TÉCNICO LISBOA

Alzheimer's Disease Diagnosis using a Variational Autoencoder

Luís Carlos Gonçalves de Castro

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. Maria Margarida Campos da Silveira

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Maria Margarida Campos da Silveira
Member of the Committee: Prof. Alexandre José Malheiro Bernardino

November 2019

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Was it worth doing? Everything's
worth doing
If the soul of the doer isn't small.

Fernando Pessoa, *Mensagem*

This work marks the conclusion of a five year journey, a path sometimes tempestuous but definitely enriching that made me grow both as person but also as a professional. Although the results were not as good as ideally I would want them to be, this work enriched me with a profound knowledge in Deep Learning topics, something that I value tremendously and will help me further in my future career. But as the word *teamwork* is almost synonymous with the word *engineering*, this crusade was supported by tons of people whose contribution cannot go unmentioned.

First, I have to express my gratitude towards my supervisor Prof. Margarida Silveira, who not only proposed my thesis but also supported me in every way possible, not only in a scientific point of view but also on a institutional one. Even when progress was not apparent, she never failed to motivate and help me through constructive feedback, resulting in ideas and solutions in which this work relies on. Also, I'd like to show my immense gratitude to my senior colleague João Pereira, for his tireless availability to discuss ideas and whose insight and experience regarding Variational Autoencoders was very handy to tackle problems I faced during this thesis.

Also, I have to thank my colleagues from IST, in particular Ricardo Martins and António Silva, who not only were *brothers in arms* throughout this entire course, but who also are people who I consider close friends and who are almost like a second family to me. Besides them, I have to thank my other friends whose support was also crucial throughout these years and who are more than I can mention.

Last but (definitely) not least, I have to thank my family. Their life example inspires me everyday and they are definitely big role models who I can't thank enough. Like a tree needs roots in order to survive, I would not accomplished half of what I have without their full and entire support. Even sometimes when things were though, they always gave me conditions to accomplish my goals and I will be forever in debt to them.

To you all, thank you.

Abstract

Alzheimer's Disease is one of the most common pathologies in aging humans, responsible for progressive brain atrophies which manifest themselves in cognitive and memory problems. Although a form of definitive diagnosis is yet to be found, it is possible to identify disease related lesions through inspection of brain scans, a task that can possibly be automated in order to aid specialists. In this work, a Deep Learning strategy will be adopted in order to build a framework that, given a scan belonging to a patient with an unknown disease status, produces an adequate diagnosis automatically.

Current approaches are all based on the concept of supervised learning through the usage of labeled datasets. This work on the other hand tries to interpret the automated diagnosis of Alzheimer's disease as a novelty detection problem with three classes (healthy, mild cognitive impairment and Alzheimer's dementia), relying on a modified Variational Autoencoder (VAE), an unsupervised and generative network equipped with convolutional layers responsible for automatic image feature extraction and encoding in the form of a chosen probability density function.

The proposed approach shows advantages, eliminating the need for labelled images and allowing one to inspect affected brain areas (which may be clinically relevant), while showing a possibility to detect anomalies through the output-input reconstruction error. However, it suffers from a highly uninformative latent space and an overdependence on hyperparameter values, which jeopardize its viability, especially when compared to a supervised deep classifier (with the same structure as the VAE's encoder).

Keywords

Alzheimer's Disease; Novelty Detection; Variational Autoencoder; Convolutional Neural Network; Variational Inference

Resumo

A doença de Alzheimer é uma patologia comum em pessoas idosas, sendo responsável por atrofias cerebrais progressivas que se manifestam em problemas cognitivos e de memória. Apesar de não existir uma forma de diagnóstico definitiva, é possível identificar lesões associadas à doença através de imagens cerebrais, uma tarefa que pode ser automatizada de forma a auxiliar especialistas. Neste trabalho, é proposta uma arquitetura baseada em Aprendizagem Profunda que, dada uma neuroimagem pertencente a um paciente cujo estado da doença é desconhecido, produz um diagnóstico adequado.

Actualmente, as propostas atuais baseiam-se no conceito de aprendizagem supervisionada através do uso de dados catalogados. Este trabalho, por outro lado, interpreta o problema de diagnóstico automático da doença como uma deteção de anomalias com três classes (saudável, défice cognitivo ligeiro e demência de Alzheimer), usando um autocodificador variacional, uma rede não-supervisionada e generativa, modificado com camadas convolucionais responsáveis pela extração automática de características da imagem e pela codificação das mesmas numa função densidade de probabilidade.

A arquitetura proposta mostra vantagens, eliminando a necessidade de dados catalogados necessários e permitindo a análise de áreas do cérebro afectadas (que podem ser medicamente relevantes) e a deteção de anomalias por análise do erro de reconstrução cometido entre entrada e saída da rede. Contudo, a solução apresenta um espaço latente pouco informativo e uma dependência grande dos valores dos hiperparâmetros, pondo em causa a sua viabilidade principalmente quando comparado com um classificador supervisionado (com estrutura igual ao codificador da rede não supervisionada).

Palavras Chave

Doença de Alzheimer; Detecção de Anomalias; Autocodificador Variacional; Redes Neurais Convolucionais; Inferência Variacional

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACC	Accuracy
AD	Alzheimer's Disease
Adam	Adaptive Momentum Estimator
ADNI	Alzheimer's Disease Neuroimaging Initiative
AE	Autoencoder
ANN	Artificial Neural Network
AUC	Area Under the Curve
CNN	Convolutional Neural Network
DBM	Deep Boltzmann Machine
DL	Deep Learning
DPN	Deep Polynomial Network
DTI	Diffusion Tensor Imaging
ELBO	Evidence Lower Bound
FC	Fully Connected
FN	False Negative
FP	False Positive
FDG	Fluorodeoxyglucose
fMRI	Functional Magnetic Ressonance Image
KL	Kullback-Leibler

MAPE Mean Absolute Percentage Error

MCI Mild Cognitive Impairment

ML Machine Learning

MMSE Mini Mental State Exam

MRI Magnetic Ressonance Image

MSE Mean Squared Error

NC Normal Control

PET Positron-Emission Topography

PDF Probability Density Function

pMCI Progressive Mild Cognitive Impairment

SAE Sparse Autoencoder

SGD Stochastic Gradient Descent

SSE Sum of Squared Errors

sMCI Stable Mild Cognitive Impairment

SVM Support Vector Machine

TCNN Transposed Convolutional Neural Network

TN True Negative

TP True Positive

t-SNE t-distributed Stochastic Neighbor Embedding

VAE Variational Autoencoder

1

Introduction

Contents

This is the first chapter of this document, which works as an introductory step in the scope of the presented thesis. First, a problem motivation is given, accompanied by a definition of Alzheimer's Disease and its main characteristics and phases. Afterwards, an overview to the topics of deep learning and neuroimaging are given. Finally, main objectives for this thesis proposed solution are set and an outline of the document is shown.

1.1 Problem Motivation

Alzheimer's disease(AD) is a progressive and irreversible brain disorder, more common among elderly people (age 65 and older) and one of the most frequent causes of dementia [2]. Patients suffering from this pathology usually experience different levels of cognitive and memory function impairment, starting with inability to remember newly learned information and eventually evolving to serious memory loss, lack of thinking skills or inability to carry out simple tasks. Although the speed of progression varies with the individual, the life expectancy after the diagnosis is from 3 to 9 years [2]. For example, Portugal places fourth in the list of OECD countries with the largest rate of AD dementia cases for every thousand habitants (roughly 205.000 people in total). When considering a bigger country as for example the United States of America, where it is estimated that this disease affects 1 in every 10 people with age 65 and older, the total amount of individuals counts up to 5.7 million, being more fatal than prostate and breast cancers combined [2]. It is estimated that Alzheimer's Disease will affect approximately 14 million people in the US alone by 2050, as shown in figure 1.1.

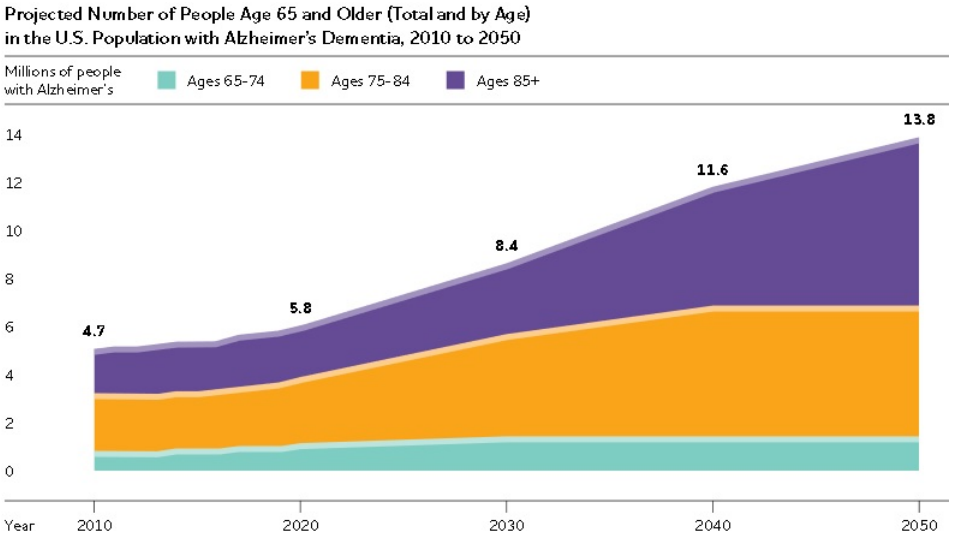

Figure 1.1: Projected progression of the total number of people with Alzheimer's disease in the United States of America. Image available in [2].

Currently, no cure or mechanism to slow down the progression of the disease has been found, despite

huge efforts being made in terms of research and investment in that sense. Therefore, early detection or forecasting of AD is crucial to not only treat symptoms in an early phase of the disease but also to aid those who are directly and indirectly affected by it to find ways to cope with the disease and to improve the quality of their future lives. However, AD's diagnosis is far from being something simple. First of all, and since the disease normally affects people with old age (as shown in figure 1.2), symptoms can sometimes be neglected and/or mistaken with effects of normal aging (e.g. memory complaints) which contribute to the patient not being aware of them or to others noticing them only in an advanced state of AD. Also, even after decades of research, an ideal diagnosis method is still yet to be found: usually, potential patients are first submitted to a cognitive test, and if the results are not normal, doctors have to rule out others causes (e.g. memory loss can also be caused by strokes or nutritional deficiencies) to then finally order brain scanning tests that combined with other medical data and family history can point out the possibility of AD.

