 64, 256)
output: (None, 64, 64, 256)
UpSampling2D
input: (None, 64, 64, 256)
output: (None, 128, 128, 256)
Conv2D
input: (None, 128, 128, 256)
output: (None, 128, 128, 128)
Concatenate
input: [(None, 128, 128, 128), (None, 128, 128, 128)]
output: (None, 128, 128, 256)
Conv2D
input: (None, 128, 128, 256)
output: (None, 128, 128, 128)
Conv2D
input: (None, 128, 128, 128)
output: (None, 128, 128, 128)
UpSampling2D
input: (None, 128, 128, 128)
output: (None, 256, 256, 128)
Conv2D
input: (None, 256, 256, 128)
output: (None, 256, 256, 64)
Concatenate
input: [(None, 256, 256, 64), (None, 256, 256, 64)]
output: (None, 256, 256, 128)
Conv2D
input: (None, 256, 256, 128)
output: (None, 256, 256, 64)
Dropout
input: (None, 256, 256, 64)
output: (None, 256, 256, 64)
Conv2D
input: (None, 256, 256, 64)
output: (None, 256, 256, 64)
Conv2D
input: (None, 256, 256, 64)
output: (None, 256, 256, 3)

Figure C.0.13 - U-NET architecture used for brain semantic segmentation in Humans (2D convolutions approach)