Deep Learning in Medical Image Analysis: Alzheimer’s Disease Classification & Whole Fetal Brain Segmentation.

A thesis presented by

J. Alejandro Valdés Valdés

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Monterrey, Nuevo León, Dec, 2019
Instituto Tecnologico y de Estudios Superiores de Monterrey
Campus Monterrey

The committee members, hereby, certify that they have read the thesis presented by J. Alejandro Valdés Valdés and that it is fully adequate in scope and quality as a partial requirement for the degree of Master of Science in Computer Sciences.

Dr. José Tamez Peña
ITESM
Principal Advisor

Dr. Emmanuel Martínez Ledesma
ITESM
Committee Member

Dr. Hugo Terashima Martín
ITESM
Committee Member

Dr. Kiho Im
Harvard University
Committee Member

Dirección Nacional de Posgrado
Dr. Rubén Morales Menéndez

Dr. Rubén Morales Menéndez
Associate Dean of Graduate Studies
School of Engineering and Sciences

Monterrey, Nuevo León, Dec, 2019
Declaration of Authorship

I, J. Alejandro Valdés Valdés, declare that this thesis titled, Deep Learning in Medical Image Analysis: Alzheimer’s Disease Classification & Whole Fetal Brain Segmentation. and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

J. Alejandro Valdés Valdés
Monterrey, Nuevo León, Dec, 2019

©2019 by J. Alejandro Valdés Valdés
All Rights Reserved
Dedication

This work is dedicated to my friends and family for always believing in me and pushing me to be a better version of myself. Specially to my mother and father, Ma. Elena and Alejandro, who have always supported me.
Acknowledgements

I want to take my advisor Dr. José Tamez, who has been a crucial part of my research, I would also like to thanks Dr. Kiho Im and HyukJin Yun both of whom were a fundamental part on my research stay at FNNDSC at Boston Children’s Hospital and Paty Sosa who always cheered me on during our stay.

As a special mention I’d like to thank Dr. Edgar Vallejo. He was extremely helpful during this work, may he rest in peace.

Finally, I would like to thank Tecnológico de Monterrey for the opportunity of carrying out my master’s degree studies by giving me a full scholarship. I would also like to thank CONACyT for the living aid provided during these past two years.
Deep Learning in Medical Image Analysis: 
Alzheimer’s Disease Classification & Whole Fetal Brain 
Segmentation. 

by 
J. Alejandro Valdés Valdés 

Abstract 

During the last couple years there has been a growing interest in Deep Learning (DL), thanks in part to an increasing availability of computational power and of massive labeled datasets. This work presents two different Deep Learning applications for medical image analysis related tasks.

The first application is the classification of Alzheimer Disease (AD) using MRI. AD is the most common form of dementia. However, it’s difficult to diagnose. During this work three different DL models are proposed to classify subjects into either AD or Normal Control (NC) groups. The models proposed are a custom CNN model, vgg16 + Transfer Learning and vgg16 + Fine Tuning. The vgg16 based methods make use of pre-learned weights trained on the ImageNet dataset. All of the models used as an input a single 2D MRI slice. This helped reduced model complexity and training times. After performing a 10 times repeated 5-Fold cross validation it was discovered that the best model was vgg16 with Fine Tuning. It reached an accuracy of 80% and an AUC of 0.86. The obtained results proved that the use of weights learned from different domains could be useful in medical image applications. The main contributions of this section were the use of a single MRI slice to classify patients, the model validation technique and the use of transfer learning.

The second application is the fully automatization of the Whole Fetal Brain Segmentation in maternal MRI scans. For this U-Net [41] and other segmentation networks were compared. The U-Net model was also modified to include two types of attention components. Squeeze and Excitation module [16] and Attention Gates [39]. To avoid overfitting the models were trained using rigorous regularization (channel wise dropout, weight decay and data augmentation). They were also trained using different loss functions, including a Hybrid (BCE+Dice) loss function. The results of the 10-fold cross validation experiments showed statistically that no model was able to outperform the vanilla U-Net (DSC of 0.93 and HD 7.59mm). The proposed Attention Gated U-Net trained with the Hybrid loss reached a 0.95 DSC and 5.03mm HD (no significant difference). However, results on challenging data show a better performance for the Attention Gated models. This led to the conclusion that the possible advantage of using attention gates outweighed the cost of adding the attention module. The results were also improved by the use of a post processing step based on traditional computer vision techniques. The main contributions to this problem were the use of 10-fold cross validation, the statistical comparison between models, the use of the Hausdorff distance metric, the post processing step and the addition of attention components.

This work also provides an overview of other deep learning applications also used within
a medical setting. The techniques mentioned include image labeling, autoencoders and GANs for data augmentation. It also provides an overview of the current challenges and limitations DL faces and the most common way of mitigating them. Thanks to the research conducted during this thesis it can be concluded that the use of DL can be used in a medical setting focused on the analysis of images. Particularly on classification and segmentation tasks. However, certain challenges such as having a large enough data set and availability of the required hardware to carry out research must be considered before attempting to solve a problem with DL.
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of a classification problem. Figure taken from Bishop (2006) [2].</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of a clustering problem. Figure taken from Bishop (2006) [2].</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Example of the Gradient Descent technique. Figure taken from Goodfellow et al. (2016) [11].</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Example of the forward pass for a two layer MLP. Figure taken from Bishop (2006) [2].</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>Example of a valid convolution operation. Taken from Goodfellow et al. (2016) [11].</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of a typical CNN layer, the convolution, activation and pooling operations are commonly refereed to as a convolutional layer. Taken from Goodfellow et al. (2016) [11].</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>Summary of transfer learning model.</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Summary of fine tuning model.</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>Summary of custom classification model.</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Model validation technique employed.</td>
<td>23</td>
</tr>
<tr>
<td>3.5</td>
<td>Accuracy and loss progression across epochs for Transfer Learning method.</td>
<td>24</td>
</tr>
<tr>
<td>3.6</td>
<td>Accuracy and loss progression across epochs for Fine Tuning method.</td>
<td>25</td>
</tr>
<tr>
<td>3.7</td>
<td>Accuracy and loss progression across epochs for custom CNN method.</td>
<td>25</td>
</tr>
<tr>
<td>3.8</td>
<td>ROC curves for all model’s validation.</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>An example of K-Fold cross validation (K=3), all the training samples are grouped into three groups. These groups are then left out one at a time while the remaining are used for training. Performance metrics are applied to the held out group. Taken from [28].</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>Example of the transformation of an image prediction CNN into a Fully Convoluional Neural Network (FCN), taken from Long and Shelhamer (2015) [34].</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>FCN using vgg19 as backbone, also shown is the not used classification layer of vgg19 which is disregarded to create this FCN model.</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>FCN prediction refinement trough the fusion of multi layer information, taken from taken from Long and Shelhamer (2015) [34].</td>
<td>39</td>
</tr>
<tr>
<td>4.5</td>
<td>Original U-Net architecture taken from Ronneberger et al. [41]. Blue boxes correspond to a set of feature maps. The number of feature maps is shown on top of the box while the dimensions are provided at the lower left edge. White boxes represent copied feature maps. Taken from Ronneberger et al. [41].</td>
<td>39</td>
</tr>
</tbody>
</table>
4.6 U-Net architecture, arrows denote different operations .................................. 40
4.7 Mask R-CNN overview, taken from He et al. (2018) [14] ................................. 41
4.8 RoIAlign overview, taken from He et al. (2018) [14] .................................. 42
4.9 Overview of how the Squeeze and Excitation Module learns individual weights for each feature channel, taken from Hu et al. (2017) [16]. .......................... 43
4.10 U-Net + Squeeze and Excitation module ................................................... 43
4.11 Attention gate module, taken from Schlemper et al. (2018) [45]. ................... 44
4.12 U-Net model with attention gate module that limits the information flow between downwards and upwards paths ......................................................... 44
4.13 Training progression across 25 epochs for the basic models used. ................... 46
4.14 Training progression across 25 epochs for models that employ either of the two attention modules (SE or Attention Gate). ................................. 48
4.15 Training progression across 25 epochs for models that employ extra regularization techniques .................................................................................. 50
4.16 Training progression across 25 epochs for different Attention Gated U-Net models with different loss functions, these models also employ extra regularization techniques. .............................................. 50
4.17 Segmentation results for multiple models, the first 4 rows show challenging images while the last 3 rows show normal cases. ................................. 53
4.18 Representation of what the vanilla U-Net model sees at the end of each convolution block. .................................................................................. 55
4.19 Representation of what the Attention Gated U-Net model sees at the end of each convolution block and of the effect that the attention gate has on the information transferred from the encoder to the decoder so that only noteworthy spatial information is shared. ................................. 56

5.1 Overall workflow for the automated image annotation, taken from [49]. ....... 60
5.2 Stacked autoencoders being used as a feature extractor [48]. ....................... 60
5.3 GAN model overview taken from [9]. ......................................................... 61
5.4 Total accuracy results for liver lesion classification using different data augmentation techniques, taken from [9]. .............................................. 62
5.5 (a) U-Net architecture (b) NAS-Unet architecture, the rectangles represent the proposed cell architectures that are automatically searched along with the gray arrows that represent a transformation operation, taken from [59]. .......... 62
List of Tables

3.1 Overview subject distribution. .................................................. 19
3.2 Comparison between number of parameters in each model ............. 20
3.3 Splitting of data for basic classification. .................................. 22
3.4 Comparison between number of trainable layers for vgg16 for the Fine Tuning approach. ......................................................... 26
3.5 Model validation results for each of the three used models, with their corresponding 95% CI range. ......................................................... 26

4.1 Plane orientation distribution for the available dataset. ................... 34
4.2 Results for basic models. Shown in *italics* are the results that show a worse performance than the U-Net results and where there is a statistical difference calculated with the Wilcoxon test. ......................... 47
4.3 Results for models using attention modules. Shown in *italics* are the results that show a worse performance than the U-Net results and where there is a statistical difference calculated with the Wilcoxon test. ......................... 48
4.4 Results for models using additional regularization techniques and different loss functions. Models with * before their name use data augmentation, channel-wise dropout and L2 weight decay. Shown in *italics* are the results that show a worse performance than the U-Net results and where there is a statistical difference calculated with the Wilcoxon test. ......................... 52
4.5 Number of trainable parameters in different segmentation models ....... 57
# Contents

Abstract ix
List of Figures xii
List of Tables xiii

1 Introduction 1
  1.1 Introduction ............................................................... 1
  1.2 Problem Definition ..................................................... 2
    1.2.1 Alzheimer Disease Classification .................................. 2
    1.2.2 Whole Fetal Brain Segmentation ................................... 2
  1.3 Objectives and Hypothesis ............................................. 3
    1.3.1 Alzheimer Disease Classification .................................. 3
    1.3.2 Whole Fetal Brain Segmentation ................................... 3
  1.4 Solution Overview ..................................................... 4
  1.5 Contributions ........................................................ 4
  1.6 Thesis Outline ........................................................ 5

2 Theoretical Framework 7
  2.1 Machine Learning .................................................... 7
  2.2 Artificial Neural Networks .......................................... 9
    2.2.1 Training of Artificial Neural Networks .......................... 10
  2.3 Deep Learning ....................................................... 11
    2.3.1 Convolutional Neural Networks .................................. 11
  2.4 Summary ............................................................. 14

3 Alzheimer Disease Classification 15
  3.1 Introduction ........................................................... 15
  3.2 Related Work .......................................................... 16
  3.3 Methodology ........................................................... 18
    3.3.1 Data Acquisition .................................................. 18
    3.3.2 Implementation of Convolutional Neural Networks ............. 19
    3.3.3 Basic Training .................................................... 20
    3.3.4 Finding Optimal N for Fine Tuning ............................... 22
    3.3.5 Model Validation .................................................. 23
  3.4 Results ............................................................... 24
3.4.1 Basic Training ................................................. 24
3.4.2 Finding Optimal N for Fine Tuning ........................ 25
3.4.3 Model Validation ............................................. 26
3.5 Discussion ....................................................... 26
3.5.1 Basic Training .................................................. 26
3.5.2 Finding Optimal N for Fine Tuning ........................ 27
3.5.3 Model Validation ............................................. 27
3.6 Conclusion ....................................................... 29
3.7 Summary ......................................................... 29

4 Whole Fetal Brain Segmentation ........................................ 31
4.1 Introduction ..................................................... 31
4.2 Related Work ..................................................... 32
4.3 Methodology .................................................... 34
   4.3.1 Data Acquisition ........................................... 34
   4.3.2 Performance Metrics ....................................... 35
   4.3.3 Model Training and Validating ............................. 35
   4.3.4 Segmentation Models ....................................... 37
   4.3.5 Attention Modules .......................................... 42
   4.3.6 Regularization .............................................. 45
   4.3.7 Loss Functions .............................................. 45
4.4 Results ........................................................ 45
   4.4.1 Basic Models ............................................... 45
   4.4.2 Attention Modules .......................................... 47
   4.4.3 Regularization and Loss Functions Results ................. 47
4.5 Discussion ....................................................... 49
4.6 Conclusion ....................................................... 57
4.7 Summary ......................................................... 57

5 Discussion ........................................................ 59
5.1 Other Deep Learning Applications .............................. 59
5.2 Current Challenges and Limitations ............................ 61
5.3 Summary ......................................................... 64
   5.3.1 Future Work ............................................... 65

6 Conclusion ........................................................ 67

Bibliography .......................................................... 78
Chapter 1

Introduction

1.1 Introduction

Ever since medical images were able to be processed by computers medical image analysis has been an area of interest for scientists. Since the 1970s Artificial Intelligence (AI) systems have been developed to accelerate research within this field. They have evolved from expert and ruled based systems into supervised techniques where training data is used to develop models and unsupervised techniques that finds patterns within data. These techniques that largely rely on pattern recognition have been dubbed Machine Learning (ML). Compared to older AI techniques that were programmed by humans to perform a specific task, ML models are trained by using feature vectors extracted from available data to find optimal solutions [33].

During the last couple years there has been a growing interest in Machine Learning. Particularly in Deep Learning (DL), a type of Artificial Neural Network (ANN). This interest has grown thanks to the increase in computational power and availability of massive labeled datasets [8]. Since the popularization of Deep Learning, it has had a tremendous impact in a wide variety of problems beating records in image and text recognition [29].

The popularity of Deep Learning on image related tasks can be traced back to 2012 when Krizhevsky et al. 2012 [27] won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [42] with a Convolutional Neural Network (CNN) that achieved a 15% error rate. While the second place was only able to reach an error rate of 26%. Ever since then, CNNs have been the most used method in ILSVRC competitions. In the 2015 competition a CNN model was capable of surpassing human performance at recognizing images. Thanks to this growing popularity, the research around applications for CNNs has increased. This includes the medical image analysis field for which CNNs have become the method of choice for many researchers [20].

Litjens et al. (2017) [33] provide an overview of possible applications for Deep Learning within medical image analysis. Some of the applications mentioned are: image classification to aid in a diagnosis, lesion detection, organ localization, organ or lesion segmentation, registration of medical images and image enhancement. It is clear that there exists a wide variety of medical image analysis applications that can benefit from the use of Deep Learning.