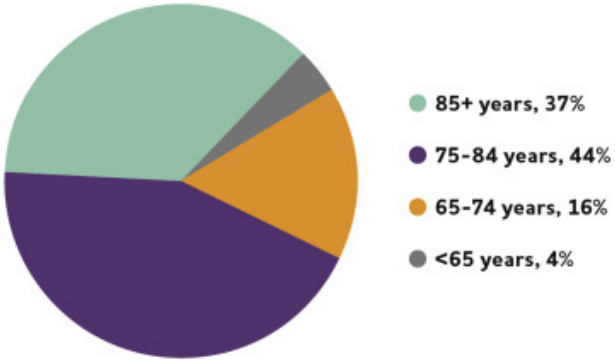

Figure 1.2: Age range of people affected by Alzheimer's Disease (in the United States of America). Image available in [2].

Although this is fairly accurate, the only way to actually confirm AD is by autopsy. Therefore, other mechanisms like computerized approaches based on neuroimaging are becoming increasingly popular, not only because of their overall good results in terms of diagnosis accuracy but also because they're able to automatically detect patterns of the disease when symptoms are not yet fully visible [3], thus facilitating medical diagnosis. In this work, it is assumed that this kind of problem can be posed as a typical machine learning classification problem: given an input (brain images from a subject), one wants to assign it to a specific class (in this case, if the patient is in any stage of Alzheimer's or not) based on a previous learning step. Besides this, ideally one would want to limit the amount of labelled images needed to train an automated tool, what would save resources (not requiring specialists to classify unlabelled brain scans, saving time and money). Of course, this problem is complex: due to the complexity of the inputs (3D brain images), the machine needs to learn how to represent the data in order to accurately classify it, leading to difficulties arising from the quality of the dataset, image feature selection, parameter tuning, learning process and so on, which will be discussed later.

1.2 Alzheimer's Disease: overview and stages

As mentioned previously in the subchapter 1.1, Alzheimer's Disease (AD) is a neurodegenerative disease associated not only with memory, thinking and behaviour problems but also with gradual loss of bodily functions. This symptoms are a direct cause of the brain's deterioration due to the destruction and death of nerve cells (such as neurons and synapses) [2]. The cause for this disease is mostly unknown, with genetic (mutations or heritage), health, environmental and lifestyle factors, among others being what researchers believe that can influence how AD manifests for the first time and eventually develops [2].

In a specific timestamp, one can distinguish three types of subjects:

- Healthy or Normal Control (NC), characterized by a state in which the brain is either healthy or the decline is normal due to the patient's age;
- Mild Cognitive Impairment (MCI), where the structural changes of the brain are bigger compared to what would be expected when taking into account a brain in the NC phase.
- Alzheimer's Disease (AD), a state described by an almost irreversible atrophy of the brain;

In figure 1.3, the three main stages listed above are exemplified with Positron Emission Topography (PET) images, a type of brain scan that assesses brain activity (as will later be explained in the subsection that concerns neuroimaging 1.3). Aside from the natural progression of the disease, it is also possible for the patient to regress to a previous state (e.g. from MCI to NC), or to progress directly from a normal state to AD, skipping the intermediate phase MCI.

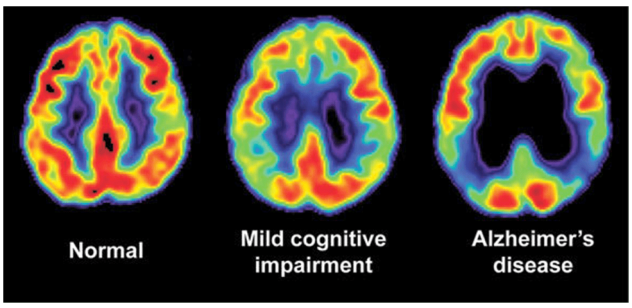

Figure 1.3: Positron Emission Topography (PET) scans of patients in the three different main stages of Alzheimer's dementia. Image taken from [7].

As previously explained in the problem motivation (subsection 1.1), the MCI phase gathers importance due to the fact that it is basically a state in which symptoms start to be noticeable, but are not yet as severe as they can be in a final period of the disease in case of conversion. Therefore, and taking into account the natural progression of the disease, the MCI phase can also be divided into two categories:

- Stable MCI, (sMCI or MCI-NC), a phase in which the patient does not experience sufficient structural changes in order to develop Alzheimer's dementia in a future timepoint;
- Progressive MCI (pMCI or MCI-C), a phase in which the disorder later evolves into AD;

Detecting these phases may aid the early treatment of symptoms, slowing down the development of the disease and thus improving patient's life quality. Besides this, monitoring MCI patients may be useful to better understand not only how symptoms evolve but also the different progression paths that the disease may follow.

1.3 Neuroimaging: definition and types

Neuroimaging is a group of techniques of three-dimensional brain *in vivo* images that have the purpose of studying the structural anatomy or activity of the brain. This category of medical image can be split in two separate groups:

- Structural images, which provide information about the brain's anatomical structure (e.g. Magnetic Ressonance Imaging (MRI) exemplified in figure 1.4 or Computerized Axial Tomography (CT));
- Functional images, representing brain activity at the instant of the scan, which is correlated with specific mental functions and faculties (e.g. Positron-Emission Tomography (PET) or Functional Magnetic Ressonance Imaging (fMRI)).

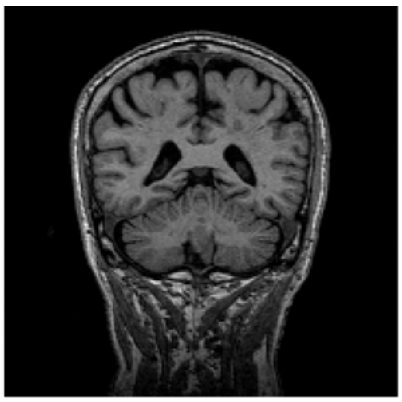

Figure 1.4: Example of a 2D slice of a three dimensional MRI. Image taken from the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset.

In the context of Alzheimer's disease, neuroimaging gathers great importance due to the fact that, as mentioned previously in 1.1, a method that infallibly diagnoses it has not yet been found. By using brain scans, one can spot brain disease patterns associated to the disease before symptoms are noticeable in the subject [3], enabling an early diagnosis.

Images like this, however, bare the burden of being highly complex due to the extensive variance of features that a human brain may present, which in turn increases the number of samples necessary for any type of disease modelling to occur. In order to tackle such difficulties, practical applications try to apply image preprocessing or feature reduction techniques [4] to these images in order to increase feature significance, thus decreasing the amount of samples needed for any form of computational tool to be used in the problem of automatized diagnosis.

1.4 Deep Learning: a brief introduction

Machine Learning (ML) [5] is a field of study that aims at solving specific tasks computer-wise, through pattern learning and mathematical inference. These tasks can vary in a wide range, from object classification to data clustering, in which machine learning algorithms mathematically represent models learned from examples within a sample dataset (also called *training* dataset) which are later evaluated using an independent and unknown dataset (known as the *testing* dataset). Depending on the task, its purpose and available resources, the learning process may be supervised (in which examples within the training dataset are labelled), unsupervised (examples in the training set don't have labels) or semi-supervised (in which only some examples belonging to the training dataset are labelled).

One subcategory of ML essential to this work is Deep Learning(DL) [6], whose goal is to learn complex data representations based on examples belonging to a training dataset through the usage of deep multi-layer networks. These are mostly used to sequentially decrease data complexity and increase meaningfulness of data representation in an hierarchical and successive fashion. DL algorithms like artificial neural networks (ANNs) try to replicate the learning process of a human brain, whose learning method is based on groups of neurons that are connected to each other and are subject to sequential electric impulses triggered by sensory parts of the human body. In the case of an ANN, connections between artificial neurons are mathematical and serve the purpose of trying to model information through input-output interactions.

In this specific scenario and as explained before in 1.1, neuroimages are highly complex, being characterized by a diverse range of features. Thus, it's logical to follow a DL-based approach that can mitigate these issues and not require extensive preprocessing, with the objective of detecting different stages of AD.

1.5 The Anomaly Detection Problem applied to AD diagnosis

As seen previously in section 1.2, an AD patient usually goes through multiple phases of the disease in different timestamps. MCI and AD are phases in which the brain presents bigger anatomical and

functional changes when compared to what would be expected from a healthy patient. However, one can approach this problem from a different point of view: by assuming that AD and MCI are anomalous classes that deviate from the normal case NC. Afterwards, the task relies in identifying them accordingly baring in mind that (as previously explained in 1.2) Alzheimer's Disease is a progressive disease in which MCI and AD are stages in which brain lesions are similar but display different degrees of severity, which allows to quantitatively measure how much they differ from the NC samples in order to accurately predict their class.

Anomaly detection [8] may be seen as a field of problems focused in detecting abnormal patterns in samples belonging to a certain dataset, i.e., finding examples (*outliers*) whose characteristic features dodge what would be expected for an example belonging to a certain class considered as normal (*inliers*). This goal may be achieved using two different strategies:

- if there are outliers in the training dataset and consequently the learned model focuses on fitting the regions in which the data is more frequent (assuming that the quantity of inliers is superior), known as outlier detection;
- if the training dataset does not have any outlier examples, the focus is now in the detection of new abnormal examples which are unknown for the model, a problem named novelty detection.

This class of problems has several applications that span through multiple areas, from bank fraud to cybersecurity threat identification. In the context of ML, anomaly detection can be applied in various forms, from clustering algorithms (e.g. Support Vector Machines (SVM)) amd linear regression problems (as depicted in figure 1.5) to deep neural networks.

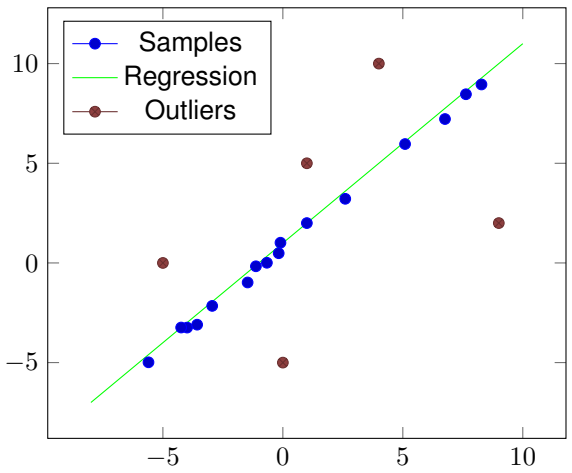

Figure 1.5: Example of anomaly detection in a regression task

As mentioned, AD and MCI are essentially anomalous states of NC. However, it is of this thesis interest to somehow not only detect the patterns of anomaly but also to model them, in this case,

to produce a classification for new unseen data. Since this type of problem deals with several types of anomalies, a novelty detection in which the framework is only exposed to healthy samples during training is ideal. The advantage of taking this approach is that it facilitates a possible strategy of only learning how to model one inlier class and, by consequence, fails to model other classes considered to be anomalous, which itself can be useful when dealing with high-dimensional and complex data such as neuroimages.

In practical terms, limiting the access of the network to labels brings advantages not only in terms of resources saved (expertise doctors who have to label the images manually, money being spent or even memory being saved) but also of serving as a guide tool which indicates problematic areas in the brain for doctors who use this tool as an aid to diagnosis (as will later be seen).

1.6 Solution Preview and Main Goals

This thesis is concerned with automatic Alzheimer's disease detection using a deep learning framework that uses neuroimages in order to accordingly detect different phases of the disease. As it will be presented in chapter 2, various types of solutions from a wide range of techniques applied to different neuroimaging modalities have already been proposed, achieving state of the art results. However, the vast majority of them follow a supervised approach, which in turn means that all datasets are labelled, something that in a real life application may have significant monetary costs.

The main goal of this thesis is to classify neuroimages by assigning three possible classes: NC, MCI and AD. In the core of this approach relies an unsupervised and generative network that follows a novelty detection approach by only utilizing healthy neuroimages during training, thus reducing dependence over labelled samples, something new when compared with existing solutions. In a classification phase, it is intended to analyze the several representation sub-spaces of the generative solution and by pre-defined metrics produce an adequate output. Later, it is intended to later split the MCI class into two subclasses (MCI-C and MCI-NC), which itself is a prediction problem of which patient will later develop AD. Since patients take part in several studies across a time period, the time window considered in the transition from non-converting to converting MCI may also change (e.g. a patient that has 3 equally spaced scans in which the first two are classified as MCI and the last one as AD can consider those same initial two images as MCI-C or the first as MCI-NC and the second MCI-C depending on the considered time frame) and thus the same problem may differ in terms of difficulty. To assess the quality of the solution by comparing unsupervised and supervised approaches, a supervised network using a similar structure will also be presented, with the same problems in hand.

1.7 Document Outline

This document is structured as follows:

- Chapter 2, where related works that aim to solve the problem of diagnosing several phases of Alzheimer's using Deep Learning are summarized;
- Chapter 3, in which a theoretical introduction to deep learning concepts that are used in the proposed solution (for instance, convolutional neural networks or autoencoders) is shown;
- Chapter 4, where proposed architectures for classification and data augmentation/anomaly detection are displayed;
- Chapter 5, in which different experiments using the proposed approach are shown and further discussed;
- Chapter 6, where the main conclusions and contributions are withdrawn and future areas of improvement are suggested;

2

State of the Art

Contents

In order to contextualize this work in the midst of existing proposed solutions, all contributions regarding procedures and algorithms towards the problem of Alzheimer's diagnosis using deep learning applied to image data will be briefly showcased in this section. Before that, an overview on the characteristics of different classification methods within the field of DL is shown, followed up with some introductory concepts on DL classification applied to AD's diagnosis that also characterize the presented solutions (for instance, type of feature selection or class discrimination).

2.1 The Task of Classification in Machine Learning

Before mentioning any related work regarding the task of AD labelling, it is important to first introduce the notion of a classification task in the areas of Machine Learning and Deep Learning, avoiding the introduction of any sort of classifying algorithm or neural network (examples of the latter which will be later presented in chapter 3).

The task of classification can be seen as a prediction operation that tries to accurately label given samples which belong to a dataset whose members may belong to only one of several classes. Labeling data can be seen as assembling a mapping function f, which given an input x produces a score y which indicates the most likely class to which x belongs to. Normally, due to the existence of labels in the learning process, classifiers naturally fall under the category of supervised learners, although there is also the possibility of approaching certain labeling problems in unsupervised fashion (for instance, using anomaly or novelty detection techniques). Classifiers may adopt several forms, p.e. decision trees to k-nearest neighbour algorithms. In the specific context of DL, a field in which this thesis is based on, classifiers take the form of deep Artificial Neural Networks (ANN). As later will be shown in the related work 2.3, all solutions presented propose different types of ANN's.

2.2 Alzheimer's Disease Classifier: a General Framework

As mentioned before in chapter 1.4, this thesis aims in identifying several stages of Alzheimer's through the usage of neuroimaging scans, and thus a deep learning approach might be ideal to tackle the natural difficulties that arise due to the complexity of this type of data. Consequently, it's plausible to compare this work to other solutions that follow this same approach in order to quantitatively and qualitatively evaluate it. Regardless of the specifics of any proposed work, a common framework of any deep learning solution for AD classification follows a four-step structure:

- Type of class discrimination, in which two or more of the several classes described before in section 1.2 are considered by the network (*AD*, *MCI*, *NC*, ...);

- **Image Data Modalities** which are used to train the overall architecture and offer information about the different considered disease stages (*MRI*, *PET*, *fMRI*);
- Feature selection, the way that the proposed solution extracts relevant information from the considered images in order to learn how to properly classify a certain sample (automatic if its done using a neural network that receives as an input the entire image and learns how to prioritize relevant diagnostic areas before classification using for instance voxel intensities, ROI-based approach if the inputs of the classifier are *a priori* known areas which are normally targeted by the disease, patch-based approach, in which images are divided into slices or collective groups and their features are either statistically chosen or automatically extracted and afterwards combined...);
- Main architecture, used to learn relevant representations of the considered disease phases for feature extraction, sample classification or both (Convolutional Neural Networks, Stochastic Deep Networks, ...).

Taking into account this four-stage concept, presented solutions in the related work in section 2.3 will be summarized relatively to this terms and all of its main contributions will be presented.

2.3 Related Work

In this section, all solutions that try to solve the problem of classification of the different phases of AD will be briefly summarized. The following subsections are divided considering the main problem that they focus on solving, whether it will be a binary or multiple-class classification problem, or the problem of forecasting the progression of the disease. In the final subsection 2.3.5, a short table is presented, summarizing the main aspects of the presented solutions.

2.3.1 AD/NC classification

This section is devoted to solutions that focus on solving the binary problem of classification between AD and NC classes. Bäckström et al. in [9] proposed a three-dimensional convolutional neural network (CNN) for automatic MRI feature extraction and classification, which also used data augmentation in the form of horizontal mirroring in order to increase the dataset's size. Hon et al. in [10] also uses a CNN based approach (with existing architectures), but applying transfer learning and using a group of *a priori chosen* two-dimensional MRI slices with largest entropy from three-dimensional scans: so, the feature extraction method is still automatic, but the network is only exposed to statistically significant parts of the image in order to increase feature significance. Zheng et al. in [11] applies a Deep Polynomial Network (DPN) (figure 2.1) which fuses MRI and PET information originated from regions of interest (named ROI, which are brain areas where the disease is known to have a considerable impact) and then applies

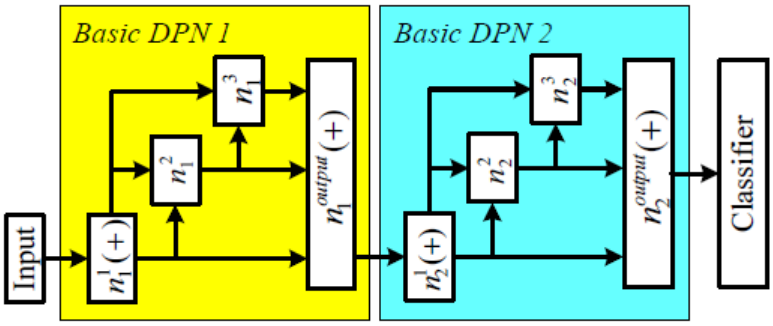

Figure 2.1: Example of a Deep Polynomial Network(DPN) based solution for AD/NC: schematic diagram of the architecture proposed in [11]

a final support vector machine (SVM) classifier. Sarraf et al. in [12] focus their work on the impact of extensive image preprocessing in fMRI and MRI's with the use of tools such as spatial normalization and smoothing, high pass filtering or motion correction (in fMRI) before applying a LeNET CNN classifier.

These solutions only focus on establishing a difference between two classes (NC and AD) and lack the ability to detect the prodromal phase MCI.

2.3.2 AD/MCI/NC classification

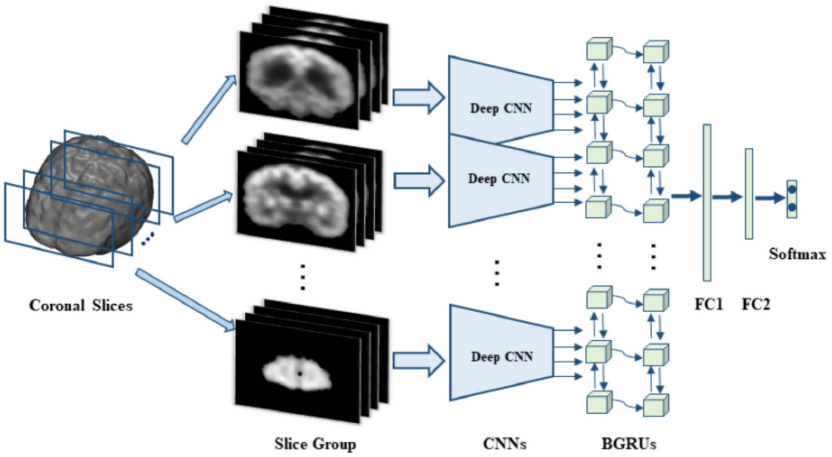

Figure 2.2: Example of a CNN and RNN based solution: schematic diagram of the architecture proposed in [19]

In this section, examples of solutions that also consider the MCI class in the problem of classification of different stages of the disease are presented. Vu et al. in [13] proposes a CNN which aims to solve two binary problems: NC/AD and NC/MCI, using a CNN who combines information from MRI and PET scans and whose input results from the output of two Sparse Autoencoders(SAEs) which are entitled to the task of automatically extracting relevant features for each modality. Aderghal et al. [14] applied a transfer-learning mechanism from MRI's to Diffusor Tension Imaging (DTI) using ROI's with a classifying

CNN. Liu et. al in [15] proposes a two-step framework that divides PET scans into two-dimensional slices according to three different directions, extracts and learns features through 2D CNN's and models the third dimension with the use of recurrent neural networks (RNN) (depicted in figure 2.2). Jabason et al. [16] used a similar approach to [13] but also included non-visual information about disease-related clinical scores in order to improve the results and solve the trinary problem NC/MCI/AD.