The objective of this work is to use CNNs to work on two different medical image
analysis related tasks, classification of Alzheimer’s Disease (AD) and whole fetal brain segmentation. This with the goal of showcasing how DL can be exploited in a medical setting.

The following section 1.2 will introduce the two problems that will be addressed during this work. Both of whom will be furthered addressed in their individual chapters. AD Classification in chapter 3 and Whole Fetal Brain Segmentation in chapter 4. Section 1.3 will outline the proposed objectives, research questions and hypothesis for each of the problems. Following this, section 1.4 provides an overview of the proposed solutions and section 1.5 highlights the main contributions of this work. Finally, section 1.6 presents an outline of the thesis document.

1.2 Problem Definition

1.2.1 Alzheimer Disease Classification

Alzheimer’s Disease (AD) is the most common form of dementia. But, due to the nature of its exclusion based diagnosis it’s difficult to detect, especially before the first symptoms present themselves [36]. There is a clear need to correctly detect subjects that suffer from AD, so they can be included in surrounding clinical trials. There are multiple works that have used 3D CNN to classify subjects into AD, Normal Control (NC), Progressive Mild Cognitive Impairment (pMCI) or Stable Mild Cognitive Impairment (sMCI) categories [26, 47, 1]. There has also been research in using Magnetic Resonance Imaging (MRI) to predict if someone will develop AD in the following years, some showing promising results [32].

What will be tried to be accomplished in this work, is the classification of patients as either having or not having AD. However, only one slice of a subjectMRI will be used. This reduces the computational power needed and also removes the use of further image processing that has been previously used to detect the presence of AD. An important part of this research will be to determine if the use of previously trained network weights, trained in a different domain provide an advantage while training of the models. The use of pre-trained weights is incorporated by either using the pre-trained weights as a feature extraction method called transfer learning or by fine tuning the pre-trained weights during the training epochs.

1.2.2 Whole Fetal Brain Segmentation

The problem of whole fetal brain segmentation is an important step in the fetal brain analysis pipeline employed at Fetal Neonatal Neuroimaging and Developmental Science Center (FN-NDSC) at Boston Children’s Hospital. It is the first step in their motion correction process necessary for further analysis. Currently this process is carried out in a manual manner. There has been previous research that seeks to automatize this segmentation step using U-Net, a specific deep learning architecture for segmentation tasks [44]. However, the resulting model was not able to correctly automate the segmentation of unseen data.

During this part of the thesis, different segmentation architectures are analyzed as possible alternatives to the U-Net based approach. Some of these architectures are augmented using specialized attention modules and extra regularization techniques,
1.3 Objectives and Hypothesis

1.3.1 Alzheimer Disease Classification

Objectives

- Obtain Access, understand and recollect the data provided by ADNI
- Use a CNN to create a binary classifier for both classes.
- Use an established CNN architecture with fixed pre-trained weights as a feature extractor that is feed into a trainable network composed of multiple dense layers (Transfer Learning).
- Use an established CNN architecture with a variable number of trainable layers pre-trained weights as a feature extractor that is feed into a trainable network composed of multiple dense layers (fine tuning).
- Find the best number $n$ of trainable layers in the fine tuning model.
- Validate all of the models using a 10 times repeated 5-Fold cross validation.

Hypothesis: For the task of classifying a subject as either having or not having Alzheimer’s Disease a Convolutional Neural Network, that takes as an input a single Axial MRI slice, can be used. Furthermore the use of pre-trained weights from a different domain is beneficial in the training process.

1.3.2 Whole Fetal Brain Segmentation

Objectives

- Manually create the goal masks of the available maternal MRI scans.
- Compute the intra-rating score of the created goal masks with those created by another masker.
- Train the vanilla U-Net model, FCN model and Mask R-CNN models.
- Train the best resulting models using extra regularization techniques as to avoid overfitting. Using dropout, weight decay, different loss functions and data augmentation.
- Insert attention modules into the best working model. In the way of Squeeze and Excitation and Attention Gate modules.
- Compare the results of the 10-fold cross validation using each of the employed models.

Hypothesis: For the task of whole fetal brain segmentation the inclusion of attention modules to the U-Net segmentation model is able to statistically show a significant improvement over the vanilla U-Net model.
1.4 Solution Overview

The proposed solution overview for the two task at hand is as follows:

- **AD Classification**: The data taken from ADNI is used to make two different classification groups. For each of the subjects in each group the latest MRI scan is located and a single MRI slice is extracted saved. These single 2D slices are then used as input for three different CNN classification networks: one custom model, one based on feature extraction and a final one based on fine tuning. The feature extraction and fine tuning models make use of weights pre-learned on the ImageNet dataset. Three different experiments are then carried out. The first one focuses on analysing the learning progression. The second one seeks to find the optimal number of layers that should be fine tuned. The final experiment validates the three different models using a 10 times repeated 5 fold cross validation technique. The final per subject results of the last experiments are achieved by averaging the 10 different predictions for each model. The models are compared using their achieved accuracy, AUC and BER.

- **Whole Fetal Brain Segmentation**: The data used for this problem was provided by the FNNDSC at Boston Children’s Hospital. The MRI scans are pre-processed to normalize their intensity. The scans were fed to U-Net, FCN and Mask R-CNN and their results compared. To control an overfitting effect noted extra regularization procedures were employed. These procedures included channel wise dropout, weight decay and data augmentation. The models were also trained using different loss functions, including a proposed hybrid loss which combines a weighted BCE loss function with a more goal oriented Dice loss function. The best working models were then modified as to include attention modules. To compare the models 10-fold cross validation was carried out for all of the proposed models and their modifications and the results compared using the Wilcoxon Rank test.

1.5 Contributions

The main contributions by this research to the current state of the art are:

1. This work presents an overview of the current state of the art for both AD classification and fetal brain segmentation using deep learning.

2. For the case of AD classification the models were capable of learning employing only a single MRI slice as an input. This also allowed for the inclusion of pre-trained weights trained on a different domain.

3. A method to find the optimal number of layers to train during a fine tuning protocol.

4. The proposed 10 times repeated 5 fold cross validation helps validate a model on a reduced dataset.

5. For the case of whole fetal brain segmentation different Deep Learning models were compared against the Vanilla U-Net model.
6. The regularization techniques employed, particularly the data augmentation procedure to increase the training dataset size, greatly aided in the reduction of overfitting.

7. The combination of binary cross entropy and dice loss functions into a hybrid loss aided in the learning progress and appears to work better on difficult images.

8. The addition of attention gates limit information flow so that only relevant areas are shared. This seemed to improve learning progression.

9. The Hausdorff Distance was used as a performance metric.

10. The Wilcoxon rank test was used to find statistical differences between models.

1.6 Thesis Outline

This work is divided into the following chapters. Chapter 2 introduces the shared theoretical framework of the two problems at hand, Convolutional Neural Network (CNN) are introduced and its inner workings explained. The following chapters, chapter 3 and 4, present the carried out research and results for the problems of Alzheimer Disease classification and automatic whole fetal brain segmentation in maternal MRI. Chapter 5 provides a discussion regarding possible future work for both of the addressed problems, not touched upon Deep Learning applications in medical image analysis and on the current limits and challenges Deep Learning faces. Finally chapter 6, provides a conclusion regarding the full thesis project.
Chapter 2

Theoretical Framework

This chapter establishes the theoretical framework by which this work is shaped and supported. It begins with an overview of Machine Learning (ML) and the way it’s techniques are categorized. Then a deeper explanation of Artificial Neural Network (ANN) is given, including how ANNs learn through the back propagation algorithm. Following this, an introduction to Deep Learning is provided. The chapter concludes with the concept of Convolutional Neural Network (CNN).

2.1 Machine Learning

ML methods are commonly divided into one of two categories, supervised learning methods and unsupervised learning methods. In this section both categories are briefly outlined. General characteristics of ML methods are also mentioned.

Figure 2.1: Example of a classification problem. Figure taken from Bishop (2006) [2].

Supervised learning method encompass applications for which not only input data is provided to the model but also its corresponding desired output data, commonly referred
to as labeled data. Examples of applications that fall into this category are classification models that require the use of labeled data to learn how to classify inputs into one or more discrete categories and regression models that aim to provide one or more continuous variables for each input feature vector [2]. Figure 2.1, shows a visual representation of a three class classification problem that can be solved using supervised learning.

![Figure 2.1: A visual representation of a three class classification problem.](image)

On the other hand unsupervised learning applications solely require the input data is available. There is no need to provide the desired outcome or the data labels. Clustering problems can be solved via unsupervised learning. In this kind of problems the end goal is the discovery of similar groups within the input data [2]. An example of a clustering application, which finds two different clusters within the input data can be observed in 2.1.

Most machine learning algorithms, such as ANNs, revolve around the optimization of some objective function $f(x)$ by constantly modifying $x$. When the objective function is minimized it is also commonly called cost or loss function [11]. To optimize the objective function one of the most widely known optimization methods is Gradient Descent (GD), which can be observed in figure 2.3. The GD technique involves the use of the derivative of a function $y = f(x)$, denoted as $f'(x)$, to find the slope of $f(x)$ for a certain value of $x$. This indicates how the input $x$ should be modified in order to obtain the desired $y$ [11]. This method is executed by using all the training data samples at once to update the model parameters. A drawback of this method is that depending on the function there could exist local minimum or maximum areas, where the surrounding points no longer show an improvement. This can be troublesome as the method could get stuck and think it has found the best results, while the correct global minimum or maximum has been overlooked. A way of fixing this is Stochastic Gradient Descent (SGD). SGD is a variation of GD that instead of using all the available training samples to update a parameter, Stochastic Gradient Descent (SGD) requires only a random subset to update the models parameters [11].
2.2 Artificial Neural Networks

In this section Artificial Neural Network (ANN), one of the most popular supervised learning ML method is introduced.

The term ANN comes from an attempt of mathematically representing the way biological systems process information. An example of an ANN is the Multi Layer Perceptron (MLP) which has proved to be of great value. MLPs are represented by a series of layers interconnected with each other in a sequential manner. The layers are made up of multiple units or nodes, that are either activated or deactivated to allow information to flow through the networks paths [2].

In the book “Pattern Recognition and Machine Learning” (2006)[2], Bishop outlines the mathematical representation of an MLP, mentioning that such procedure can be described
as a series of functional transformations. The formulas 2.1 to 2.5 highlight an example of a forward pass in a two layer MLP. The visual representation of said procedure can be found on figure 2.4.

The first step of a forward pass through an ANN is to create a linear combination for each of the variables found in the input vector $\overrightarrow{X}$. This can be seen in equation 2.1, where the superscript (1) indicates it is the first hidden layer of the network. ANN refer to their layers as either being input, output or hidden.

The first element inside the sum $w_{ji}^{(1)}$ is the weights between two nodes while the second element $w_{j0}^{(1)}$ is the bias parameter for each node.

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} .$$

(2.1)

The resulting $a_j$ combinations are called the “activations” which are then passed on to a differentiable, non linear, activation function $h()$ such as the Sigmoid or ReLu functions.

$$z_j = h(a_j).$$

(2.2)

These result of the activation function are then used as inputs for the next hidden layer, which behaves just like the first layer.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} .$$

(2.3)

Finally the resulting activations are transformed using an activation function chosen taking into consideration the nature of the data and the assumed distribution of the target variables. For binary classification the Sigmoid function is used to arrive to a final solution, while the Softmax activation is used for multiclass problems.

$$Y_k = \sigma(a_k).$$

(2.4)

The past equations can be combined into equation 2.5 to show all of the steps in a forward pass. As mentioned before the visual representation of this can be observed in figure 2.4.

$$y_k(x, w) = \sigma \left( \sum_{j=1}^{M} w_{kj}^{(2)} \left( \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right).$$

(2.5)

2.2.1 Training of Artificial Neural Networks

The training of ANN consists on an iterative procedure that aims to minimize a cost function. This is carried out by making small adjustments to the weight vector for each unit. This procedure consists in finding the derivatives of the error function with respect to the weights which are then used to compute the needed adjustments to the weights that will further minimize the cost function [2].
2.3 Deep Learning

As mentioned in section 2.2 a MLP consists of many neurons or nodes ordered in multiple sequential layers. Up until recently these layers had been shallow. In other words, they employed only a few hidden layers between the input and output layers. However, thanks to advances in computing power the number of hidden layers increased. This gave birth to the concept of Deep Learning (DL).

A Deep Learning architecture is comprised of multiple hidden layers, most of them being able to learn non-linear mappings. An architecture made up of 5 to 20 layers can implement extremely complicated functions that are both sensitive to minute details as insensitive to large irrelevant variations [29]. This being extremely helpful for image analysis, where minute details are needed and large variations, such as background, rotation or scaling, are irrelevant. This section introduces Convolutional Neural Network (CNN) a popular DL architecture for image analysis work.

2.3.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) are a DL method able process data with a grid-like structure, such as time series, images or video frames. During the past 5 or so years they have gained tremendous popularity thanks to their performance on the ImageNet challenge and other similar image analysis related challenges. These challenges include detection and segmentation on popular massive datasets such as ImageNet [7] and COCO [30].

Such as with ANN, CNNs are inspired by biology and work in a similar way. They work by limiting the view of each unit within the network so that it’s only connected to local receptive fields from a prior layer. Each of these units learn to activate themselves when a particular element is detected. This is commonly done using the ReLu activation function. Each layer within a CNN learns to detect different elements. For example the first units get activated when a non-task specific low-level feature, such as an edge or a corner, is detected. While following units learn to detect combinations of elements found in past layers to detect more complex representations such as the presence of a particular object. CNNs main strength is that the layers do not have to be explicitly told what to look for in an input. Each unit within layer is capable of learning and adjusting it’s own weights so that the whole network can learn the correct combinations to detect the needed features for a particular task [61].

The convolution operation is the principal component in a CNN. A CNN can be defined as a ANNs that employ the convolution operation instead of matrix multiplication in at least one of their hidden layers [11]. During this subsection the convolution operation is described. The pooling operation and upsampling techniques, particularly important for segmentation tasks, are also briefly mentioned. The subsection ends with a general outline of a CNNs architecture.

A convolution is an operation on two grid-like inputs that provides a filtered version of one of the two as its output. It is typically denoted with an asterisk, the following formula denotes a 1D convolution:

\[ s(t) = (x * w)(t). \]  \hspace{1cm} (2.6)
where $x$ is commonly referred to as the input and $w$ as the convolution kernel and the output $s(t)$ is called the feature map, where the value $t$ denotes a point in time. However for this project we want to focus on 2D convolutions. Using image $I$ as input and a 2D kernel $K$ we can define the convolution operation as [11]:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n).$$  \hspace{1cm} (2.7)

A convolution operation on a 2D input with a 2D kernel can be seen on figure 2.5. This figures shows a valid convolution, this means that the output is restricted to positions where the kernel is completely within the image dimensions [11]. The use of valid convolutions naturally changes the output dimensions from those of the input. A convolution that takes into consideration positions that cover only a part of the kernel and thus do not change the output dimension to that of the input are said to have a padding of type same. A convolution of type same pads the outside of an input with zeros as to conserve the same the input dimensions on the output. Normally after a convolution operation the output is activated via an non-linear activation function such as ReLu.