2.3.3 MCI-C/MCI-NC classification

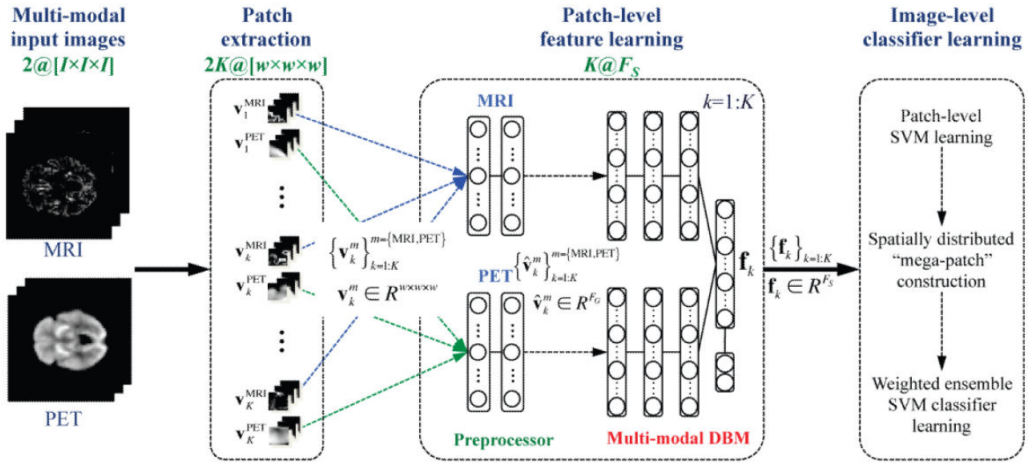

Figure 2.3: Example of a DBM-based framework: schematic diagram of the architecture proposed in [18]

As previously mentioned in section 1.2, the Mild Cognitive Impairment class can be split into two different groups: converters (to AD) or non-converters, and thus, can be considered as a binary classification problem. This classification problem can also be seen as a sort of predictive problem of a patient potentially being prone to develop Alzheimer's or not, according to the kind of MCI he possesses. Liu et al. in [17] propose a multi-task framework that is composed of a classification part (AD/MCI-C/MCI-NC/AD, with conversion window of 36 months) and a estimation part that estimates values of the clinical scores related to AD. The architecture proposed in [17] consists of multiple CNN's for each extracted brain patch (originating from identified statistically significative landmarks), whose information is later combined using a deep FC network. Suk et al. [18] propose a network (depicted in figure 2.3) that combines PET and MRI statistically significant information using a Deep Boltzmann Machine (DBM) and later classifying the resulting ouputs with a hierarchical SVM for 18 month conversions. Li et al. in [19] proposed a simple classifier pre-trained by a Restricted Boltzmann Machine (RBM), that learned features produced after applying Principal Component Analysis (PCA) and LASSO selection to MRI and PET ROI's, finishing it off with a SVM-based classifier, identifying conversions in an 18-month period. Choi et al. [20] apply a CNN which has been trained to solve the AD/NC binary problem using PET

images to MCI inputs in order to classify them as converting or non-converting (within a 3 year period). The MCI images are not used to train the CNN: instead, the networks learns specific features related to the AD and NC classes and based on that establishes how close the MCI sample is to each one of them in a test phase. This works by the assumption that a converting sample will more likely produce an AD label in the output of an AD/NC binary classifier, due to the bigger feature similarity existing between a MCI-C and AD brains relatively to MCI-C and NC brains, as in the same logic may be applied to the MCI-NC and NC classes.

2.3.4 Modeling the evolution of Alzheimer's Disease

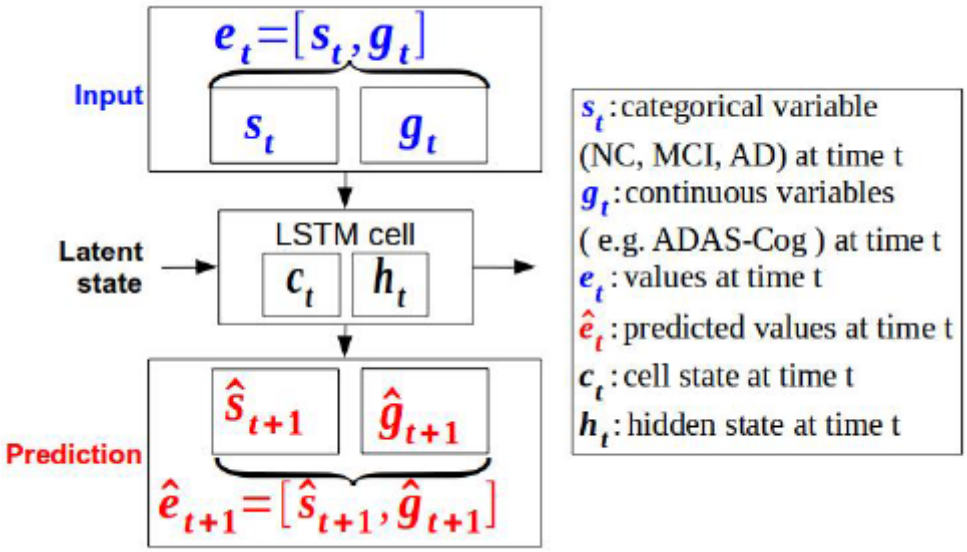

Figure 2.4: Example of an AD forecasting RNN solution: schematic diagram of the architecture proposed in [23]

The present section focuses in proposed networks that predict the progression of AD. Given the fact that AD is a progressive and degenerative disease, efforts to predict how it will evolve gather some attention within medical and scientific research. Lim et. al in [21] [22] proposed a Disease Atlas architecture that consisted of a shared temporal layer in the form of a LSTM-RNN that learns temporal correlations between variables (being images or scores), a so called "task-specific layer" that learns shared representations between the output trajectories from the temporal layer, and an output layer which computes parameters for a probabilistic distribution intended to model trajectories using generative methods like Monte Carlo Simulation. Applied to AD in [21], this solution works off ROI-features extracted from MRIs and PETs, combined with cognitive scores such as ADAS-Cog or CSF(cerebrospinal fluid), to produce conversion predictions in periods of 0.5, 1, 1.5 and 2 years. In another work, Nguyen et al. [23] proposed a deep-recurrent neural network (shown in figure 2.4) to model AD progression using ROI features like

ventricle and hippocampal volumes coming from different image modalities, also using a forward-filling technique to solve the problem of temporal inconsistencies in the provided dataset.

2.3.5 State of the Art Summary

In this section, a summary table of previously presented frameworks is given in table 2.1, comprising information such as the solution's main goal, data modalities, type of feature selection, choice of classifier and main results.

Article	Problem	# Subjects	Data Modality	Feature Selection	Classifier/Regressor	Main Results
[9]	AD/NC	199(NC)/141(AD)	MRI	CNN		ACC=90,1%
[10]	AD/NC	100(NC)/100(AD)	MRI	CNN		ACC=95,25%
[11]	AD/NC	52(NC)/51(AD)	MRI + PET	ROI	DPN +SVM	ACC=97,27%
[12]	AD/NC	52(AD)/92(NC) 211(AD)/91(NC)	fMRI MRI	CNN		(fMRI) ACC = 99.9% (MRI) ACC = 98.84%
[13]	AD/NC MCI/NC	172(NC)/–[1](MCI)/145(AD)	MRI + PET	SAE	CNN	ACC = 91,14% (NC/AD) ACC = 89,1% (MCI/NC)
[14]	AD/NC MCI/NC MCI/AD	228(NC)/399(MCI)/188(AD)	MRI + DTI	ROI	CNN	ACC = 92,5% (NC/AD) ACC = 85,0% (MCI/AD) ACC = 80% (MCI/NC)
[15]	AD/NC MCI/NC	93(NC)/146(MCI)/100(NC)	PET	CNN+RNN		ACC = 91,2% (NC/AD) ACC = 78,9% (MCI/NC)
[16]	AD/MCI/NC	653(NC)/1002(MCI)/238(AD)	MRI + Scores	SAE	CNN	ACC = 98,78%
[17]	AD/MCI-C/MCI-NC/NC	226(NC)/225(MCI-NC)/ 165(MCI-C)/181(AD)	MRI + Personal info.	Patch extraction from statistically significative image landmarks	CNN	ACC=60%(NC) ACC=51,3%(MCI-NC) ACC=24,3%(MCI-C) ACC=49%(AD)
[18]	AD/NC MCI/NC MCI-C/MCI-NC	93(AD)/204(MCI)/101(NC)	MRI + PET	Patch extraction from statistically significative image landmarks	DBM + SVM	ACC=95,35%(AD/NC) ACC=85,67%(MCI/NC) ACC=75,92%(MCI-C/NC)
[19]	AD/NC MCI/NC MCI/AD MCI-C/MCI-NC	51(NC)/56(MCI-NC)/ 43(MCI-C)/51(AD)	MRI+PET+Scores	ROI+PCA+LASSO + FCNN (initialized with RBM)	SVM	ACC=91,4%(AD/NC) ACC=77,4%(MCI/NC) ACC=57,4%(MCI-C/NC) ACC = 70,1%(MCI/AD)
[20]	MCI-C/MCI-NC	139(AD)/79(MCI-NC)/ 92(MCI-C)/182(NC)	PET	CNN		ACC=84,2%(MCI-C/MCI-NC)
[21]	AD's progression	1373(AD)	PET+MRI+CSF +Scores	ROI	Disease Atlas Network	AUC = 0.954 (0.5y) AUC = 0.935 (1y) AUC = 0.906 (1.5y) AUC = 0.899 (2y)
[23]	AD's progression	1600(AD)	PET+MRI+CSF	ROI	RNN	BCA = 79% [2.5pt]

Table 2.1: State of the Art Summary Table

[1] Not specified in the article.

3

Theoretical Background

Contents

This chapter's aim is to introduce the theoretical concepts in which this work relies on, starting from a brief introduction to the concept of an artificial neural network and examples of more simple/basic architectures and then progressing into more complicated frameworks that are actually used in the proposed solution (later shown in chapter 4).

3.1 Neural Networks: A Brief Introduction

A neural network [5] [6] is a type of machine learning model that can be seen as a weighted graph, formed by two or several layers of nodes (the first being called input layer, the final output layer, and those in between as hidden layers). This branch of models is also named Artificial Neural Networks (ANN) due to the concept similarity to how human brains function: every node (or neuron) may be connected to nodes of the previous or next layer in terms of weighted edges, thus allowing to establish a mathematical relationship between input and outputs of every layer within a neural network as

$$o^{(k)} = \begin{cases} x \text{ if } k = 0 \\ \phi_{(k)}(o^{(k-1)}) \text{ if } k \geq 1 \end{cases}, \tag{3.1}$$

where o^k represents the output vector of the nodes of the kth layer (which will be denoted as $l^{(k)}$), x the input vector of the network and $\phi_{(k)}$ as the function that given $o^{(k-1)}$ (input) produces $o^{(k)}$ (output).

3.2 Fully Connected Networks

3.2.1 Linear case

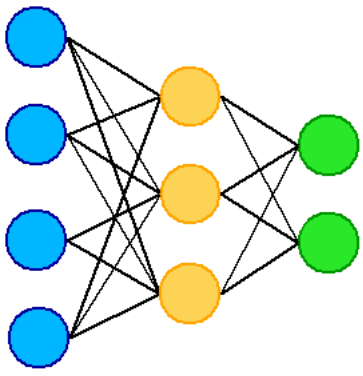

Figure 3.1: Example of a fully-connected network

Fully connected neural networks (FC neural networks) are, as the name suggests, constituted by consecutive layers whose nodes are all connected with each other, as depicted in figure 3.1. These

connections, as described in sub-section 3.1, can be defined in terms of weights. Formally, one can define two consecutive layers as $l^{(k-1)}$ and $l^{(k)}$ with n and m units respectively, thus resulting in a weight matrix $w^{(k)} \in \mathbf{R}^{m \times n}$. Weight matrices are how FC networks relate information held by different neurons in different layers in order to achieve the architecture's goal (for instance, a classification problem for a set of labelled data or parameter estimation for a *a priori* chosen model that tries to represent a group of values).

Additionally, fully connected networks may include a bias term in each unit, resulting in an array $b^{(k)} \in \mathbf{R}^{n \times 1}$ with n being the number of neurons in a layer l^k. Consequently, equation 3.1 may be written as

$$o^{(k)} = \begin{cases} x \text{ if } k = 0 \\ (o^{(k-1)})^T w^{(k)} + b^{(k)} \text{ if } k \geq 1 \end{cases}, \tag{3.2}$$

with $\phi_{(k)} = (o^{(k-1)})^T w^{(k)} + b^{(k)}$ being an affine transformation that produces $o^{(k)}$ from $o^{(k-1)}$.

3.2.2 Non-linear case by means of Activation Functions

Depending on the application, one can choose to change the complexity of a FC network by either stacking several layers of neurons or adding/reducing the amount of neurons per layer. In this context, adding units implies growing the amount of connections in between layers, which results in a better approximation to a certain dataset in exchange of adding more parameters to one's network (a consequence of a DL-based approach which was introduced in section 3.1). Nonetheless, regardless of the complexity of the network, sticking with the approach formalized by equation 3.2 will result always in a linear model, which may not be ideal in real-life problems.

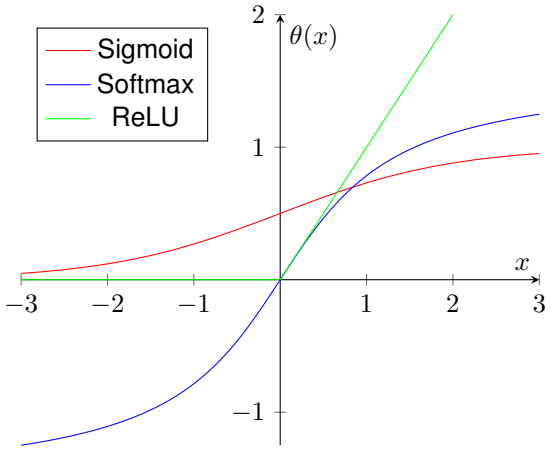

Figure 3.2: Example of three different activation functions

Thus, the concept of activation functions has high importance in the fields of ANNs and DL. An activation function is a non-linear function θ that is applied to the result of $\phi_{(k)}(o^{(k-1)})$, defining a neural

network as

$$o^{(k)} = \begin{cases} x \text{ if } k = 0 \\ \theta((o^{(k-1)})^T w^{(k)} + b^{(k)}) \text{ if } k \geq 1 \end{cases} . \tag{3.3}$$

There are several types of functions that can serve as a source of non-linearity, and are usually chosen depending on the context and application of the network itself [24]. Examples of widely used activation functions are shown in figure 3.2.

3.2.3 Network Training and Backpropagation

Although neural network training and optimization as a whole is an extensive research area that does not belong to the scope of this thesis, it has tremendous importance performance-wise in any ML application and will be shortly presented in this section and referred in later sections.

In order to achieve optimal parameters in any type of machine learning system that more accurately represent a sample dataset and, ideally, have a good generalization ability when tested with unknown datasets, one must determine a cost function (also denoted as loss function), that is written in terms of a predicted and real output error measure) which the network during a tuning phase has to minimize. In the case of FC networks, for a classifier (network which predicts the label of an input by assigning probabilities to each possible class), a common cost function $L(\theta)$ is

$$\mathcal{L}(\theta) = -(y \log(p(\theta)) + (1 - y) \log(1 - p(\theta))), \tag{3.4}$$

if there are two possible classes (binary classifier), in which y denotes the class value (0 or 1) and $p(\theta)$ the output probability for class 1 that results from the computations of the network with parameters θ. Equation 3.4 is usually denoted as the Binary Cross-Entropy Loss. For a multi-class classifier (more than two possible labels), $L(\theta)$ usually assumes the form of

$$\mathcal{L}(\theta) = -\sum_{i=1}^{M} y_i log(p_i(\theta)), \tag{3.5}$$

also known as the Categorical Cross-Entropy Loss, in which M is the number of possible classes y_i and $p_i(\theta)$ is the output probability for class i.

Solving analytically loss functions such as eqs. 3.4 and 3.5 is difficult and sometimes intractable. Instead, an iterative and recursive method known as the backpropagation algorithm [5] [6] [25] is used. Backpropagation is a method that, using a gradient descent based approach, updates networks' weights in order to minimize a chosen loss function through computation of its gradient, given one example at a time (*online* mode), all at a time (*batch* mode) or using a small group (*mini-batch* mode). Several optimization techniques, such for example the classic stochastic gradient descent (SGD) or the more

recent Adadelta [26] or Adaptive Momentum Estimator (Adam) [27], offer ways to reach optimal loss minima by tuning hyperparameters that act on the rate of weight changes (also known as the learning rate). For the sake of brevity, mathematical formulations and algorithms are available in the optimization algorithms' original papers, available in the bibliographic section of this document.

3.3 Convolutional Neural Networks

3.3.1 Background and Main Idea

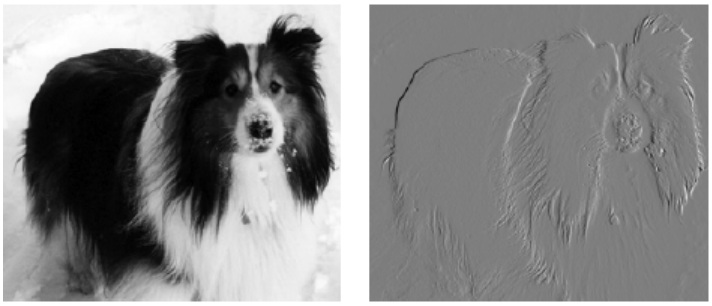

Figure 3.3: Edge Detection: a common operation in image processing. Image taken from [5]

When applying FC neural networks to large inputs, such as images or other types of high-dimensional data, one may face the problem known in the literature as *curse of dimensionality* [5]: as the feature space grows in dimensions, the range of configurations of an input increases, thus requiring a larger and deeper network (number of parameters increases almost exponentially), which itself requires a larger quantity of training examples in order to achieve a satisfactory performance. Additionally, a larger network implies a big computational cost and a larger amount of memory being spent, which may not be ideal.

In order to tackle the dimensional problem of image-based data within deep learning, an architecture based on the concept of receptive fields (regions in the visual cortex in which neurons only respond to certain stimuli) [28] is widely used: a Convolutional Neural Network(CNN). CNNs [5] [6] are a class of deep neural networks which, as the name suggests, rely on the mathematical concept of convolution, a linear operator denoted formally as

$$s[n] = (f * g)[n], \tag{3.6}$$

in which f is the input, g is a convolution filter (also called kernel) usually smaller than f and s is the feature map. For instance, if $f \in \mathbf{R}^2$, e.g. a two-dimensional image, the feature map resulting from

equation 3.6 takes the form of

$$s(i,j) = (f * g)[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m,n)g(i-m, j-n), \tag{3.7}$$

which can be seen as an array in which every element corresponds to the dot product between the kernel's values and those belonging to a part of the image f of the same size. In order to capture information on the border of the image, padding is used in order to fill positions outside of the image with numbers (zeros with zero padding for example), so that the kernel's window can be centered in those values and thus capture more information. Padding can also be used to maintain the image's size, due to the fact that non-padded convolution results in feature maps which are in size smaller than the original input.

Convolution in the context of image processing can intuitively be seen as an operation in which the smaller kernel slides through the different dimensions of the image, and through dot product captures relevant information about it, similar to the stimuli operation done in the visual cortex of the human brain. Convolutions are widely used in various image processing applications such as edge detection (as exemplified in image 3.3), image sharpening and blurring, with the difference in this applications relying on the kernel values being used or the amount of kernels being applied to the input himself. In the context of machine learning, several CNN-based architectures have been achieving state of the art results in object recognition problems (e.g. AlexNet [29], VGG-16 [30]), and have been also applied to other problems such as video analysis [31] or natural language processing [32].

3.3.2 Basic Model

Convolutional Neural Networks are, as said, a class of deep neural network that employ convolutions to its input. By assembling and setting different kernels with different values, one can form a convolutional layer, in which those values are regarded as network parameters that can be learned during training, being its multi-dimensional size a hyperparameter. CNN's are, as exemplified in figure 3.4, assembled by stacking several convolutional layers and usually are finished off a fully-connected network.

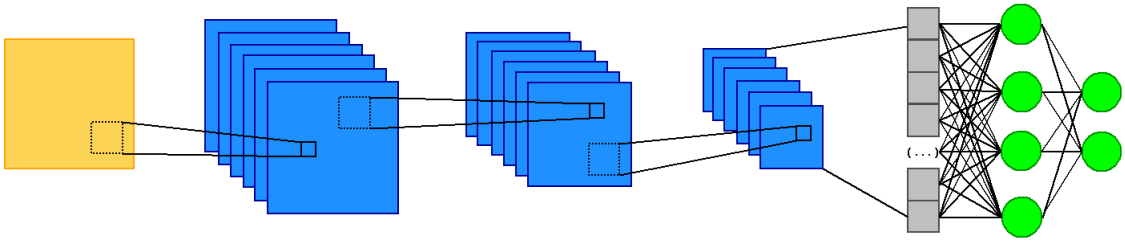

Figure 3.4: Basic Structure of a CNN. Input is represented as a yellow square, feature maps in blue, the resulting flattened feature map array in grey and finally the FC network units in green.