Figure 2.5: Example of a valid convolution operation. Taken from Goodfellow et al. (2016) [11].

Goodfellow et al. (2016) [11] highlight three important ideas that improve a machine learning model which are present in CNNs, they are: sparse interactions, parameter sharing and equivariant representations. Sparse interactions refer to the ability of the neurons to not interact with every input. Normally a ANN has neurons that interact with all the input neurons, by using a kernel smaller than its input CNNs achieve sparse connections. Parameter sharing refers to the use of the same parameters for more than one function inside a model. CNNs use the same kernel with the same weights for each operation within an input image. Finally the concept of equivariant representations means that if the input is transformed in some way the output should also be transformed in the same way. In other words if we modify an image
$I$ and calculate the result of a convolution operation $S$ the result would be the same as if we used an unmodified image $I$ and then modified the output $S$ the same way.

The pooling operation, is one of the most important operations in a CNN. It consist of replacing the output of the network at a specific layer with a summary representation of itself, for example one can take the max pooling operation to take the maximum element within a region, or the average pooling to take the average instead. This operation is critical to reducing the input size and to making the network invariant to small artifacts [11].

Upconvolution and upsampling, are techniques to connect coarse outputs to dense pixels. This can be done by upsampling, using for example bilinear interpolation, or by having an upconvolution (deconvolution, transpose convolution) that has learnable weights[34].

A typical CNN architecture is made up by stacking multiple convolution layers [11]. Such as the one that can be found in figure 2.6. This figure shows a relatively simple convolutional layer but other architectures employ multiple convolutional operations before the pooling step, or they employ more complicated mechanisms such as skip connections. However, most CNNs architecture do follow this structure as a general guideline.

![Complex layer terminology](image)

Figure 2.6: Example of a typical CNN layer, the convolution, activation and pooling operations are commonly refereed to as a convolutional layer. Taken from Goodfellow et al. (2016) [11].

It is worth noting that even though CNNs were introduced as early as 1989, it was not until 2012 that they began to become the current state of the art for these image related applications. After being used to win the ImageNet challenge [7] and excelling in different challenges such as object detection (mitosis detection in breast cancer) and segmentation (animal brain segmentation) [46]. Since then they have obtained massive popularity. They have been used in a wide range of domains, such as: manufacturing, health care and military. Naturally research surrounding applications for CNNs is rapidly growing and changing.
2.4 Summary

This chapter provided the necessary theoretical framework required to understand the work presented in this thesis. It gave a quick overview on Machine Learning and its subsets. This was followed by an explanation of ANNs, which included an introduction to their training procedure. Finally, the concept of deep learning and convolutional neural networks were examined in a deeper level.
Chapter 3

Alzheimer Disease Classification

3.1 Introduction

Alzheimer’s Disease, from here on out referred to as AD, is the most common neurodegenerative disease. Its pathology normally consists in the entanglement of neuronal fibers and abnormal deposits of amyloid plaques, which lead to memory loss and deterioration of other cognitive functions [36]. AD is the most common form of dementia having between 60 and 80% of the 44 million worldwide dementia cases attributed to it [36]. Up to now the diagnosis of AD has been based on exclusion of other conditions, furthermore physicians are frequently hesitant to diagnose dementia caused by AD. This uncertainty and unwillingness to diagnose leads to a late diagnosis that comes 2 to 3 years after the appearance of the first symptoms [43]. The need to control the economic and social cost caused by AD has lead to a rise in interest surrounding clinical trials that could modify the disease. Nevertheless, the difficulty of successfully identifying patients that suffer from the disease and also patients in the early stages of it has limited the ability of obtaining meaningful results [37].

The Alzheimer Disease Neuroimaging Initiative (ADNI), a longitudinal multicenter study, designed to developed clinical, imaging, genetic and biochemical biomarkers for the early detection and tracking of AD [23], has had an essential role in AD focused research; In a study evaluating different algorithms for computer aided diagnosis of dementia, using structured MRI, it was found that most used data granted by ADNI [3]. Thanks to this initiative the development of both classification and prediction models has been greatly facilitated. It’s worth mentioning that most of these models are based around magnetic resonance images (MRI), as it’s a non-intrusive method that offers a large amount of information [6].

Another important initiative that has also driven the development of methodologies for AD’s prediction is “The Alzheimer’s Disease Prediction of Longitudinal Evolution” (TADPOLE) challenge. This challenge arises from a collaboration between the EuroPOND consortium and ADNI, it involves the use of historical patient data to make a prediction that tries to identify which at-risk individuals will show AD symptoms within a short-medium term (1-5 years) [35]. The challenge encourages the use of different strategies such as manual predictions by experts, use of machine learning or the use of regression to achieve a statistical prediction.

Convolutional Neural Networks (CNN) have achieved incredible advances in the field of computer vision. Their success has been propelled by to the creation of large open annotated
image datasets, the availability of GPU computing and the popularity of Deep Learning of which CNNs are a part of [50, 52]. The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [42] has also been fundamental to their success. This yearly challenge seeks to find the best algorithms for object detection and image classification while providing participants with a large training labeled dataset. CNNs have dominated the competition during the last years, ever since AlexNet won the ILSVRC-2012 competition with a top-5 test error rate of 15.3%, beating the second best entry by more than 10% [27]. Since then the yearly winners of the ILSVRC have all used CNNs and there exist publicly available trained weights for most of the models which now serve as an important part in the improvement of image and object classification models that work with different datasets [50].

The goals of this research are twofold. First it intends to develop a Deep Learning model capable of classifying patients as either being affected by AD or as being part of the normal control (NC) group. This will be attempted using only a single 2D Axial MRI slice to keep training times and computational power at an acceptable level. Secondly, it will include the use of pre-trained weights, from a non-medical domain, trained using 2D images from ImageNet. The goal of this is finding out if pre-learned features from a non-medical domain can aid in the AD classification problem. To achieve this data provided by ADNI and pre-trained weights from the ImageNet challenge will be used.

As mentioned one of the goals of this research is the use of pre-trained weights trained with non-medical images. There are 3 main protocols when fitting a CNN model to a different dataset to that with which was it originally trained for:

- Training the model from scratch with random initial weights also referred to as “training from scratch” [50].
- Using a pre-trained model with “off the shelf” weights as a feature extractor which is then introduced as an input to another classifier [50, 57].
- Fine Tuning certain layers of a previously trained model while keeping other weights as they were previously trained [50, 62, 55].

Throughout this work these approaches will be respectively referred to as: training from scratch, Transfer Learning and Fine Tuning.

This chapter has the following structure: the first section offers a review of related work. This is followed by an overview of the methodology used. Then the results obtained are presented and discussed. Finally the AD classification project conclusion is given.

### 3.2 Related Work

**Residual and Plain Convolutional Neural Networks for 3D Brain MRI Classification:** Korolev et al. (2017) [26] argue that recent ML applications on neuroimaging data generally require multiple pre processing steps, but that recent advances allow for DL models to skip those steps. Since the DL networks are capable of extracting them themselves. They present two 3D CNN architectures used for six binary classification tasks between four different categories (AD, NC, EMCI and LMCI). To train their models they used 231 images taken from...
ADNI. They also employ a plain CNN and a 3D implementation of the ResNet model, which employs skip connections. Their best accuracy was reached by comparing AD vs NC, where they plain model achieved an accuracy of 0.79 (AUC 0.88) and the Resnet model achieved an accuracy of 0.80 (AUC 0.87).

**Deep Fusion Pipeline for Mild Cognitive Impairment Diagnosis:** Senanayake et al. (2018) [47] propose a method that fuses 3D MRI data with 1D Neuropsychological based features to classify a subject into one of two classes. To achieve this they propose a Deep Learning method that combines a CNN with a fully connected network, the challenge of combining the features extracted from the 3D image input is solved by using 1x1 convolution, a method for dimensionality reduction of feature maps while maintaining spatial information. The result of the 1x1 convolution is later flatten and concatenated to an intermediate layer of the fully connected network. Effectively fusing features extracted from both input sources. The researches used ADNI data for their work. Their dataset consisted in 515 MRI volumes from three different classes AD, MCI and cognitively normal (CN). Their method was compared to the one proposed by Korolev et al. (2017) [26] for the case of Normal vs AD and it was unable to beat their results. However, they report that the fusion of input data improves the performance over the conventional 3D CNN model. It’s worth mentioning that they do not cross validating their approach, mentioning that the time complexity of training multiple folds was too high.

**Multi-class Alzheimer’s disease classification using image and clinical features:** Altaf et al. (2018) [1] also propose a method capable of combining MRI Images with clinical data. However, they achieve this without resorting to Deep Learning. Instead they use a pre processing step on the image to extract Scale Invariant Feature Transform (SIFT), Local binary pattern (LBP) and Histogram of oriented gradient (HOG) features to form a Bag of Words model that can extract texture features from an image. This features together with a Gray-level co-occurrence matrix (GLCM), also extracted from the image, and the fusion of the clinical features is then added to multiple classification methods, such as SVM, K-NN, Decision Trees and Ensemble methods, to arrive to a final classification. The authors report their result on different binary classification as well as on a multi-category classification. The dataset they used was obtained from ADNI and it consisted on 287 subjects. They report results for 3 different experiments. First they try to use the extracted texture features and the GLCM to classify subjects reporting sub optimal results, i.e. for the AD vs NC classification no feature was able to arrive to an accuracy over 0.60. Their second experiment combines GLCM features with clinical features and this reaches an accuracy of 0.97 by the K-NN classifier. Finally they concatenated clinical features with one of four segmented images of areas of interest (grey matter, white matter (WM), cerebrospinal fluid (CSF) and the combination of them all), all of these combinations were able to reach an accuracy of over 0.98 for the AD vs NC classification task.

**Applying Convolutional Neural Networks for Pre-detection of Alzheimer’s Disease from Structural MRI data:** Using 818 MRI scans from ADNI, Lin et al. (2018) [32] trained a model capable of reaching an accuracy of 0.79 for the task of predicting whether a progressive MCI (pMCI) or an stable MCI (sMCI) subject would convert into AD. They achieve this by designing a Deep Learning model that first take NC and AD MRI images and extracts local patches from each of the anatomical views to assemble a 2.5 dimension patch. These patches
are then used to train a CNN that extracts a feature vector of size 1,024 for each 2.5D patch in an image. This feature extraction CNN is then used to extract features from images belonging to MCI subjects. These features are bundled with brain image features mined from FreeSurfer and feed into an extreme learning machine classifier, a feed-forward artificial neural network with a single hidden layer of nodes, to predict the AD conversion. The training of this extreme learning machine is validated via leave one out cross validation, which is ran 308 times, once for each image of the pMCI and sMCI subsets.

### 3.3 Methodology

This section presents the methodology followed specifically for the task of classifying a patient to either belonging to the AD or NC group. The section provides information on the data acquisition, the implementation of CNNs and on the design of the three different experiments carried out.

#### 3.3.1 Data Acquisition

Data used in the preparation of this work was obtained from the Alzheimer’s Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). The ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer’s disease (AD).

The subject data was retrieved from the section Download/Study Data/Test Data/Data for challenges/Tadpole Challenge, specifically from the TADPOLE_D1_D2 dataset. It is worth noting that the ADNI cohort includes data from different sources and even though they try to standardized their protocols there might be differences in the acquisition procedures carried out for each subject.

The Tadpole dataset was used to identify patients that belong to the AD or NC classes. Following the subject identification task the latest MRI Axial T2 FSE/TSE weighted image was downloaded for each of the subjects.

Once all of the MRI images had been downloaded a single slice was then selected for each subject. Only a single 2D slice per subject was used as to maintain computational requires at a reasonable level. The use of 3D CNNs would have required additional hardware not available at the time of this research. Also a positive side effect of using 2D MRI slices was that the 2D CNNs could be initialized with weights learned from a different domain. The procedure for selecting a 2D MRI slice per subject followed a straight forward heuristic. The slice was extracted at eye level aiming to have the head of the hippocampus present, as its atrophy is a pathological criteria for AD diagnosis.

The final step during the data acquisition pipeline was to convert the image slices from DCM to PNG. This was done so that they could be easily feed into the pre-trained CNN. In the end 379 2D images were obtained, an overview of this data can be seen in table 3.1.
3.3. METHODOLOGY

Table 3.1: Overview subject distribution.

<table>
<thead>
<tr>
<th>Number of AD Subjects</th>
<th>169</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of NC Subjects</td>
<td>210</td>
</tr>
</tbody>
</table>

Figure 3.1: Summary of transfer learning model.

### 3.3.2 Implementation of Convolutional Neural Networks

The implementation of the different CNNs was carried out using the Python library Keras [5]. Keras is a high-level artificial neural network API that runs on top on TensorFlow. Its main advantage is that it enables fast experimentation and is user friendly. Keras provides pre-implemented Deep Learning models offering the use of pre-trained weights. The out of the box provided models include: Xception, vgg16, vgg19, ResNet50, InceptionV3, InceptionResNetV2, MobileNet, DenseNet, NasNet and MobileNetV2. All of these models can be trained from scratch, or “off the shelf” using the pre-trained weights for either Transfer Learning or Fine Tuning [5]. During this research project vgg16 was used as a base for the experiments carried out, as it proved to offer a better accuracy than MobileNet and ResNet50 during early experimentation.

Three different CNN models were designed for this experiment. The first model followed a Transfer Learning protocol in which pre-trained and non-trainable (frozen) weights from ImageNet were used together with a vgg16 model as a feature extractor. The extracted feature vector was subsequently flattened and fed into a fully connected layer with 512 units connected to an output layer which gave a final prediction using the Softmax function. A diagram of this model can be seen in figure 3.1. The second model, which can be seen in figure 3.2, follows a Fine Tuning approach that allows the last \( N \) pre-trained layers of the vgg16 model to be trained. This approach also uses a fully connected layer. However, this layer uses 1,024 units, and is then connected to an output layer that provides a prediction by using the Softmax function. Finally, the last model is a “shallower” custom model. It’s diagram can be found in figure 3.3. The custom model uses 4 convolutional layers which then connect to
20

CHAPTER 3. ALZHEIMER DISEASE CLASSIFICATION

Figure 3.2: Summary of fine tuning model.

Table 3.2: Comparison between number of parameters in each model

<table>
<thead>
<tr>
<th>Model</th>
<th>Total Parameters</th>
<th>Trainable</th>
<th>Non-Trainable</th>
</tr>
</thead>
<tbody>
<tr>
<td>vgg16</td>
<td>14,714,688</td>
<td>14,714,688</td>
<td>0</td>
</tr>
<tr>
<td>Transfer Learning</td>
<td>27,561,282</td>
<td>12,846,594</td>
<td>14,714,688</td>
</tr>
<tr>
<td>Fine Tuning (last 4 layers)</td>
<td>40,407,879</td>
<td>32,722,610</td>
<td>7,635,264</td>
</tr>
<tr>
<td>Custom</td>
<td>1,220,994</td>
<td>1,220,994</td>
<td>0</td>
</tr>
</tbody>
</table>

2 fully connected layers before giving a final prediction. To help regularize the networks all these models made use of a dropout layer, with a dropout rate of 0.5%, right before the output layer.