Conceptually, one can split a CNN into a feature extraction part, in which low-level features such as edges are extracted in the first layers and then progressively higher-level features are detected (e.g. shapes), and then a classification part, in which features are flattened into a 1D array, and then combined and modelled through the use of FC layers. The usage of convolutions let's one take advantage of properties which can simplify feature extraction and improve the overall quality of a machine learning system, such as:

- Sparse Connectivity, accomplished by using a kernel smaller than the input, which minimizes the amount of network parameters (when compared for instance with a pure FC network);
- Parameter Sharing, made possible by the fact that the kernel itself applies the same weight to every position in the input (or almost all of them, depending on how the operation is configured in terms of padding or strides, the latter which will later be explained later in section 3.3.3).
- Equivariance to Translation, achieved by the fact a convolution occurs when a kernel moves throughout the input always applying the same weights to different input positions, only focusing in detecting certain input properties and not specific ranges of values.

Much like FC networks, CNN's are also trained using backpropagation algorithms in order to adjust its parameters to more accurately model a sample dataset, in a similar way to what has been shown in sub-section 3.2.3.

3.3.3 Downsampling Operators: Pooling and Strides

In practice, convolution layers are not only composed of a convolution step, but also take into account a non-linearity step, in which the result of the convolutions are passed through a non-linear activation (similar to what happens in FC networks as explained in section 3.2.2), and a pooling step.

Pooling is no other than a downsampling method, used to decrease data size, increasing computational speed and keep data in a manageable amount for posterior classification. Given a post-activation feature map, pooling basically reduces data volumes into smaller values by simplifying information in a rectangular neighbourhood. There are many ways to perform pooling from taking maximum values (max pooling, exemplified in figure 3.5) to averaging values (average pooling) belonging to a rectangular neighbourhoods inside feature maps (the first is usually preferred over the alternatives due to performance reasons [33]). Pooling also produces invariance to small translations image-wise [5], due to the fact that it summarizes information coming from small neighbourhoods within the input. Outside pooling, there is another way to effectively reduce the size of the data: the usage of strides in the convolution operation, which define the amount of pixels its receptive field shifts in order to capture information. In the example of a 2D image convolution, formalized in equation 3.7, it is implicit that the convolution kernel

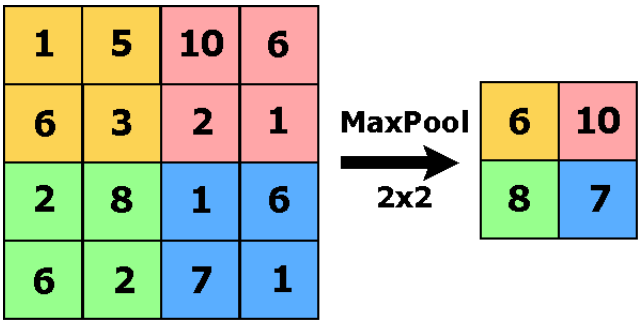

Figure 3.5: Example of a max pooling 2x2 operation

passes by every pixel (ignoring, for now, the pixels belonging to the border of the image): this operation has stride 1. On the other hand, stride n implies that a kernel captures information in a rectangular window centered in every *n-th* pixel of the image, obviously reducing the size of the feature maps. The usage of non-unit strides in convolution for downsampling has already been proven to match or even slightly outperform pooling, in exchange for a bigger number of trainable parameters [34].

3.3.4 Transposed Convolution Networks

As mentioned in the past subsections 3.3.1 and 3.3.2, the convolution operation produces feature maps which are defined in a lower dimensional space in comparison to the original dimension space in which the convolution input is defined. However, certain applications such as image denoising or interpolation require feature maps to be defined in bigger dimensions, thus giving room to the use of transposed convolution networks (TCNN) [35].

Convolving an input and a kernel can be expressed in terms of a matrix multiplication between a flattened version of the first and a convolution matrix $m \in \mathbf{R}^{k \times q}$, in which every row defines one single convolution operation (k is the number of convolutions between kernel and input and q is the length of the flattened array). Basically, a standard convolution maps a many-to-one relationship that is imposed between input and output feature map. When one transposes the convolution matrix, $m^T \in \mathbf{R}^{q \times k}$, and multiplies it with a flattened input, a larger one-dimensional feature map is obtained, corresponding now to a one-to-many relationship.

In a DL perspective, one can think of a CNN as a set of convolutional layers in which several filters are learned in order to map groups of receptive fields of an image to single values, while a TCNN has an opposite goal: instead, transposed convolutional layers are composed of several kernel parameters that are learned in order to accordingly associate single values to receptive windows.

3.4 Autoencoders

3.4.1 Standard Autoencoders

In this section, a specific kind of neural network is presented: autoencoders (AEs) [5] [6]. So far, all types of neural networks that have been presented in sections 3.2 and 3.3 try to learn how to classify inputs, and are trained using labeled data (in a supervised manner). However, autoencoders have a different goal: instead of being trained to label an input, they focus on learning encoded compressed representations of the latter in unsupervised fashion.

An AE is simply a neural network that is trained in order to copy an input to an output, by compressing the first into a lower dimension representation and then converting the latter back to its original form. Formally, defining the input as $x \in \mathbf{R}^n$ and the reconstruction in the output as $\hat{x} \in \mathbf{R}^n$, one can view an autoencoder as the junction of two parts. First, the encoder, which can be defined as:

$$z = f(x), \tag{3.8}$$

where $f(.)$ is the encoding function which takes the input x as an argument and produces an encoded representation $z \in \mathbf{R}^m,\ m \leq n$. In the other hand, the decoder is defined as:

$$\hat{x} = g(z) = g(f(x)), \tag{3.9}$$

where $g(.)$ is the decoding function with argument z and that produces a reconstruction $\hat{x}$ of the input. An AE is an unsupervised network which is trained in order to minimize a cost function $\mathcal{L}(x, \hat{x})$ that evaluates the degree of dissimilarity between input and output (L2-loss function, for instance). In other words, the goal of the autoencoder is to reduce the so called reconstruction error. An overview of the structure of a standard autoencoder is illustrated in figure 3.6.

However, autoencoders are supposed to not reconstruct a generic input x in a flawless way; instead, they are designed in a way that restricts the dimensions of the encoded representation z so that the network learns how to replicate only an approximation of the input. By lowering the dimensions of the encoded space representation of the input, the network learns how to prioritize certain features over less relevant ones, thus learning powerful underlying properties about the data in hand. Taking into account

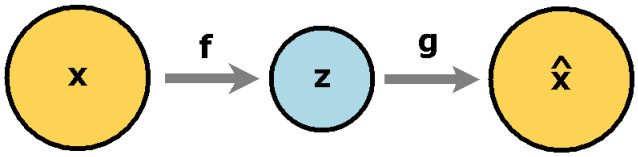

Figure 3.6: General structure of an autoencoder.

equation 3.1 defined in section 3.1, one can also define the encoder and decoder as a specific type of neural network with an input-output relationship, for instance a fully connected neural network or a CNN.

3.4.2 Variational Autoencoders

3.4.2.A Concepts of Stochasticity and Sparsity in Autoencoders

Standard autoencoders are fundamentally deterministic, as shown in subsection 3.4.1. However, recent autoencoder applications stretch the notion of encoder and autoencoder to a stochastic approach [5], by defining a probabilistic model $p_{model}(x, z)$ such that

$$p_{model}(z|x) = p_{encoder}(z|x) \tag{3.10}$$

is a stochastic encoder and

$$p_{model}(x|z) = p_{decoder}(x|z) \tag{3.11}$$

is a stochastic decoder. The notion of stochasticity applied to autoencoders allows for an extension of possible applications, such as the denoising autoencoder [36] and the generative variational autoencoder [37] (VAE), the main subject of this section which will later be properly explained.

Before introducing the VAE, another concept applied to autoencoders is important in order to fully understand it: sparsity. By introducing a penalty to original autoencoder loss function in the form of

$$\mathcal{L}_{sparse} = \mathcal{L}(x, \hat{x}) + \Omega(z), \tag{3.12}$$

in which $\mathcal{L}(x, \hat{x})$ is the original reconstruction error and $\Omega(z)$ is the penalty applied to the encoder output z, one forces the autoencoder to respond to unique and more significant statistical features of the training dataset, instead of trying to learn a compressed approximation of the identity function [5] [38]. Thus, the sparsity term $\Omega(z)$ acts as a regularizer term that forces the latent arrays z produced by the inputs x to be diverse depending on significant features. Depending on the problem in hand, Ω can take various forms, from a Laplace prior to the Kullback-Leibler (KL) divergence. Due to its selective nature, the class of Sparse Autoencoders (SAE), they are a popular solution concerning the problem of feature selection [39].

3.4.2.B Main Idea, Variational Inference and Expectation Maximization

Considering a set of parameters ϕ that model a set of functions $f_\phi(z)$ with argument z and given a certain dataset x, let one assume that the objective in hand is maximizing the probability of the output of $f_\phi(z)$ belonging to x, but that z is stochastic and is drawn according to some prior distribution $p(z)$.

To accomplish the set objective, one has to infer the values of z that are most likely to generate samples belonging to x [40] [41], which is equivalent to the computation of

$$p_\phi(z|x) = \frac{p_\phi(x|z)p(z)}{p(x)} \tag{3.13}$$

which is no other than a Bayesian Inference problem, in which $p_\phi(z|x)$ is the *posterior distribution*, $p(z)$ is the *prior*, $p_\phi(x|z)$ the likelihood distribution which is the result of the parametrization ϕ by means of f with argument z and $p(x)$ the probability of the dataset averaged over all possible parameters (the marginal likelihood). However, computing the denominator of equation 3.13 may poise a problem. The distribution $p(x)$ is calculated using

$$p(x) = \int_{-\infty}^{\infty} p_\phi(x|z)p(z)dz, \tag{3.14}$$

which is intractable, requiring the computation of $p_\phi(x|z)$ for every possible value of z. Thus, one possible solution is the usage of variational inference, which relies in the approximation of $p(z|x)$ by means of a family of distributions $q_\theta(z|x)$ parametrized by θ. Consequently, variational inference is formalized as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, D_{KL}(q_\theta(z|x)||p_\phi(z|x)) \tag{3.15}$$

in which

$$D_{KL}(q||p) = \int_{-\infty}^{\infty} q(u) \log \frac{q(u)}{p(u)} du \tag{3.16}$$

is the Kullback-Leibler Divergence, which is a positive real function that quantitatively measures the degree of difference between two distributions q and p. To solve the optimization problem 3.15, one cannot compute the KL divergence directly; however, it can be shown that the this objective can be fulfilled by maximizing

$$ELBO = E_{q_\theta(z|x)}[log p_\phi(x|z)] - D_{KL}(q_\theta(z|x)||p_\phi(z)), \tag{3.17}$$

known as the Evidence Lower Bound (ELBO), which is expressed in terms of the parameters ϕ and θ (see appendix A for mathematical proof). The original optimization problem is written as a function of an ideal parameter for the inference model θ. Since equation 3.17 is also written as a function of ϕ, this model parameter can also be optimized in order to maximize the Evidence Lower Bound, a technique called Variational Expectation Maximization.