An overview of the total number of parameters of each model is shown in table 3.2. This table also shows the relationship between the number of non-trainable parameters for Transfer Learning and the vgg16 base model, as well as the relationship between “deepness” and total parameter size. For the Fine Tuning model a variation which only trains the last four layers is shown.

3.3.3 Basic Training

The first experiment consisted in running a single simple training procedure for each of the models. For this the dataset was divided into training and validation subsets consisting on 80% and 20% of the subjects for each class respectively. Table 3.3 gives an overview on the size of these divided sets. All of these training procedures were carried out using the exact same dataset, which was made up of 379 subjects. As mentioned for each subject only a single slice of Axial MRI image was selected and used. The models all used RMSprop as an optimizer using a learning rate of 0.0001. The vgg16 model was used as the base model for the Transfer
Figure 3.3: Summary of custom classification model.
Learning and Fine Tuning models, where it was loaded with the pre-trained weights trained on the ImageNet dataset. For the Transfer Learning model only the extra densely connected network was trainable, while the complete vgg16 model was kept frozen with the pre-trained weights. In the case of the Fine Tuning model the extra densely connected network along with the last 4 layers of the vgg16 model were unfrozen and trainable and the rest of the vgg16 model was kept frozen with the ImageNet pre-trained weights.

As mentioned, the training was carried out using the DL python library Keras. The models were trained for 50 epochs or until the validation accuracy did not improve for 10 consecutive epochs. This was controlled using Keras early stopping callback function. An important goal of this experiment was to visualize the learning behavior of each model during their training procedure, this was done by plotting the model’s loss and accuracy for training and validation across the training epochs.

3.3.4 Finding Optimal N for Fine Tuning

Using Keras the Fine Tuning training works by toggling a parameter for each of the models’ layers, and either setting them as trainable. Normally the last layers are the ones that are retrained while earlier ones are kept frozen. Previous work has shown this to be effective as generally the first layer of a CNN learn low-level image features, transferable to most vision tasks, while later layers learn higher-level features specifically related to the task at hand for which they were trained for [55]. However, the question of how many layers should be trained of the base vgg16 model to achieve a desired performance for this classification problem remained unanswered. To find the optimal number of layers that should be trained the Fine Tuning model was trained five different times, with a different $N$ number of trainable layers each time. The number of last $N$ layers that were unfrozen and set as trainable were: 4, 8, 12, 16 and 18, training the last 18 layers is equivalent of training the whole network. The numbers of $N$ last layers were chosen as they were clear points between convolutional layers and other layers e.g. dropout and max pooling layers.

It’s important to note that this experiment was carried out using a GPU, which significantly improved the training time from that of a CPU. This experiment was carried out using the same training and validations set for each N and as with the Basic training experiment. All of the models employed the RMSprop optimizer with a learning rate of 0.0001 and the early stopping callback in order to optimize the training time. Because of time constraints the experiment was only carried out once, i.e. only one training procedure was carried out for each of the five possible N values.
3.3. METHODOLOGY

3.3.5 Model Validation

To validate the predictive capabilities of the model a thorough experiment had to be carried out. As the previous experiments only executed a single training procedure, without making use of any validation technique. To be confident of the model’s classifying capability a 10 times repeated 5-fold cross validation was carried out for each of the three models. For the Fine Tuning approach only the last 4 layers were trained as this proved itself to be a good enough number of layers to achieve a good accuracy without needing a large training time. The k-fold cross validation provides k different performance scores that can be used to determine how the model will behave with unseen data. This cross validation method required all of the labeled data to be divided into five subsets so that five training runs could be carried out, using four subsets for training and one for validation. This also ensures that each subject forms part of the validation subset exactly one time, and is used for training during the other four training procedures.

The model validation method was carried out 10 times, providing 50 different models and performance scores. For each of the 10 iterations each subject received one “blind” prediction as they were part of one of the validation subsets. These subject-specific predictions were saved for the duration of the experiment, so that once finished each subject would have 10 different predictions from 10 different models. These 10 different predictions included the probabilities of belonging to each class. Using these probabilities an ensemble classifier was created which used the average of the 10 predictions to determine the subject’s final class. Figure 3.4 shows a representation of this method. For each training procedure the weights that performed the best were saved and later employed for the classification task and that the early stopping technique was used to end the training if the validation accuracy stalled.

To carry out the K-fold cross validation split the python’s library scikit-learn was used. This library offers various tools for data analysis. It was also used to calculate the accuracy, Area under ROC Curve (AUC) and Balanced Error Rate (BER) validation metrics, and to plot the ROC curves for each of the 3 models final “ensemble” predictions. The accuracy reported per model refers to the number of correctly classified subjects divided by the total number of subjects, the reported accuracy takes into consideration only the final classification obtained by averaging the 10 probabilities predicted per subject.
CHAPTER 3. ALZHEIMER DISEASE CLASSIFICATION

3.4 Results

3.4.1 Basic Training

For the basic training experiment three simple training procedures were carried out, one for each model. They all used the RMSprop optimizer with a learning rate of 0.0001, which was empirically found to provide decent results. The training visualization of each of the models can be seen on figures 3.5, 3.6 and 3.7.

Figure 3.5 shows the accuracy and loss curves across training epochs for the Transfer Learning model, which only uses the frozen vgg16 model as a feature extractor and employs a second trainable network to arrive to a classification. From these graphs it can be noted that the training accuracy improves constantly achieving an accuracy of almost 0.9, however the maximum accuracy reached by the validation set is just over 0.75. The validation accuracy also shows a different behavior throughout the epochs to that of the training accuracy, the validation accuracy fluctuates greatly from below 0.5 (random guess) to over 0.7, and does not show a steady continues progress, whereas the training accuracy steadily increases throughout the training procedure. Comparing the maximum accuracy reached by the training set and the validation set it can also be noted that the accuracy scores differ greatly. A similar behavior can be observed from the loss progress, the training loss shows a steady decline while the validation loss stalls and shows a different progress to that of the training loss.

Figure 3.6 shows the visualization of the accuracy and loss curves through the epochs of the Fine Tuned model training procedure. It is evident that both the accuracy and loss curves for the two sub sets (training and validation) show a similar growing behavior. One can see that even though the training accuracy shows a more uniform increment the validation accuracy constantly grows and achieves a similar performance. It is also evident that fluctuation in validation accuracy scores progression is also present as was the case with the Transfer Learning model. However, the validation accuracy does not stall as much and it does not get as fat behind as is the case of the Transfer Learning model in figure 3.5. Instead a growing pattern can be noted. Nevertheless the model does stall and is early stopped after 23 epochs, the model was able to reach a maximum validation accuracy of 0.77. The loss curve also shows a similar behavior constantly following the training loss behavior up until the last epochs where it stalls.

Figure 3.5: Accuracy and loss progression across epochs for Transfer Learning method.
3.4. RESULTS

Figure 3.6: Accuracy and loss progression across epochs for Fine Tuning method.

Finally figure 3.7 shows the accuracy and loss curves for the training of the Custom CNN model. The validation behavior for both training and validation show an increasing behavior, in this case both the training and validation accuracy sway greatly between epochs, still, they also show a steady incremental conduct. The difference between the achieved accuracy by the training and validation sets is smaller than in the other two cases. The loss curves also show a steady decrease for both the training and validation loss, the difference between the two different sets is also minimal. This model was the only one to exceed 30 epochs before stopping and was able to reach a maximum validation accuracy of 0.73.

3.4.2 Finding Optimal N for Fine Tuning

The different training times for and max accuracy reached by the Fine Tuned model for each of the \( N \) number of trainable layers is shown in table 3.4. Also shown in said table is the total number of vgg16 trainable parameters for each \( N \). It is worth remembering that training 18 layers retrains the whole network. One can naturally observe that a larger number of parameters would leads into a longer training time. The max accuracy reached for each of the \( N \) number of layers is also shown in table 3.4, the highest accuracy (0.74) was reached by training the last 12 layers. The second highest accuracy was of 0.72 and was reached by only training the last 4 layers. Training the whole model (18 layers) and the last 16 layers
Table 3.4: Comparison between number of trainable layers for vgg16 for the Fine Tuning approach.

<table>
<thead>
<tr>
<th># of Trainable Layers</th>
<th>Training Time</th>
<th>Max Accuracy Reached</th>
<th># of vgg16 Trainable Params</th>
<th># of vgg16 Non-Trainable Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>154s</td>
<td>0.72</td>
<td>7,079,424</td>
<td>7,635,264</td>
</tr>
<tr>
<td>8</td>
<td>114s</td>
<td>0.68</td>
<td>12,979,200</td>
<td>1,735,488</td>
</tr>
<tr>
<td>12</td>
<td>190s</td>
<td>0.74</td>
<td>14,454,528</td>
<td>260,160</td>
</tr>
<tr>
<td>16</td>
<td>284s</td>
<td>0.71</td>
<td>14,675,968</td>
<td>38,720</td>
</tr>
<tr>
<td>18</td>
<td>293s</td>
<td>0.71</td>
<td>14,714,688</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.5: Model validation results for each of the three used models, with their corresponding 95% CI range.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>AUC</th>
<th>BER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer Learning</td>
<td>0.77 [0.71, 0.80]</td>
<td>0.81 [0.77, 0.86]</td>
<td>0.25 [0.20, 0.29]</td>
</tr>
<tr>
<td>Fine Tuning</td>
<td>0.80 [0.76, 0.84]</td>
<td>0.86 [0.82, 0.90]</td>
<td>0.20 [0.17, 0.25]</td>
</tr>
<tr>
<td>Custom</td>
<td>0.70 [0.65, 0.74]</td>
<td>0.71 [0.66, 0.77]</td>
<td>0.32 [0.27, 0.36]</td>
</tr>
</tbody>
</table>

yielded similar results both reaching a maximum accuracy of 0.71 and both needing around 290 seconds to train. As mentioned before, this training time comes from training using a GPU.

3.4.3 Model Validation

The results for the 10 x 5-fold cross validation can be found on table 3.5. The highest accuracy was reached by the Fine Tuning model obtaining an accuracy of 80%. The Transfer Learning protocol came in second place reaching 77% accuracy. The Custom model was only able to reach an accuracy of 70%. As previously mentioned, this accuracy was calculated as the total number of correct predictions divided by the total number of subjects. To make the subject specific predictions each subject considered its 10 blind predictions obtained from every one of the 10 iterations of the 5-fold cross validation. Table 3.5 also shows the AUC and BER for each model along with their 95% confidence interval values. The model with the highest AUC was also the Fine Tuned model reaching an AUC score of 0.86, while the Transfer Learning based model achieved an AUC score of 0.81 and the Custom model 0.71, the Fine Tuning model was also the one with the lowest BER with 0.20.

3.5 Discussion

3.5.1 Basic Training

Regarding the first experiment which consisted in having the three different models trained, one can see that the Transfer Learning model clearly overfitted. This can be appreciated in figure 3.5. It is clear that the validation accuracy stalls even though the training accuracy keeps
improving throughout the epochs, also when the early ending function is ignored the training validation goes up to an accuracy of 100%, a perfect score, meaning that the model learned to perfectly classify the images used for training, while the validation accuracy stalls and has sudden large changes. Also the difference between the maximum validation accuracy score and the maximum training accuracy score is almost 20%. This is a clear sign that this model has overfitted to the training data. This could be attributed in part to the limited dataset size and also to the relatively small trainable classification network. Regarding the Fine Tuning (figure 3.6) and the Custom model (figure 3.7). We can see that overfitting is not as evident. Both models are capable of learning on a similar rate as that of the training set. However, at the end of the training it is clear that the validation accuracy stalls while the training accuracy continues to increase, another sign of overfitting. Overall the best performance was achieved by the Fine Tuning model. It was the model that most closely followed the training accuracy and also the training loss, and in contrast to the Custom model there is less variance between the accuracy scores. In other words the low and high peaks are kept in a smaller range than those of the Custom model.

### 3.5.2 Finding Optimal N for Fine Tuning

As seen in table 3.4 there is a clear trade-off between model size (number of trainable parameters) and the training time needed to train a model. This was to be expected. However, a rather unusual finding is that the accuracy of the model does not show a considerable increase as more parameters are trained. The average maximum accuracy reached by the different N values was of 0.71, and with the fastest training time ($N = 8$) the model achieved an accuracy of 0.68. However, the speed of this model could be attributed to the early stopping function triggered if the model stalled. To get more trustworthy results this experiment should be carried out more times and have the accuracy and time represented as the average across those iterations. Nevertheless, it’s clear that for this data set an N of 4 is enough to achieve promising results without compromising the needed training time.

### 3.5.3 Model Validation

Finally, for the validation of the models a 5-k fold cross validation was carried out 10 times, so that for each of the 10 iterations a total of 5 different models were trained for each of the 3 proposed protocols, and each model gave a prediction for all of the subjects in their validation set. This meant that every one of the 377 subjects would get exactly one prediction per iteration. After concluding the experiment each subject had a set of 10 different predictions. These predictions were then averaged to arrive to a final prediction, the accuracy scores reached by each of the models can be found on table 3.5. The highest accuracy score was achieved by the Fine Tuned model with an accuracy of 80%, which can be compared to previous binary classification scores reported [26, 47, 32]. This result is followed by the Transfer Learning model reaching an accuracy of 77% and the custom model being only able to achieve 70%. All of the models were capable of outperforming the random guess accuracy, which allows us to confirm that these deep learning CNNs models are capable of discriminating between AD and NC classes. We can also attest that the used of pre-trained ImageNet weights for medical image classification is proven to be useful to improve the overall performance of a CNN, even
though the original training task, of 1,000 different object classes, is extremely different to that of AD classification.

Figure 3.8 shows the ROC curves for each of the proposed models. These curves were plotted using the average probabilities predicted for each of the subjects. It’s easy to see that all of them outperform the random guess, and that the Fine Tuning approach proves to be the overall best model for the problem of AD classification.

These results are consistent with those found by Tajbakhsh et al. [55], where they found that the Fine Tuning protocol of CNNs was useful for medical images, even when using pre-trained weights of a largely different dataset, showing that the transfer of knowledge from natural images to medical images is possible. They found that CNNs that benefited for pre-trained weights could perform as well as fully trained nets and even managed to outperform fully trained nets when the training data was limited. Similar results have been presented from the experimentation carried out during this work, which allows us to confirm that the Fine Tuning of pre-trained CNNs weights have a high transfer capability that can be helpful on very different classification tasks. Similar results have also been reported in the work of Shine et al. [50], where the use of previous knowledge from ImageNet fined tuned on the target medical images also proved to be beneficial for computer aided detection applications.
3.6 Conclusion

The two main goals of this chapter were to determine if single slice of an Axial MRI image could be used to train a CNN so that it could correctly classify between AD and NC subjects. The second goal was to figure out if the use of CNNs using pre-trained weights from ImageNet could improve the classification ability, even when the original training dataset greatly differs from the current data set consisting on MRI images. Both of this goals were met as it was proved that a data set consisting of single slice MRI Axial image could be enough to train a CNN and achieve an accuracy better than that of a random choice. It was also found that the use of Fine Tuning where the last 4 layers of the vgg16 model were unfrozen and the rest of the model remained frozen using the pre-trained ImageNet weights, posed an advantage in the training of the models. Further work remains to be done, such as introducing the progressive Mild Cognitive Impairment (pMCI) and stable Mild Cognitive Impairment (sMCI) classes and working with classifiers that use whole 3D MRI images to arrive to a final prediction.

3.7 Summary

During this chapter the problem of AD classification was examined. The chapter began with an introduction to the problem at hand and the exposition of related work. This was followed by the proposed methodology. The methodology section included information on data acquisition, use of deep learning in three different deep learning networks, experiment specification and model validation techniques. The obtained results were then showed and discussed, the main findings being that it was possible to classify patients into AD or NC control groups by using only a single MRI image and that the use of features pre learned on a non medical domain posed an advantage in model training. The chapter ends with a conclusion on the work carried out.
Chapter 4

Whole Fetal Brain Segmentation

4.1 Introduction

Ultrasound screening is the most widely used prenatal imaging technique, the reasons behind this are its inexpensive costs, its real time results and the safety provided to both fetus and mother. However, the ultrasound results are not always able to accurately depict enough information about the fetal brain and its development [53]. For this Magnetic Resonance Imaging (MRI) is considered a better approach. MRI is useful for both monitoring the development of the fetal brain and to confirm suspected abnormalities that are commonly previously found in ultrasound scans [44, 51, 53, 54].

Even though MRI is commonly used to research the fetal brain throughout the gestational age. The recollection of such scans is difficult, as the fetal brain MRI acquisition process greatly differs from that of a brain scan in a neonate, an infant or an adult subject [22]. The biggest challenges during MRI scanning of the fetal brain revolve around the movement of the fetus, the brains position and orientation, and the undesired inclusion of the surrounding maternal tissue and organs [22, 44, 21]. Some of these difficulties are mitigated by the use of fast 2D snapshot imaging. In which a stack of 2D slices are acquired in quick succession to construct a whole 3D image in which every slice is generally artifact free. However, the full 3D representation is typically not fully coherent [44, 54, 21]. To resolve this incoherence there exist motion correction methods which require only the fetal brain region as an input [44, 21]. To create this input a fetal brain mask which identifies the region of interest must be created. This mask not only helps with the correction of inter-slice motion but as direct consequence also discards the maternal surrounding tissue and organs that are undesired during the following research steps.

In the Fetal Neonatal Neuroimaging and Developmental Science Center (FNNDSC) at Boston Children’s Hospital the fetal MRI motion correction step also depends on the creation of a fetal brain mask. Typically produced manually by a human masker. This is both time consuming and biased towards the masker. Fetal Neonatal Neuroimaging and Developmental Science Center (FNNDSC) is not the only research center in need of faster, more consistent and unbiased fetal mask generation results. This need has driven research into both semi automatic and fully automatic fetal brain segmentation techniques. Section 4.2 provides an overview of some of the research surrounding this problem.

The aim of this work is to fully automate the process of generating a whole fetal brain
mask. To achieve this multiple deep learning approaches will be trained and compared. These models include U-Net \[41\], Fully Convolutional Network (FCN) with stride size 8 \[34\] and Mask R-CNN \[14\]. In addition the best working models will be modified as to use attention components, which come in the way of: The Squeeze and Excitation module \[16\] and attention gates \[39, 45\].

The Squeeze and Excitation module \[16\] makes models not only learn from spatial information, as most CNNs do. But are also capable of learning channelwise information, giving specific weights to particular channels or feature maps. On the other hand the attention gate module \[39, 45\] can be used within an encoder/decoder architecture, such as U-Net, by limiting the information that is shared between encoder and decoder as to only share relevant spatial information. This is done by using an attention gate that restricts the information flow from encoder to decoder so that only relevant spatial information is taken into consideration. The gate is created using information from lower dimensional layers just before the upsampling step. All of these processes are further explained in subsection 4.3.5.

### 4.2 Related Work

**Automated fetal brain segmentation from 2D MRI slices for motion correction:** Keraudren et al. (2014) \[21\] propose a method for fixing motion correction in fetal brain MRI. They counted with a total of 458 scans. The steps they followed were: localization of brain, segmentation of brain, motion correction and fine tuning. The first two steps directly relate to the problem at hand. To localize the brain they used a Bag of Words classification model. Trained on labeled images with a bounding box around the fetal brain, to detect the brain in 2D images. The Bag of Words model used SIFT (Scale Invariant Feature Transform) features as a vocabulary. The authors incorporated prior knowledge about the expected size of the fetal brain by having the SIFT features grouped within similar regions which were found using Maximally Stable Extremal Regions (MSER), a computer vision method to detect blobs within images. This allows the use of the expected brain size as a filter and produced bundled SIFT features. This bundle of features is then used to produce a histogram of words by matching each feature to its nearest neighbor within the vocabulary, this is then used to classify the bundle using a Support Vector Machine. They finish this process by producing a 3D cube around the localized fetal brain. This resulting area of interest and the partially segmented regions produced by MSER are then used during their second step, segmentation of the fetal brain. A patch based segmentation was used to register patches within an atlas. They used two kind of patches, those that fell on top of the partial segmentation (brain) and those that fell on the boundaries of the bounding box (non-brain). A mean Dice Score Coefficient (DSC) of 0.93 was reported for this step, to evaluate their results 10-fold cross validation was used.

**Automatic Brain Extraction in Fetal MRI using Multi Atlas based Segmentation:** Toubier et al. (2015) \[56\] use a template based approach to segment the fetal brain. This method that works by finding the transformation between the atlas reference and the target image. In this work the authors propose the combinations of Multiple Atlas based segmentation via a global weighted voting strategy. They compare their results to those obtained by using a single atlas to reach a segmentation. The validation of the method was carried out using leave
one out cross validation and they had 46 available image stacks. Their results show that single atlas brain extraction reaches a DSC of around 0.95 while the use of multiple atlas reached a DSC of around 0.96.

**Fetal Head Localization and Fetal Brain Segmentation from MRI using the Center of Gravity:** Somasundaram et al. (2016) [53] used 25 MRI volumes to propose a segmentation method based on the center of gravity. They first localize the fetal head in 2D MRI slice located at the middle of the volume by using the center of gravity. They assume that the brain would occupy the primary space of the image and thus use the pixels' intensity to find the brain, a circle is drawn around this area to denote a region of interest (ROI), discarding all of the elements that fall outside of it. This ROI is then processed via ”traditional” computed vision techniques. First unwanted surrounding tissues and high intensity pixels are eliminated as CSF and amniotic fluid appear brighter in the T2 MRI. Following Otsus method is performed to find a threshold that separates the pixels into foreground and background. Then a morphological opening is executed as to remove weakly connected tissues. Flood filling is then used to fill possible holes, finally the a connected component around the middle point is found and considered the final segmentation. Using the proposed method an average DSC of 0.88 was reached.

**Automatic Segmentation of the Intracranial Volume in Fetal MR Images:** Khalili et al. (2017) [22] used a patch wise multi branch convolutional neural network (CNN), to segment the fetal brain a 2D MRI slice. This model takes as an input three different 2D patches of different sizes, but all within the largest one. These patches are then fed to three different CNNs, which are merged after three convolutional and a fully connected layer into a final fully connected layer that classified the patch as either intracranial volume or negative classes. To produce the final segmentation only the largest 3D connected component was retained. To train their model the authors counted with 30 MRI volumes. They were able to reach a DSC of 0.91 for axial, 0.94 for coronal and 0.92 for sagittal images. No validation method such as cross validation was employed during this work.

**Fetal brain extraction from magnetic resonance image (MRI) of human fetus:** Somasundaram et al. (2018) [54] use an unsupervised method to automatically segment the fetal brain. Their propose method focuses solely on ”traditional” computer vision technique, similar to the second step in the work proposed by Somasundaram et al.. First they perform contrast enhancement to improve the contrast of the boundaries, this is then followed by Otusu’s threshold method, a seeded region growing around the center pixel and flood filling to fill holes inside the brain region, arriving to a final segmentation mask. Evaluating their method on 25 scans they were able to achieve a DSC of 0.90.

**Real-Time Automatic Fetal Brain Extraction in Fetal MRI by Deep Learning:** Most recently Sadegh et al. (2018)[44] used U-Net [41], a deep learning approach based in an encoder/decoder architecture, which has proven to be successful in general medical images segmentation [24, 60, 38] and also in the specific task of brain segmentation [18, 25]. The results achieved on the task of whole fetal brain segmentation were of a dice score of 0.96 and
0.78 for normal and challenging test sets respectively. During this work the dataset size was of 250 MRI volumes.

It should be pointed out that this work did not dive into the validation of the model, the process of manually constructing the training and validations masks nor did it try to modify the preexisting U-Net architecture. The authors made their trained model available to the public. However, when testing the model with the available FNNDSC data, the results proved that the model was unfit for fully automatic use within the research center. Nevertheless this previous work serves as a fundamental foundation for this current research work.

It’s interesting to note that none of these works reported a spatial distance metric such as HD. Also the work of Somasundaram et al. (2016) [53] and Somasundaram et al. (2018) [54] rely on the fact that the brain is the biggest organ in a scan and that it will be present around the center of the image. This is a dangerous assumption to make, as inter slice motion can produce a volume with middle slices that do not show a clear image of the brain and the movement of the fetus does not guarantee that positioning of the brain will always fall exactly in the middle of the image.

4.3 Methodology

This section presents the methodology followed specifically for the task of automatically segmenting the whole fetal brain. The section provides information on the data acquisition, the employed performance metrics to compare model performance and the training and cross validation procedures. The section ends with an overview of the different CNN segmentation architectures used and of the proposed attention modules.

4.3.1 Data Acquisition

The available data comes from the FNNDSC at Boston Children’s Hospital. The dataset is made up of 293 T2 weighted HASTE (Half-Fourier Acquisition Single-Shot Turbo Spin-Echo) MRI volumes taken from 65 subjects, ranging from 24 to 30 gestational weeks. The volumes have had a main orientation chosen so that the fetal brain is presented in the orientation it’s viewed the best, this could be axial, sagittal or coronal with respect to the fetus. This means that the available volumes come from all plane orientations. The distribution of how many volumes belong to each orientation can be seen in table 4.1.

To generate the masked dataset the volumes were manually masked slice by slice using the application ITK-SNAP [63]. These masked volumes served as the gold standard for the task of whole fetal brain segmentation. To provide a sense of the desired accuracy and also of the possible accuracy that could be reached. 30 of the volumes were also segmented by

<table>
<thead>
<tr>
<th>Plane Orientation</th>
<th>Number of Volumes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axial</td>
<td>94</td>
</tr>
<tr>
<td>Sagittal</td>
<td>99</td>
</tr>
<tr>
<td>Coronal</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4.1: Plane orientation distribution for the available dataset.
a second masker and compared against the first masker results to get an intra-rater reliability score. The dice score reached was of 0.89 and the Hausdorff 95 percentile distance was 3.89mm, this score can be seen as a benchmark of the desired performance. The metrics mentioned, dice score and Hausdorff distance, are presented in the following subsection 4.3.2.

### 4.3.2 Performance Metrics

The two main metrics that will be used throughout this work are the Dice Similarity Coefficient (DSC) and the Hausdorff Distance (HD).

**Dice Similarity Coefficient** measures the overlap between a produced binary segmentation A and a target binary segmentation B, it is defined as:

\[
DSC = \frac{2 \times A \cap B}{A + B}.
\]

(4.1)

The resulting DSC varies from 0 to 1, resulting in 0 when there is absolutely no overlap, to 1 in the case of a perfect overlap between A and B [13]. For this work the Python library SciPy [19] was used to calculate the DSC.

**Hausdorff Distance** is a metric that measures the extent to which each point of a produced segmentation A lies near some point of a goal segmentation B, it is defined as:

\[
H(A, B) = \max(h(A, B), h(B, A)).
\]

(4.2)

Where \( h(A, B) \) is defined as:

\[
h(A, B) = \max_{a_i \in A} \min_{b_j \in B} d(a_i, b_j).
\]

(4.3)

and \( d(a, b) \) is some distance function for comparing point a and b, this function measures the degree of mismatch between two inputs, by providing the distance of A that is the farthest from any point of B [17]. For this work the Python library MedPy was used to calculate HD, the output of this function returns the distance in millimeters.

### 4.3.3 Model Training and Validating

**Training**

All of the employed models were built using the Keras Python library [5] and depending on availability they were trained on servers equipped with either an NVIDIA GeForce GTX 1080 or a TITAN V graphics card. This significantly aided in keeping training times acceptable.

All models were trained under a 10-fold cross validation procedure. They were trained for 25 epochs each time saving the best performing weights for the validation subset specific to each fold. They all used the Adam optimizer with a learning rate of 0.0001, chosen based on early experimentation and it’s promising results and they all used Binary Cross Entropy (BCE) as a loss function. This loss function was chosen as the task at hand can be considered a pixel-wise binary classification between two different categories: brain and not-brain.

The images used underwent a per slice pre-processing step focused on normalizing the input values. The process consisted in first carrying out a type of Winsorization technique in
which all the negative values were set to zero and the remaining values were limited so that the values greater than the 97% percentile were set to the value at the 97% percentile. This was done with the goal of censoring outlying values and to mitigate their possible influence. After this Winsorization step the values found in the slice were normalized to fit between 0 and 1. This normalization was carried out through a Min-max normalization, defined by:

$$ S(x) = \frac{x - \text{Min}}{\text{Max} - \text{Min}}. $$

(4.4)

where $S$ is an MRI slice, Min is the minimum value found within the slice, Max the maximum value and $x$ a single pixel value. This process was applied to every single $x$ ins $S$, in other words every pixel in every slice. Since all of the images had negative values and the previous Windsorization operation sets the minimum value to 0 the normalization formula can be rewritten as:

$$ S(x) = \frac{x}{\text{Max}}. $$

(4.5)

the images were then multiplied by 255 to be able to be represented as a visual representation of the image using certain Python libraries. However prior to being used by the CNNs they were again divided by 255, to keep their available value range between 0 and 1.

Model Validation

As briefly mentioned before, to validate the models K-Fold cross-validation ($K = 10$) was used. This is a method to evaluate the performance of a model and a good way to evaluate the model on limited data, an overview of it can be found in figure 4.1. During this method all the available data samples are randomly divided into $K$ groups, the model in question will then be trained $K$ times, each time using $K - 1$ of the $K$ samples as the training subset and the remaining sample will be held out and used as the validation subset, that will have the performance metrics performed on them. This is then repeated $K$ times until every one of the $K$ groups have been used exactly once as a validation subset [28]. After finishing all of the performance metrics are gathered and averaged into a final estimate used to understand the performance of the model.

![K-Fold Cross Validation](image)

Figure 4.1: An example of K-Fold cross validation ($K=3$), all the training samples are grouped into three groups. These groups are then left out one at a time while the remaining are used for training. Performance metrics are applied to the held out group. Taken from [28]

During this research work the K-Fold Cross Validation gets carried twice for each of the models. During the first run no post processing steps are applied, the 2D image slices are just
4.3. METHODOLOGY

stacked on top of each other to form the original 3D MRI shape. However, during the second run the resulting inference mask goes through a series of post processing steps which applies a 3D morphological closing operation, to fill holes within the final mask. Then the largest connected component is located and smaller disconnected components are discarded. This is done to attempt to remove unwanted artifacts.