3.4.2.C Basic Model

In a ML perspective, one can extend the notions and objectives presented in subsection 3.4.2.B to a deep learning context. The Variational Autoencoder (VAE) [37] is no other than a deep generative network, that works on the premise that any training set x can be generated from any underlying unobserved representation z according to some density function $p(z)$. VAE's basically model the parameters ϕ and θ as neural networks. Taking into account the ELBO expression 3.17 which one intends to maximize, the VAE is splitted into two main parts:

- Inference Model as an *encoder* with parameters θ, which tries to solve the variational inference problem of approximating $q_\theta(z|x)$ to $p_\phi(z|x)$;
- Generative Model as a *decoder* with parameters ϕ, who focuses the variational expectation maximization problem of maximizing the likelihood of x given z, or $p_\phi(x|z)$.

Relatively to what happens in other types of autoencoders, there's also one main difference: the output of the encoder network is not a deterministic latent vector z but instead is a set of parameters that model a target distribution $p(z)$. In fact, VAE's make no assumptions regarding the distribution of the training data x, but they do make assumptions regarding the distribution of the latent variables z, which is chosen *a priori* (e.g. multivariate Gaussian Distribution with zero mean and unit variance, $\mathcal{N}(0, I)$). Afterwards, a latent vector z is sampled according to the resulting modeled distribution, which is later decoded into a reconstruction of the input.

An example of a VAE model is depicted in figure 3.7, in which both encoder and decoder networks are simple fully-connected layers and $p(z)$ is assumed to be a multivariate Gaussian distribution (consequently the encoder outputs mean μ and variance Σ).

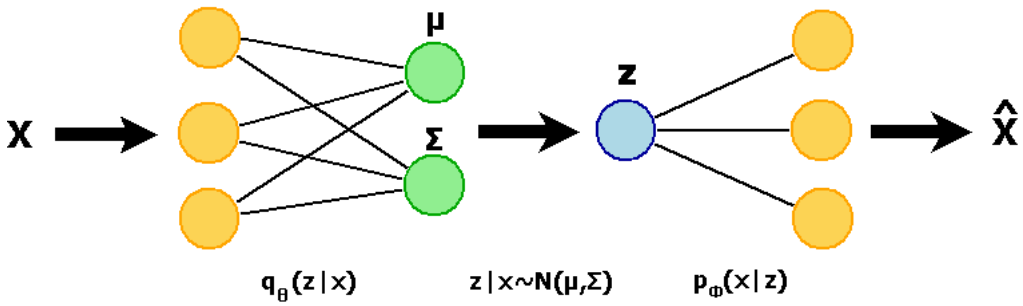

Figure 3.7: Example of a variational autoencoder with latent Gaussian Distribution and FC neural networks' as encoding and decoding functions.

As explained in subsection 3.4.2.B, the main goal is to maximize the Evidence Lower Bound (ELBO).

One can formalize a equivalent loss function for the VAE as:

$$\mathcal{L}_{VAE} = -E_{q_\theta(z|x)}[logp_\phi(x|z)] + D_{KL}(q_\theta(z|x)||p_\phi(z)), \tag{3.18}$$

which in a discrete case of a dataset x with N independent and identically distributed samples [40] assumes the form of

$$L_{VAE} = \sum_{i=1}^{N} \big[- E_{q_\phi(z|x_i)}[logp_\theta(x_i|z)] + D_{KL}(q_\phi(z|x_i)||p(z))\big]. \tag{3.19}$$

Interpreting the loss function 3.19 in a ML perspective, the LHS of the equation can be seen as a reconstruction error (expected negative log-likelihood), in which the output is encouraged to approximate the input in order to maximize the expected value of the likelihood of the data samples x given the latent code z; on the other hand, the RHS is a regularizer term, which forces the latent distribution q obtained by the encoder to approximate a target distribution $p(z)$.

3.4.2.D Network Training and Reparametrization Trick

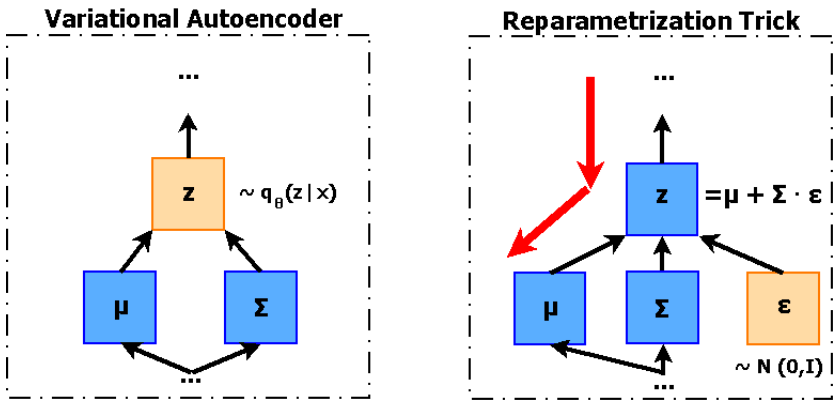

Figure 3.8: Reparametrization Trick applied to a VAE with latent Gaussian Distribution. The backpropagation path is marked with red arrows. Orange blocks are stochastic and blue are deterministic.

In order to minimize the loss function expressed in equation 3.19, by adjusting the network parameters, the VAE can be trained using a backpropagation-aided gradient descent method. However, as previously explained, the encoding part of the VAE produces a set of distribution parameters which the decoder then uses to sample z according to $q_\theta(z|x)$. Since standard backpropagation can't be applied to non-deterministic nodes, a technique known as the reparametrization trick [37] is used to allow weight adjustment in order to minimize the VAE's loss. Let one assume that the target distribution $p(z)$ is a

multivariate Gaussian Distribution. Thus,

$$z \sim N(\mu, \Sigma). \tag{3.20}$$

The reparametrization trick consists of expressing the latent variable z as a function expressed in terms of μ and Σ which are a direct and deterministic output of the encoding network, and a sampled parameter ϵ, whose distribution is fixed and independent of the model. Therefore, returning to the example of $p(z)$ modelled as a normal distribution, the latent variable z can be expressed as

$$z = \mu + \Sigma \times \epsilon, \ \epsilon \sim N(0, I), \tag{3.21}$$

which is a sum of the parametrized μ with the inferred variance Σ multiplied by a stochastic variable ϵ that poses as multivariate unit Gaussian noise. By applying the reparametrization trick, backpropagation stochastic gradient descent is made possible, opening a *deterministic path* in the network for the gradient to be calculated in successive fashion (due to the loss function now also being expressed in terms of the stochastic input ϵ) and weights correspondently updated, as shown in the diagram 3.8.

4

Proposed Solution

Contents

In this chapter, the deep learning framework used to fulfill the goal of multi-stage Alzheimer's Dementia detection is presented. There are two parts in this section: first, the classifier network is presented with the goal of accurately classifying different stages of AD in the form of binary problems; second, the generative network (in the form of a Variational Autoencoder) which is intended itself to serve as mechanism of novelty detection for classification of different stages of AD.

4.1 Classifier Network

4.1.1 Architecture

First off, the classifier network is presented with the goal of labeling inputs corresponding to subjects in different phases of the disease (NC/MCI/AD), with only two of them being considered at a time as shown later (binary problem). Since neuroimages are used as data, its logical to choose a Convolutional Neural Network (CNN) as the classifier to use. The chosen architecture is depicted in figure 4.1 and is composed of two levels:

- Feature extraction part, composed of N stacked convolutional and pooling layers for sequential and hierarchical automatic feature extraction and downsampling;
- Classification part, with M fully connected layers that establish a mathematical relationship between the extracted features and produce as an output a binary-valued label.

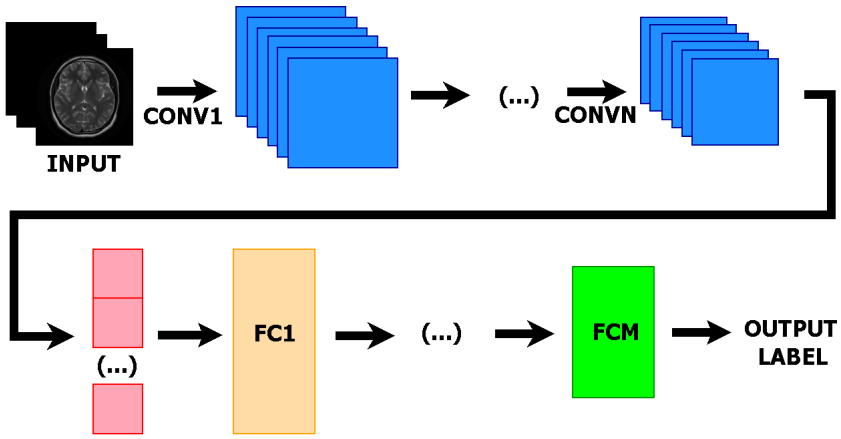

Figure 4.1: Architecture of the CNN used for classification purposes.

Consequently, this CNN has a two-step task ahead of it: first, through training, it has to learn how to, automatically, extract relevant features from the input neuroimages. Conceptually, this can be seen as a selective step in which the network sequentially learns how to prioritize disease relevant areas, that although may vary from patient to patient belonging to the same class, present a much greater

variance when compared to subjects from a different phase of the disease. Lastly, the CNN has to combine the extracted features, learning how to model them in order to produce a target label (in this case, the disease status of the patient). Through the use of activation functions, pooling and padding operators, the CNN assures a non-linear relationship between input and output with the goal of avoiding the learning of an oversimplified and biased model.

4.1.2 Considered Disease Phases

As explained before in section 1.2, a potential patient can be characterized by multiple labels which themselves correspond to different phases of the disease. In this work, the following classification problems will be considered:

- AD/NC, the binary problem that distinguishes healthy subjects from Alzheimer's Dementia patients;
- AD/MCI-NC, a binary problem in which the two anomalous states that indicate a bigger than expected mental health decline are set apart from each other (patients with MCI that eventually convert to AD are not considered);
- MCI-NC/MCI-C, which can be seen as a forecasting test, where the network tries to model this state specific patterns that imply that a patient with MCI will eventually progress into AD or not. This problem can itself be divided into multiple sub-problems depending on when it is assumed that a patient starts converting: whether it would be the simpler case in which a sample is considered to be MCI-C only at the time of the conversion to AD, or the hardest case in which before any conversion takes place, all MCI timestamps are considered as progressive. A mock example for a patient is shown in table 4.1, in which different conversion time windows lead to different labels being considered in the same image timestamp.