The compare the models the statistical test Wilcoxon signed-rank test will be carried out. This test compares paired samples to find if there exist a significant difference between models and unlike the Students t-test does not assume that the samples come from a normal distribution.

4.3.4 Segmentation Models

This subsection concerns itself with examining the multiple CNNs architectures used for the task at hand, particularly Fully Convolutional Network (FCN), U-Net and Mask R-CNN. After each of the architectures is shown, an overview of the attention modules: Squeeze and Excitation, and attention gates is provided.

Fully Convolutional Networks (FCN)

Figure 4.2: Example of the transformation of an image prediction CNN into a Fully Convolutional Neural Network (FCN), taken from Long and Shelhamer (2015) [34]

FCNs were introduced by Long and Shelhamer (2015), the approach comes from the relatively recent success that CNNs had on image related tasks. This approach is built upon CNNs models used for whole image classification, where fully connected “dense” layers are used to arrive to a categorical prediction class. However, in this process the models completely disregard all spatial information, which is essential for image segmentation. FCNs replace these fully connected layers with convolutional layers, that allow for outputs that can be represented as classification maps of the same dimensions of that of the input images. To achieve an output with the same dimensions of the input a deconvolution operation is needed to expand the feature maps dimensions. Because typically classification CNNs reduce the the feature maps dimensions through both convolutional and pooling operations.

The transformation of a prediction model CNN into a FCN segmentation model can be seen in figure 4.2. This figure clearly illustrates that any image prediction CNN can be
transformed into a FCN. For example the authors of [34] make use of VGG, AlexNet and GoogLeNet networks. The architecture of the FCN model that was used during this research project can be seen in figure 4.3.

A particular drawback of FCNs for semantic segmentation is that the final output is usually coarse and lacks fine detail. This is because of the down sampling through the network and the use of a single deconvolution with a stride of 32 pixel at the final layer that explode the dimensions size. This large stride is used because for FCNs the information that gets upsampled comes only from the last layer and finer details are by then lost. To improve the resolution of the final mask information from previous hidden layers from the CNN can be extracted, upsampled and merged to the final result. As seen in figure 4.3, the results from the intermediate blocks 3 and 4 get upsampled making use of a much a smaller stride deconvolution, size 8 and 16. These intermediate output masks are then added to create a final finer prediction. The effect of the inclusion of intermediate layers information can greatly improve the final performance, this can be seen in figure 4.4.
4.3. METHODOLOGY

Figure 4.4: FCN prediction refinement through the fusion of multi layer information, taken from Long and Shelhamer (2015) [34]

Figure 4.5: Original U-Net architecture taken from Ronneberger et al. [41]. Blue boxes correspond to a set of feature maps. The number of feature maps is shown on top of the box while the dimensions are provided at the lower left edge. White boxes represent copied feature maps. Taken from Ronneberger et al. [41]
CHAPTER 4. WHOLE FETAL BRAIN SEGMENTATION

U-Net

This approach was first used in 2015 by Ronneberger et al. [41]. The main goal was to extend the architecture used in FCNs so that it could work with fewer images and also to improve the final precision of the segmentation maps. The resulting network architecture seen in figure 4.5, consists of a downward contracting path and an upward expansive path, which are also often called encoder an decoder paths. The decoder profits from information from the encoder via a skipped concatenation of feature maps. This connection allows the network to take into consideration finer spatial information and helps mitigate the vanishing gradient problem.

The U-Net network proposed in [41], consists of a downwards path that has groups of two convolution operations followed by two autonomous paths, a skip connection to a concatenation to its upwards equivalent part and a max pool operation that halves the input space and continues the downwards path. After the bottleneck layer the model has an upwards path made up of groups of two convolutions that take as input the result of an deconvolution operation from a previous block, this deconvolution step effectively doubles the input space, as to maintain compatible dimensions to that of the encoder and the initial input. As mentioned before, the result of the deconvolutional layers is concatenated with previous outputs from the downward stage regaining finer spatial information.

U-net has been used in a wide variety of medical segmentation tasks proving to be extremely useful for such tasks. Figure 4.6 shows the U-Net model that will be used for the whole fetal brain segmentation. The original architecture used in [41] represented in figure 4.5 is closely followed, however there are some differences worth noting:

- The input size and output size is of course different [41] has an output size of $388 \times 388 \times 2$

![U-Net Upsampling + Convolution](image_url)

Figure 4.6: U-Net architecture, arrows denote different operations
while the output size sought here is of $256 \times 256 \times 1$ the identical dimensions of the input image, this is achieved by using convolution operations of ”same” as the padding type.

- Ronneberger et al. in [41] mentions that they use dropout in the bottleneck portion but do not mention exactly where. There exists some discussion on the use of dropout in CNNs as the traditional dropout technique is equivalent to applying a salt and pepper filter. However, the authors argue that it works as a kind of data augmentation [41]. Therefore, it was decided to include two dropout layers right before entering and right before leaving the bottleneck area.

- For the up-sampling operation Ronneberger et al. in [41] make use of a transpose convolution with filter size $2 \times 2$, here the up sampling was executed using nearest neighbor interpolation followed by a convolution operation with filter $3 \times 3$.

Even though there were multiple changes made into the original architecture designed by Ronneberger et al. [41] the changes are minimal and do not affect the overall idea behind the architecture.

**Mask R-CNN**

![Mask R-CNN overview](image)

Figure 4.7: Mask R-CNN overview, taken from He et al. (2018) [14]

Mask R-CNN was proposed by He et al. (2018) [14]. This model is presented as an intuitive extension of Faster R-CNN [10], an object detection model. This extension adds a segmentation network that predicts a mask for each proposed region of interest (RoI). This segmentation network is made using a small FCN. A large overview of the workings of said method can be seen in figure 4.7.

A particular challenge that rises within this model is that Faster R-CNN does not provide a pixel to pixel alignment between the input and output. This is caused by the RoIPool layer they employ. To overpass this challenge, Mask R-CNN uses a RoIAlign layer which maintains the spatial information between the inputs and outputs aligned. The addition of this layer improved the mask accuracy by a relative 10% to 50% [14]. The RoIAlign operation can be seen in figure 4.8, where the dash grid represents a particular feature map, the solid $2 \times 2$ grid represents a RoI and the 4 dots inside each bin are the sampling points in each bin. RoIAlign computes the value of the sampling points by bilinear interpolation from the near grid points.
in the feature map. Previously both Fast R-CNN [40] and Faster R-CNN [10] employed a RoIPool based on max pooling in which they would simply choose the highest value in the grid for each sample point.

![Figure 4.8: RoIAlign overview, taken from He et al. (2018) [14]](image)

As mentioned Mask R-CNN is an extension of Faster R-CNN and therefore follows the same procedure that uses a Region Proposal Network and then makes use of parallel class prediction and bounding box networks. The additional mask branch within Mask R-CNN then makes use of a FCN to provide a mask for each RoI. This additional branch has an output dimension of size $Km^2$ where $K$ represents a feature map for each one of the classes or categories, and each of these maps encode an $m \times m$ mask, naturally belonging to each class.

### 4.3.5 Attention Modules

Semantic segmentation CNNs such as FCN or U-Net, work by performing pixel-wise classification across every pixel in an image, using combinations of multiple filters before arriving to a final prediction. In these models each hidden filter has the same weight as those within the same layer, and also when sharing information between layers, as is the case with the U-Net model. Where every pixel has the same importance and they are all shared, even when they do little or nothing at all to contribute to the final prediction. The following modules, Squeeze and Excitation, and attention gate modules serve as a way of focusing the networks attention to particular characteristics typically overlooked by CNN.

#### Squeeze and Excitation

The Squeeze and Excitation module focuses on the relationship between channels. It aims to allow networks to use each channels global information as a learnable feature that can emphasize specific feature channels while ignoring less important ones [16].

An overview of this module can be found on figure 4.9. The process consists of two steps. During the first “Squeezing” step, the module uses global average pooling to generate channel wise statistics. The following “Excitation” step uses the previous calculated information as an input for two fully connected layer with a ReLu activation, and a fully connected layer with a Sigmoid activation to compute a vector of size $N$, $N$ being the number of channels, this means that for each channel a weight is calculated. Finally those weights are used
4.3. METHODOLOGY

Figure 4.9: Overview of how the Squeeze and Excitation Module learns individual weights for each feature channel, taken from Hu et al. (2017)[16].

to carry out channel-wise multiplication, so that the original filter channels are scaled with respect to their calculated weight, effectively giving different weights to each individual feature channel [16]. This module was added to the already established U-Net model. The Squeeze and Excitation operation takes place right before the max pooling operation in the encoder’s path and right before the upsampling step in the decoder’s path, this model can be observed in figure 4.10.

![U-Net + Squeeze and Excitation](image)

**Figure 4.10: U-Net + Squeeze and Excitation module**

**Attention Gate**

Similar to the Squeeze and Segmentation module, the attention gate module can be easily added to CNN architecture, that employ skip connections. In this work the gating component is added to the U-Net model. It functions by restricting the information flow that is used to
concatenate intermediate results from one side to another. It uses the information from lower dimension layers in the upwards path to restrict the information shared. This ensures that the information shared via the skip connection to the upwards path focus solely on areas that are predetermined as areas of interest.

\[ g \quad W_g : 1 \times 1 \times 1 \]

\[ F_x \times H_x \times W_g \times D_g \]

\[ \text{ReLU}(\sigma_1) \]

\[ \psi : 1 \times 1 \times 1 \]

\[ \sigma_2 \]

\[ \text{Resampler} \]

\[ \alpha \]

\[ x' \]

\[ F_i \times H_i \times W_i \times D_i \]

\[ H_i \times W_i \times D_i \]

The Attention Gate module represented in figure 4.11 works using two set of feature maps, the first one serves as the gating component and it’s responsibility is to restrict the information flow from the second set of feature maps. The process consist on applying a channel-wise addition to the two sets of input maps, followed by a ReLu activation, a 2D convolution using filter size $1 \times 1$ and a Sigmoid activation. The result of this process is a feature map that specifically highlights the grid elements that are worthy of special attention. This resulting filter is then multiplied with the original set of feature maps that was going through the attention gate, with the aim of only letting through the gird elements that are shown to be of interest by the gate component [45]. Figure 4.12 illustrates how the attention gate is incorporated into the U-Net model. The addition of this module creates the Attention Gated U-Net.

\[ \text{U-Net model with attention gate module that limits the information flow between downwards and upwards paths} \]
4.3.6 Regularization

To avoid overfitting of the models certain regularization techniques were employed to both the U-Net and the Attention Gate U-Net. The techniques considered include: data augmentation as to double the available training set, channel-wise dropout right before the first max pooling operation using a dropout rate of 0.10 and l2 weight decay with a penalty of 0.0001 which was found to be an acceptable penalty value. The models which employ this are referred to as extra regularized or have a * in front of their name.

4.3.7 Loss Functions

The U-Net and the Attention Gate U-Net models, were further trained using four different loss functions as to compare the different learning abilities. The four loss functions employed were: binary cross entropy, dice loss, focal loss [31], and a hybrid loss function which combined binary cross entropy and dice loss to which we will refer to as hybrid loss. This function gives a weight of 0.5 to the BCE loss function, this value was established based on results achieved from early experimentation.

Binary Cross Entropy:

\[
BCE = -\frac{1}{N} \sum_{i=1}^{N} y_i \log (\hat{y}_i) + (1 - y_i) \log (1 - \hat{y}_i). 
\]  
(4.6)

Dice Loss:

\[
dice\_loss = 1 - \frac{2 \cdot A \cap B}{A + B}. 
\]  
(4.7)

Weighted Focal Loss (\(\alpha = 0.25, \gamma = 2\)):

\[
WFL = - \sum_{i=0}^{N} \alpha (1 - \hat{y}_i)^\gamma y_i \log (\hat{y}_i) + (1 - \alpha) \hat{y}_i^\gamma (1 - y_i) \log (1 - \hat{y}_i). 
\]  
(4.8)

Hybrid loss (combined weighted binary cross entropy and dice loss):

\[
Hybrid = 0.5 \cdot BCE + dice\_loss. 
\]  
(4.9)

4.4 Results

4.4.1 Basic Models

The results of the training progression for the basic models used can be seen in Figure 4.13, here the progression for both the DSC metric and BCE loss is shown. This figure clearly shows that the models are overfitting to the training data as the training dice and loss metrics evidently shows a superior performance to that of the validation subset metrics. The figure also shows the variance that was present during the training as the thick solid colored lines show the average of the progress made by all of the k iterations during the k-fold cross validation, there is clearly much more variance present on the validation subset metrics than there is on the training set.
CHAPTER 4. WHOLE FETAL BRAIN SEGMENTATION

Table 4.2 shows how the models performed and how they perform against the vanilla U-Net. Results shown in *italics* represent a score statistically worse than that of U-Net. It can be observed that vanilla U-Net not only had on average a better score both for DSC and Hausdorff Distance with scores of 0.93 and 7.59mm, but it also performed statistically better than all of the other models compared in this experiment, all of the p-values calculated with the Wilcoxon test resulted in $p \leq 0.05$ showing that a significantly statistical difference existed between the results of the models. If we ignore the test statistic and look at only the average of the metrics it can be noted that vgg19 FCN appears to perform in a similar manner to that of vanilla U-Net however the p-value reached by vgg19 FCN on the DSC is $9E - 20$. This clearly shows that for most of the paired values U-Net outperformed vgg19 FCN. Looking at the average results it is also clear that the worse model is Mask R-CNN with a dice score of only 0.86 and a Hausdorff Distance of 26.87. The p-values obtained for this model compared against U-Net were of $9E - 41$ for DSC and $2E - 36$ for Hausdorff distance. The table 4.2 also shows the results after the post processing step where the vanilla U-Net achieved an average of 0.94 and a Hausdorff Distance of 3.15mm. All of the other models still performed statistically worse than the vanilla U-Net.

One of the problem of training Deep Learning models is the heavy computational cost it requires, therefore an important thing to consider is the training time. During this experiment the models were trained using an NVIDIA GeForce GTX 1080 GPU and the training time wildly differed. U-Net and vgg19 FCN each took around 8 hours to train the 10 models needed for the 10-fold cross validation while Mask R-CNN took more than 20 hours.
4.4. RESULTS

Table 4.2: Results for basic models. Shown in *italics* are the results that show a worse performance than the U-Net results and where there is a statistical difference calculated with the Wilcoxon test.

<table>
<thead>
<tr>
<th>No Post Processing</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
<td>Dice Similarity Coefficient</td>
</tr>
<tr>
<td>U-Net</td>
<td></td>
<td>0.93±0.09</td>
</tr>
<tr>
<td>vgg19 FCN</td>
<td></td>
<td>0.93±0.06</td>
</tr>
<tr>
<td>ResNet50 FCN</td>
<td></td>
<td>0.92±0.07</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td></td>
<td>0.86±0.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>With Post Processing</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Model</td>
<td>Dice Similarity Coefficient</td>
</tr>
<tr>
<td>U-Net</td>
<td></td>
<td>0.94±0.09</td>
</tr>
<tr>
<td>vgg19 FCN</td>
<td></td>
<td>0.94±0.06</td>
</tr>
<tr>
<td>ResNet50 FCN</td>
<td></td>
<td>0.93±0.08</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td></td>
<td>0.88±0.11</td>
</tr>
</tbody>
</table>

4.4.2 Attention Modules

Figure 4.14 shows the training progress of the vanilla U-Net model alongside U-Net models modified as to employ the two attention modules previously discussed, Squeeze and Excitation and attention gate, the models trained are shown in figures 4.10 and 4.12. Looking at figure 4.14 it can be seen that there is still an overfitting to the training data. However, the Squeeze and Excitation U-Net appears to reduce the variance present in the DSC computed in the validation subset during the training progress of each of the K iterations.

The results for the 10-Fold cross validation can be seen in table 4.3, here it can be noted that all of the models achieved the a DSC of over 0.93. However, the Wilcoxon test showed that the U-Net + Squeeze and Excitation performed statistically worse than U-Net on the DSC. The statistical test also showed that there is no significant difference between the vanilla U-Net model and the Attention Gated U-Net. Regarding the HD the average of the resulting distances shows that the Attention Gated U-Net reached a better metric score of 6.60mm. However, there exists no statistical significant difference between any of the results reached with attention modules augmented models and the results achieved using the vanilla U-Net model.

4.4.3 Regularization and Loss Functions Results

As the previous experiment results showed an undeniable overfitting problem extra regularization steps were required. The steps that were taken were: channel wise dropout with rate of 0.1 after the first convolution block, L2 weight decay, extensive data augmentation. Besides these regularization steps different loss functions were also tested, to find out if they could give an advantage on the task at hand.

Figure 4.15 shows the training progression for four vanilla U-Net models trained with
Figure 4.14: Training progression across 25 epochs for models that employ either of the two attention modules (SE or Attention Gate).

Table 4.3: Results for models using attention modules. Shown in *italics* are the results that show a worse performance than the U-Net results and where there is a statistical difference calculated with the Wilcoxon test.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dice Similarity Coefficient</th>
<th>Hausdorff Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No Post Processing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U-Net</td>
<td>0.93±0.09</td>
<td>7.59±20.70</td>
</tr>
<tr>
<td>U-Net + Squeeze and Excitation</td>
<td>0.93±0.10</td>
<td>7.40±19.63</td>
</tr>
<tr>
<td>Attention Gated U-Net</td>
<td>0.93±0.07</td>
<td>6.60±15.62</td>
</tr>
<tr>
<td><strong>With Post Processing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U-Net</td>
<td>0.94±0.09</td>
<td>3.15±11.83</td>
</tr>
<tr>
<td>U-Net + Squeeze and Excitation</td>
<td>0.93±0.10</td>
<td>3.22±11.83</td>
</tr>
<tr>
<td>Attention Gated U-Net</td>
<td>0.94±0.07</td>
<td>2.57±1.92</td>
</tr>
</tbody>
</table>
extra regularization and each using a different loss functions. It can be noted that the over-
fitting problem was effectively mitigated, as the validation metrics outperformed the testing
metrics that could not be overfitted as the training subset was being constantly modified by
data augmentation. It is also be observed that some loss functions also allowed for a better,
more stable, training progression. This can be particularly noted on the U-Net model trained
with the Hybrid Loss function defined in equation 4.9. The training for the model using the
Hybrid Loss was able to reach a validation DSC of over 0.90, the model that made use of
only the BCE Loss function was kept around 0.85. The figure also shows a strong presence
of variance on the first half of the Dice Loss trained model and that the focal loss showed the
worse training progression out of the four models in question.

Figure 4.16 shows the training and validation progression for a vanilla U-Net model
and for two Attention Gated U-Net models, trained with Dice and Hybrid Loss functions and
with extra regularization. It can also be seen that there is no overfitting and that there is an
increase in performance. Again it’s noted that the better, more stable progress, is present on
the Attention Gated U-Net that made employed the Hybrid Loss function.

Table 4.4 shows the results of comparing the models with extra regularization and at-
tention gate components. Here it’s noted that the *U-Net using BCE Loss (5.16mm), Focal
Loss (5.16mm), Hybrid Loss (5.93mm) and the Attention Gated U-Net with Hybrid Loss
(5.03mm) have a better Hausdorff distance average than that of a vanilla U-Net (7.59mm).
However, no statistically significant difference was noted. Wxcept for the case of *U-Net
Focal Loss, which performed statistically worse than the vanilla U-Net.

The extra *U-Net with a Hybrid Loss and the *Attention Gated U-Net Hybrid Loss
models were the only two models that show no significant difference to the vanilla U-Net
model. All of the other models show a statistically worse performance than the vanilla U-Net.

Figure 4.17 shows the final segmentation results of different input images for some of
the employed models, these results will be furthered examined in the following section.

## 4.5 Discussion

The training progressions graphs shown in figures 4.13 and 4.14 clearly shown a problem with
overfitting. Even though the resulting DSC score is shown to be acceptable, those models are
clearly overfitted and may have trouble segmenting unseen images that can arise with different
qualities, such as different intensities, angles and locations that are not present in the training
data.

The addition of only attention modules did not have an improving effect in the data,
figure 4.14, still shows an overfitting effect and table 4.3 do not show a statistical improvement
over the vanilla U-Net model. However, the addition of extra regularization was shown to have
a positive effect on the training progression, this can be seen in figures 4.15 and 4.16, where
the validation steps showed a better performance than that of previous models and even to that
of the training data. This was to be expected as the training subset was changing constantly
across all epochs due to the data augmentation method employed.

The use of different loss functions also proved to be an important consideration. The
vanilla U-Net only employed BCE as the semantic segmentation problem can be described as
a pixel wise binary classification challenge. Nevertheless, the use of Dice as a loss function
Figure 4.15: Training progression across 25 epochs for models that employ extra regularization techniques.
4.5. DISCUSSION

Figure 4.16: Training progression across 25 epochs for different Attention Gated U-Net models with different loss functions, these models also employ extra regularization techniques.
Table 4.4: Results for models using additional regularization techniques and different loss functions. Models with * before their name use data augmentation, channel-wise dropout and L2 weight decay. Shown in italics are the results that show a worse performance than the U-Net results and where there is a statistical difference calculated with the Wilcoxon test.

<table>
<thead>
<tr>
<th>Results for Models with Regularization</th>
<th>No Post Processing</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Dice Similarity Coefficient</td>
<td>Hausdorff Distance</td>
<td></td>
</tr>
<tr>
<td>U-Net</td>
<td>0.93±0.09</td>
<td>7.59±20.70</td>
<td></td>
</tr>
<tr>
<td>*U-Net BCE Loss</td>
<td>0.93±0.07</td>
<td>5.16±12.99</td>
<td></td>
</tr>
<tr>
<td>*U-Net Dice Loss</td>
<td>0.93±0.07</td>
<td>7.06±15.79</td>
<td></td>
</tr>
<tr>
<td>*U-Net Focal Loss</td>
<td>0.92±0.06</td>
<td>4.16±7.64</td>
<td></td>
</tr>
<tr>
<td>*U-Net Hybrid Loss</td>
<td>0.94±0.05</td>
<td>5.93±13.54</td>
<td></td>
</tr>
<tr>
<td>*Attention Gated U-Net Dice Loss</td>
<td>0.93±0.05</td>
<td>8.95±20.78</td>
<td></td>
</tr>
<tr>
<td>*Attention Gated U-Net Hybrid Loss</td>
<td>0.94±0.06</td>
<td>5.03±12.49</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>With Post Processing</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Dice Similarity Coefficient</td>
<td>Hausdorff Distance</td>
</tr>
<tr>
<td>U-Net</td>
<td>0.94±0.09</td>
<td>3.14±11.83</td>
</tr>
<tr>
<td>*U-Net BCE Loss</td>
<td>0.93±0.63</td>
<td>2.85±2.62</td>
</tr>
<tr>
<td>*U-Net Dice Loss</td>
<td>0.94±0.06</td>
<td>2.62±2.42</td>
</tr>
<tr>
<td>*U-Net Focal Loss</td>
<td>0.93±0.05</td>
<td>2.78±2.13</td>
</tr>
<tr>
<td>*U-Net Hybrid Loss</td>
<td>0.94±0.04</td>
<td>2.33±1.50</td>
</tr>
<tr>
<td>*Attention Gated U-Net Dice Loss</td>
<td>0.94±0.04</td>
<td>2.59±2.14</td>
</tr>
<tr>
<td>*Attention Gated U-Net Hybrid Loss</td>
<td>0.94±0.05</td>
<td>2.32±1.44</td>
</tr>
</tbody>
</table>
4.5. DISCUSSION

Figure 4.17: Segmentation results for multiple models, the first 4 rows show challenging images while the last 3 rows show normal cases.
and also the use of the proposed Hybrid loss proved to work just as well. The important difference being that the training of a model using a Dice loss tried to maximize one of the metrics that was being use as an evaluating metric.

By looking at figure 4.15 it can be seen that the Hybrid Loss had a more stable learning progression in all of the cross validation runs. On the other hand the use of Dice Loss showed more variance between cross validation runs, especially during the first 15 or so epochs. Still, it manages to stabilize itself after those first troublesome epochs. The use of Focal Loss, a weighted loss function, did not improve the models performance and it actually worsened it, as can be seen in table 4.4. However, it must be noted that the hyper parameters were not further explored and could potentially be modified as to improve the performance. The Hybrid Loss, which combined BCE and Dice Loss, was the only one that showed a reduction in variance across the cross validating runs. Similar results can be seen on the regularized Attention Gated U-Net models seen in figure 4.16.

Table 4.4 shows that none of the models were significantly better to the vanilla U-Net model. However, looking only at the average of the results and to the example results presented in figure 4.17 it is clear that the models that employed the regularization techniques performed in an acceptable compatible manner. It must also be noted that the *U-Net model that made use of the Hybrid Loss function and both *Attention Gated models (Dice Loss and Hybrid Loss) were the only models that did not show a significant difference using the Wilcoxon test for the DSC. The *Attention Gated U-Net with Dice Loss did show a worse performance in the HD distance metric. The large difference between the HD averages showed in table 4.4 and the lack of significant difference using the Wilcoxon test, can be an indicator that large outlier values exists in the results, these outliers can influence the average but may not have a large effect on the rank signed test.

The effect that the regularization techniques had on the models can be seen in the training progression charts earlier mentioned. But more interestingly, the effect that the attention gate has on a U-Net model can be visualized in figure 4.19. This figure shows how the attention gate limits the information shared between the encoder and decoder paths, as it was illustrated in the model diagram showed in figure 4.12. The obvious alternative to this is not using attention gate, which can be seen in figure 4.18. This vanilla method shares all of the information from the encoder to the decoder and where it is concatenated with the information of previous steps. So the question remains if the use of attention gate module poses an advantage over training. It was stated that there was no significant difference between attention augmented models and the U-Net model. However, looking at the example results of unseen images in figure 4.17. It is clear that while all of the models are capable of performing an acceptable segmentation on the last three images the Attention Gated U-Net is much more capable of segmenting difficult images. Such as with the case of the second image.

Furthermore, if we focus on table 4.5, which gives an overview of the number of parameters it can be seen that the vgg19 FCN has the most number of trainable parameters with more than 20 million trainable parameters and that vanilla U-Net has less than half of them at around 7.7 million, while the regularized Attention Gated U-Net has 8.1 million parameters, this is of course more than the vanilla U-Net but the difference is insignificant. Therefore for the task of whole fetal brain segmentation the inclusion of the attention gate can be seen as a little extra work for an compatible performance in the case of easy images and a better performance on challenging images.
Figure 4.18: Representation of what the vanilla U-Net model sees at the end of each convolution block.
Figure 4.19: Representation of what the Attention Gated U-Net model sees at the end of each convolution block and the effect that the attention gate has on the information transferred from the encoder to the decoder so that only noteworthy spatial information is shared.
Table 4.5: Number of trainable parameters in different segmentation models

<table>
<thead>
<tr>
<th>Model</th>
<th>Trainable Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Net</td>
<td>7,759,521</td>
</tr>
<tr>
<td>vgg19 FCN</td>
<td>20,024,567</td>
</tr>
<tr>
<td>U-Net Se</td>
<td>7,814,049</td>
</tr>
<tr>
<td>Attention Gated U-Net</td>
<td>8,108,645</td>
</tr>
<tr>
<td>*U-Net</td>
<td>7,759,521</td>
</tr>
<tr>
<td>*Attention Gated U-Net</td>
<td>8,108,645</td>
</tr>
</tbody>
</table>

4.6 Conclusion

During this work different segmentation architectures were employed to carry out a fully automatic segmentation of the whole fetal brain. It was found that the base model vanilla U-Net trained with a BCE loss function works pretty well for this task, it does not require an exaggerated training time such as Mask R-CNN and it performs much better than not only Mask R-CNN but also to FCN-8s with a vgg19 back bone. The experimentation with attention modules did not significantly improve the results obtained by the vanilla U-Net. However, the inclusion of different loss functions, most importantly the Hybrid loss function proposed in equation 4.9 allowed for a more stable learning process and that more rigorous regularization in the way of data augmentation, weight decay and channel wise dropout, mitigated the problem of overfitting caused by a limited data set size. It was found that thanks to the inclusion of those techniques the Attention Gated U-Net and the U-Net both trained with the Hybrid Loss function were the only two models that performed in a comparable manner to that of the vanilla U-Net. The inclusion of the attention gate is particularly of interest as it almost adds no extra work to the training. It can be easily added even if it does not significantly improve the performance of the model. For this problem the addition of the Attention Gate allowed the models perform a better segmentation for challenging images. This improvement had of course an almost insignificant cost in computational operations.

4.7 Summary

This chapter dealt with the problem of whole fetal brain segmentation. It began with an introduction to the problem which outlined the need of automating the segmentation task. The introduction was followed by related work surrounding this problem. This was followed by a section outlining the proposed methodology. This section included: information data acquisition, an explanation on the two performance metrics employed (dice score and Hausdorff distance), training and validating procedures and an overview of different deep learning architectures used for segmentation tasks. The results of the experiments carried out were then presented and discussed. The main findings were that the no models performed statistically better than the vanilla U-Net but that heavy regularization on the models aided in their performance and that the proposed hybrid loss function and the attention gate module both improved training. The chapter ends with a brief conclusion on the work that was carried out.
Chapter 5

Discussion

The last two chapters focused on the two research problems that were outlined for this thesis work, AD classification and whole fetal brain segmentation. They presented the problem definition, the methodology followed, the results obtained and gave a discussion on the results and their implications. This chapter focuses on a deeper discussion that will focus on presenting other DL based medical image analysis applications and on discussing the current challenges and limitations that Deep Learning faces in the area of medical image analysis.

5.1 Other Deep Learning Applications

During this work some Deep Learning applications specific for medical image have been mentioned, such as classification, segmentation, prediction and detection. Classification and segmentation being the two most discussed. However, there exist multiple other applications that have not been discussed and are worth noting. This subsection focuses on providing a brief overview on different Deep Learning applications that are being researched.

Image Annotation, Shin et al. (2016a) [49] in their work "Learning to Read Chest X-Rays: Recurrent Neural Cascade Model for Automated Image Annotation" employ a Deep Learning model created by combining CNN with Recurrent Neural Network (RNN) with the goal of detecting a disease from an image while also annotating context such as location, severity and affected organs. The way they achieve this is by using a CNN to first detect certain context and using the output to subsequently train a RNN model capable of providing the image context in a human readable way. An overview of the methodology they followed can be seen in figure 5.1

Feature Extraction using Autoencoders, Sharma et al. (2016) [48] propose a method to effectively compare images using stacked autoencoders, a special type of DL networks that find a compressed encoding of its input value. The goal behind this work is to use the stacked autoencoder as a feature extractor and to use it’s compressed encoding as a feature vector that can then be used to effectively compare images to find similarities between images. Figure 5.2 shows how the autoencoders are used as a feature extraction tool.

GANs for Data Augmentation, Frid-Adar et al. (2018) [9] use Generative Adversarial Network (GAN) to generate synthetic medical images to enlarge small datasets with the goal of improving classification performance. Generative Adversarial Networks (GANs) work by
CHAPTER 5. DISCUSSION

Figure 5.1: Overall workflow for the automated image annotation, taken from [49].

Figure 5.2: Stacked autoencoders being used as a feature extractor [48].
5.2. CURRENT CHALLENGES AND LIMITATIONS

Neural architecture search. Deep Learning is extremely powerful because the models are capable of representing complex functions that can achieve a specific goal without being explicitly programmed by a human, Weng et al. (2019) [59] take this a step further by using a Deep Learning method dubbed Network Architecture Search (NAS), this method uses a topology of nodes, a search strategy and performance estimation to find the best Deep Learning architecture, this way the architecture must not be explicitly designed. The authors followed a U-Net architecture but they built it using two cell architectures, DownSC and UpSC, these architecture of these cells was defined using NAS. A diagram of the employed model can be seen in figure 5.5. An important finding was that the typical skip connection that concatenated the information from the encoder to the decoder was replaced by a Squeeze and Excitation operation, this is similar to the Attention Gated U-Net model that was proposed in chapter 4.

5.2 Current Challenges and Limitations

Throughout this work Deep Learning is regarded as a huge benefactor on the analysis of medical image, and it really is. Deep Learning has greatly aided in solving hard time consuming tasks such as image segmentation and it’s also shown to be able to infer complex diagnostics in medical areas such as dermatology, radiology and pathology [8]. However Deep Learning
CHAPTER 5. DISCUSSION

Figure 5.4: Total accuracy results for liver lesion classification using different data augmentation techniques, taken from [9].

Figure 5.5: (a) U-Net architecture (b) NAS-Unet architecture, the rectangles represent the proposed cell architectures that are automatically searched along with the gray arrows that represent a transformation operation, taken from [59].
5.2. CURRENT CHALLENGES AND LIMITATIONS

is definitively neither a “golden hammer” nor a “silver bullet”. There exists many challenges and limitations that hinder the progress of further research, these are now presented along with the current accepted solutions that mitigate their negative effect.

**Limited Data**, the availability of a large dataset is of upmost importance to the training of Deep Learning models. However, in medical image analysis the acquisition of large labeled data sets is difficult. And even when there is unlabeled data available the process of annotating it is very time consuming. Cho et al. (2016) [4], found that with just over 200 images an acceptable accuracy could be reached on the problem of identifying six different anatomical classes. This result is obviously attached to the problem they tried to solve but it can serve as a good starting point when considering carrying out Deep Learning research and being faced with a reduced dataset.

The most employed method to increase the dataset size is data augmentation [15], which was employed during the previously presented whole fetal brain segmentation project. This technique applies a series of random transformations, such as: flipping, rotating, cropping and scaling, to the available data. It is of upmost importance that this technique is employed after splitting the data into training and validation subsets. This because the random transformations can have little to no effect on the image and if the split is performed after augmenting the data, the same image can be present in both training and validation datasets [58]. Another method that’s used to mitigate the challenge is the use of Transfer Learning [15]. This method aided in the problem of AD classification. Where it was concluded that even model weights trained on a different domain were capable of aiding in the training of models on a medical domain. This finding has also been proved in previous works [55, 50]. Finally a different strategy that was not tried during this work is patch-wise training. This strategy consists in taking various patches from an image disregarding possible overlaps and treating those patches as individual images [15].

**Class Imbalance** is an issue commonly present in medical image analysis, where the number of training images that do not present a condition, such as rare diseases or conditions, typically out weight the number of samples that do show the condition [20]. This problem is also present on segmentation problems where the area of interest only covers a small percentage of a whole image [15]. The use of weighted loss functions, such as Focal Loss that gives different weights to well classified examples can be of assistance in solving this particular problem of having an large class imbalance [31].

**Expensive Training** by far the biggest limitation during this present work was the lack of a Graphical Processing Unit (GPU) equipped machine that was always available. This hindered the process of quick experimentation and of time consuming long validation procedures. Hesamian et al. proposes max pooling and batch normalization as ways of reducing training times and achieving faster convergence. However, these methods are worthless without a GPU that can be used to train the models. If research in Deep Learning is considered, access to a device with a GPU is an absolute must.

**Ethical Considerations** Ker et al. (2017) [20] makes an important consideration that even though it could be seen as an exaggeration must be considered. During their report they analyse work that have already outperformed radiologists and dermatologists. They now argue that ethical and legal aspects must be considered, as the use of DL as an aid in medical processes can lead to a patient being misdiagnosed or it could provoking further complications and even death.
5.3 Summary

This chapter presented other deep learning applications for medical image analysis that were not further explored during this work. Four different applications were showed: image labeling, feature extraction with autoencoders, data generation with GANs and architectural neural search. After this the current challenges and limitations that deep learning faces regarding medical image analysis were outlined alongside some possible solutions that mitigate these challenges.
5.3. SUMMARY

5.3.1 Future Work

AD Classification

The work presented in this thesis revolving around the classification of AD, served only as a proof of concept that the MRI images possessed discriminative features that could be learned and used to create a Deep Learning model capable of classifying the subjects into either the AD or NC groups, it achieved this only using a single MRI slice and knowledge transferred from a different problem domain in the form of trained weights.

Further work can and should be carried out. Particularly using 3D convolutional neural networks to take advantage of the spatial information it provides and not only designing the model as a binary classifier but as a predictive model that could determine weather a subject will progress into suffering from AD in a particular time frame. This would be extremely useful as there is a clear need to control the economic and social costs that arise from this disease [37]. Also the fusion of clinical features into a DL model can prove to be of aid such as it was reported by Altaf et al.(2018)[1] and Senanayake et al.(2018)[47], and it should be something worth considering. The work of [36] (2018)[36] has found the most relevant clinical features to detect AD, it should be considered the fusion of these features to a DL model as to improve its predictive abilities.

There exists a lot of research surrounding the prediction of this disease, some showing very promising result. However, work by Wen et al.(2019)[58] has shown that a lot of the methodologies followed in recent research commit some sort of mistake that contaminates their validation dataset such as augmenting data before splitting into training and validation data subsets or not splitting their dataset at a subject level which could lead to having multiple images for the same subject end up on both training and validation sets. It is then of upmost importance that future work on this area should take all of this into consideration.

Whole Fetal Brain Segmentation

During this work extremely good results were shown, reaching DSC of 94% and small HD values, less than 3mm in difference. It must be considered that the training and the results validation was carried out using data available only to FNNDSC and that the manually segmented masks that were used as a desired outcome were done by only one person. This of course leads to the data being biased towards that particular masker. The resulting score of the inter-rater reliability was of a DSC of 89% and a Hausdorff Distance of 3.89, which is beat by the proposed models. Still the proposed models are not capable of fully automating the segmentation, a human still has to make correction for the mistakes made. This of course is the same problem that was found when trying to use the weights provided by Sadegh et al. [44]. However, it most be mentioned that the proposed *Attention Gated U-Net method is currently being used by student researchers at FNNDSC, they have reported that the Deep Learning tool greatly accelerates their work, but that it is not 100% perfect. Therefore further work is definitively needed for this task.

A limiting factor for possible advances regarding this task is the lack of publicly available data. Researchers all used different datasets and since each mask used is different depending on particular needs comparison between results from different sources is not reliable. There is a need of having a publicly available dataset that can serve as a benchmark of
model performance, or if this is impossible for legal reasons, at least the goal masks should be masked by multiple experts as to avoid biased results to one particular masker.

Regarding different Deep Learning methods the work proposed by Weng et al. [59] which uses Network Architecture Searches (NASs) to generate architectures, can be implemented to try and find an optimal segmentation architecture, also the use of GANs [12] can be of use to synthetically augment the available datasets.

Regarding the training procedure, it could be beneficial to train 3 different expert models each one focusing on only learning about a certain anatomical plane view, axial, sagittal and coronal. An initial input network could classify incoming data as to belonging to one of those views and sending the input image to the correct expert model, thus working in a hierarchical way to produce a final result. During this work data augmentation was used to effectively double the training set size, however time restrictions during training did not allow for a much more intensive procedure. A more exhaustive data augmentation procedure could be used to produce a much larger data set which could prove to be beneficial in the progress of fully automating the whole fetal brain segmentation process.
Chapter 6

Conclusion

This work focused on the use of Deep Learning techniques within the field of medical image analysis. First the problems were briefly introduced during the introductory chapter 1, then the shared theoretical framework was presented in chapter 2, where the most important aspects of CNNs where explained. After this, during chapters 3 and 4, two research problems were outlined together with the proposed solutions and their corresponding results and discussion. After that, there was a general discussion on the use of deep learning and on some of the main challenges it faces in a medical environment, specifically to when it’s related to image analysis such as when it uses MRI scans.

During the introduction the main objectives and established hypothesis were presented, it is worth remembering and drawing the conclusion on what was reached.

For the task of Alzheimer Disease Classification the main objectives were:

- Obtain Access, understand and recollect the data provided by ADNI
- Use a CNN to create a binary classifier for both classes.
- Use an established CNN architecture with fixed pre-trained weights as a feature extractor that is feed into a trainable head composed of multiple dense layers (Transfer Learning).
- Use an established CNN architecture with a variable number of trainable layers pre-trained weights as a feature extractor that is feed into a trainable network composed of multiple dense layers (fine tuning).
- Find the best number $n$ of trainable layers in the fine tuning model.
- Validate all of the models using a 10 times repeated 5-Fold cross validation.

And the established hypothesis was: “For the task of classifying a subject as either having or not having Alzheimer Disease a Convolutional Neural Network can be used, taking as an input only a single Axial MRI slice. Furthermore the use of pre-trained weights, trained in a different domain, is beneficial in the training process”.

All of these objectives were achieved, three different methods were presented all of them being able to outperform the random guess. It was also found that the use of pre trained
weights and its fine tuning can provide an advantage in the training and model validation reaching an accuracy of 80% using vgg16 with ImageNet pre trained weights and fine tuning the last 4 layers along with the training of two extra dense layers that take as an input the results of vgg16. These results are consistent with those found in the literature which specify that these kind of problems can benefit from a fine tuning approach. The results back up the established hypothesis. A single Axial MRI image is capable of classification between AD and NC categories. And the use of pre-trained weights trained on ImageNet images was proven to provide an advantage in the training process.

For the task of Whole Fetal Brain Segmentation the main objectives were:

- Manually create the goal masks of the available maternal MRI scans.
- Compute the intra-rating score of the created goal masks with those created by another masker.
- Train the vanilla U-Net model, FCN model and Mask R-CNN models
- Train the best resulting models using extra regularization techniques as to avoid overfitting. Using dropout, weight decay, different loss functions and data augmentation.
- Insert attention modules into the best working model. The modules employed are: filter attention based (Squeeze and Excitation) and grid specific (Attention Gate).
- Compare the results of the 10-fold cross validation using each of the employed models.

And the established hypothesis: “For the task of whole fetal brain segmentation the inclusion of attention modules to the U-Net segmentation model is able to statistically show a significant improvement over the vanilla U-Net model”.

During this work it was found that the U-Net model performed statistically better than vgg19 FCN, ResNet50 FCN and also Mask R-CNN, however heavy overfitting was noted which lead into the use of extra regularization such as feature map wise dropout, weight decay and extensive data augmentation. These measures had a positive effect on reducing the overfitting. The use of different loss functions was proposed, and they also proved to aid during the training of the network.

The addition of two attention modules was also proposed, in the way of U-Net + Squeeze and Excitation and Attention Gated U-Net. Out of all the employed segmentation models only the Attention Gated U-Net trained using the proposed Hybrid loss function showed no statistical difference to vanilla U-Net for the DSC and HD metrics. However, on the field results for unseen data show that the use of Attention Gated U-Net outperform vanilla U-Net for challenging MRI scans. It was then concluded that the extra computation complexity added by introducing Attention Gates could be disregarded and that they could provide an advantage for a small computational cost. The hypothesis must be rejected as there was no statistical improvement when using attention modules, but further work could be carried out to analyze the effect of Attention Gate on a set of challenging features.

Thanks to the research conducted during this thesis work it can be concluded that the use of Deep Learning can be used in a medical setting focused on the analysis of images, particularly on classification and segmentation tasks. However, certain challenges such as
having a large data set and the required hardware to carry out research must be considered before trying to solve a problem with Deep Learning methods.
Acronyms

**AD**  Alzheimer Disease. ix
**ADNI**  Alzheimer Disease Neuroimaging Initiative. 15
**AI**  Artificial Intelligence. 1
**ANN**  Artificial Neural Network. 1
**AUC**  Area under ROC Curve. 23
**BCE**  Binary Cross Entropy. 35
**BER**  Balanced Error Rate. 23
**CNN**  Convolutional Neural Network. 1
**DL**  Deep Learning. ix
**DSC**  Dice Similarity Coefficient. 35
**FCN**  Fully Convolutional Network. 31, 37
**FNNDSC**  Fetal Neonatal Neuroimaging and Developmental Science Center. 2, 31
**GAN**  Generative Adversarial Network. 59
**GD**  Gradient Descent. 8
**GPU**  Graphical Processing Unit. 63
**HD**  Hausdorff Distance. 35
**ML**  Machine Learning. 1
**MLP**  Multi Layer Perceptron. 9
**MRI**  Magnetic Resonance Imaging. 2
**NAS**  Network Architecture Search. 61, 66
NC  Normal Control. 2

pMCI  Progressive Mild Cognitive Impairment. 2

RNN  Recurrent Neural Network. 59

SGD  Stochastic Gradient Descent. 8

sMCI  Stable Mild Cognitive Impairment. 2
Bibliography


Curriculum Vitae

Master student born in Guadalajara, México, on October 26, 1993. He earned the Computer Science and Technology Engineering degree from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey Campus in Dec 2017. He was accepted in the graduate program in Computer Sciences in Dec 2017. He completed a research stay at the Fetal Neonatal Neuroimaging and Developmental Science Center (FNNDSC) at Boston Children’s Hospital while being affiliated to Harvard University.