	Baseline	**Month 12**	**Month 24**
Labels	MCI	MCI	AD
Immediate Conversion	Non-Converting (MCI-NC)	Non-Converting (MCI-NC)	Converting (MCI-C)
12-Month Conversion	Non-Converting (MCI-NC)	Converting (MCI-C)	Converting (MCI-C)
24-Month Conversion	Converting (MCI-C)	Converting (MCI-C)	Converting (MCI-C)

Table 4.1: MCI-NC/MCI-C problems for a patient that fully develops Alzheimer's Dementia considering different conversion time windows.

4.1.3 Cost Function and Network Training

Depending on the problem at hand, the chosen loss function, as explained before in section 3.2.3, may adopt different forms. So, for the AD/NC and the MCI-NC/MCI-C problems, the loss function used to

train the CNN will be the Binary Cross-Entropy cost function:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \Big(y_i \log(p(\theta)) + (1 - y_i) \log(1 - p(\theta)) \Big), \tag{4.1}$$

in which θ is the matrix of weights of the network, $p(\theta)$ is the produced probability parametrized by the network parameters of the input data x belonging to a certain class y. Regardless of the chosen cost function, the network is trained using a backpropagation aided mini-batch stochastic gradient descent (SGD) for network training, as presented in algorithm 4.1.

Algorithm 4.1: Classifier Mini-Batch SGD Training

Data: $(x_1, y_1), ..., (x_N, y_N) \in X$
Result: Optimal network parameters $\hat{\theta}$
Initialization: $\theta = \theta_0$, $\eta = \eta_0$;
while algorithm has not converged **do**
  split the data into M random batches of $\frac{N}{M}$ (x, y) examples;
  while every batch has not been used as input **do**
    input current batch into the network;
    obtain the batch output label probabilities $p_i(\theta)$, $i = 1, ..., \frac{N}{M}$;
    compute $\mathcal{L}(\phi)$ for each batch example i;
    compute $\Delta\theta = -\eta \sum_i \frac{\partial \mathcal{L}}{\partial \theta}$;
    compute $\theta_k = \theta_{k-1} + \Delta\theta$;

4.2 Convolutional Variational Autoencoder

4.2.1 Architecture

In order to fulfill the main goals of this work, a Convolutional Variational Autoencoder will assume the role of a generative network, which is intended to be able to not only generate new instances of a certain class for data augmentation purposes but also to serve as an unsupervised mechanism that facilitates the task of novelty detection applied to AD diagnosis.

As mentioned in section 3.4, the VAE is composed of a inference model, parametrized by θ and a generative model, in turn parametrized by ϕ. This two components are embodied by encoding and decoding networks: in this case, since this work relies on the usage of neuroimages who are very complex and high-dimensional, these will assume the form of a CNN and a TCNN, respectively. The used Convolutional VAE is depicted in figure 4.2.

The target latent PDF will be a zero mean and unit variance multivariate Normal distibution. Through the inference encoder, relevant image features will be extracted and a set of Gaussian parameters μ (mean) and Σ that ideally are approximated to the target values will be generated. Afterwards, a latent

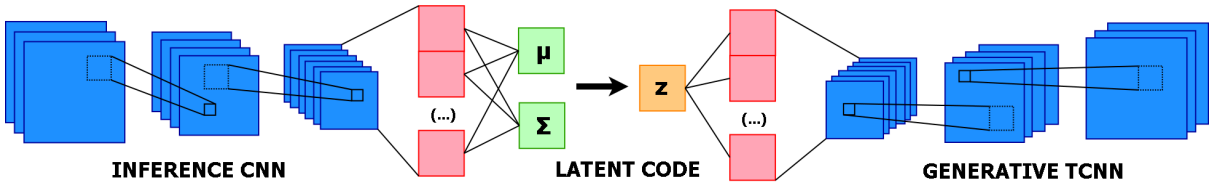

Figure 4.2: Proposed architecture of a convolutional variational autoencoder.

multivariate array will be sampled according to the obtained distribution, and then decoded through the generative network in order to rebuild the input.

A VAE is generative in the sense that the encoder learns how to select and map high-dimensional relevant information into a latent distribution expressed in a smaller dimensional space as compared to the input. Since the latent code does not represent a vector but a distribution, by sampling a latent vector z according to the obtained PDF (which by definition, is continuous in an infinite domain), this vector will not be the same for the different runs of the model with the same input x, as opposite to what happens with other types of autoencoders. Consequently, since z works as an input for the decoder network, it is expected that the output reconstruction image, although with some resemblances, shows to some extent different characteristics to those shown in the encoder input image.

4.2.2 Training Objective and Scheduling

As previously mentioned in section 3.4, the optimization problem that the VAE is keen to minimize is, given a dataset $x_1, ..., x_N \in \mathbf{R}^{\mathbf{n}\times\mathbf{m}\times\mathbf{p}}$

$$L_{VAE} = \sum_{i=1}^{N} \big[- E_{q_\phi(z|x_i)}[logp_\theta(x_i|z)] + D_{KL}(q_\phi(z|x_i)||p(z)) \big]. \tag{4.2}$$

in which the LHS is the negative expected log-likelihood that penalizes reconstructions far off the pretended input, and the RHS is a regularizer term, who imposes a chosen latent distribution (in this case, a unit variance and zero mean multivariate Normal PDF). In a ML point of view, the LHS of the cost function 4.2 can be seen as a sum of squared errors calculated between pixels of the input x and the reconstruction $\hat{x}$, since it too penalizes flaws in the approximation to the input. As previously seen in section 3.8, a backpropagation technique may be used to train the VAE, with the aid of the reparametrization trick in order to surpass the difficulty of having a non-deterministic step in the network's main frame. Again, the algorithm of VAE training (4.2) is showed assuming a SGD mini-batch approach for parameter tuning.

In practice, it has been shown that the cost function 4.2 may lead to reconstruction homogeneity due to poor latent disentanglement, which may be a direct result of disparity between possible value ranges

Algorithm 4.2: Mini-Batch SGD Training with Reparametrization Trick applied to a VAE

Data: $(x_1, y_1), ..., (x_N, y_N) \in X$
Result: Optimal network parameters $\hat{\Psi} = (\hat{\theta}, \hat{\phi})$
Initialization: $\Psi = \Psi_0 = (\theta_0, \phi_0)$, $\eta = \eta_0$;
while algorithm has not converged **do**
  split the data into M random batches of $\frac{N}{M}$ (x, y) examples;
  while every batch has not been used as input **do**
    generate $\epsilon_i \sim N(0, I)$, $i = 1, ..., \frac{N}{M}$;
    input current batch into the network;
    obtain the output reconstructions $\hat{x}_i(\Psi)$, $i = 1, ..., \frac{N}{M}$;
    compute $\mathcal{L}(\Psi, \epsilon)$ for each batch example i;
    compute $\Delta\Psi = -\eta \sum_i \frac{\partial \mathcal{L}}{\partial \Psi}$;
    compute $\Psi_k = \Psi_{k-1} + \Delta\Psi$;

between LHS and RHS of the equation. Consequently, variants of the cost function 4.2 [42] like:

$$L_{VAE} = \sum_{i=1}^{N} \big[- E_{q_\phi(z|x_i)}[log p_\theta(x_i|z)] + \beta D_{KL}(q_\phi(z|x_i)||p(z)) \big], \tag{4.3}$$

propose the addition of a term β which balances the input reconstruction capability and the latent-space distribution constraint. This can be seen as a trade-off between the degree of similarity between obtained and goal PDFs and quality of the reconstructions.

In the other hand, other works [43] [44] have pointed out another effect in which the KL term unintendedly becomes small (KL vanishing problem), resulting in uninformative features being learned and the latent code being ignored in most part by the decoder, who itself is able to reproduce viable reconstructions, shifting the behaviour of the VAE to a standard AE. These works point that the cause is due to initial uninformative latent codes that are the result of poor network initialization, thus proposing methods of varying β in order to avoid the KL vanishing term. Thus, in this work, the VAE will be trained using a simple KL-term annealing scheduling method, in which the term β starts at 0, and slowly grows until reaching its final value in a certain epoch, maintaining this same value until the end of the network's training.

4.2.3 Novelty Detection Modes

As previously mentioned in section 1.5, the task of anomaly detection has as primary goal the observation of abnormal patterns in data. In the context of the problem of AD classification, inliers are considered to be NC samples and outliers as MCI and AD images. In this work, a novelty detection approach will be followed: this means that in the training dataset, only inliers will be used, and the outlier patterns are discovered after the VAE is trained.

As previously mentioned, variational autoencoders are unsupervised networks which aim is to build

latent probabilistic representations of inputs by assimilating their most important features and using them to reconstruct and generate new instances accordingly. Using a novelty detection approach, the network (during training) will be exposed to only NC samples, thus implying that the VAE will learn distributions associated to only images that fall in this category. If the image is either MCI or AD, it's implied that not only it will produce outputs that little resemble the input but also dissimilarities between the target Gaussian distribution and the obtained one. Works on anomaly detection-based VAE's applied to medical diagnosis have sprung into existence recently (for instance, multiple sclerosis [45] or skin diseases [46] where only two classes are considered), and most define several metrics in which the different representation spaces of the VAE are evaluated. In short, these novelty detection modes are divided into three different categories:

- Detection in the latent feature space,
- Detection in the original input space,
- Detection as a function of the VAE loss.

In all of them, scores are produced for a certain neuroimage input into the network. The classification process holds two key steps and is pointed in algorithm 4.3 (this exemplifies the procedure in solving the trinary problem NC/MCI/AD).

Algorithm 4.3: Sample Classification Algorithm applied to the NC/MCI/AD problem

Data: $x_1, ..., x_N \in X_{test}$
Result: Labels $y_1, ..., y_N$ for X_{test}
Classification Thresholds: th_{anom},th_{AD};
while all samples haven't been classified **do**
  input x_i into network;
  produce output $y = y_i$ and PDF parameters (μ_i, Σ_i);
  calculate score $s_i = f(y_i, \mu_i, \Sigma_i)$;
  if $s_i \leq th_{anom}$ **then**
    y_i :=NC
  else
    if $s_i > th_{AD}$ **then**
      y_i :=AD
    else
      y_i :=MCI

MCI samples are expected to produce scores which are closer to NC samples when compared to AD images. The same rule applies for MCI-C and MCI-NC samples, with the latter corresponding to a state in which the brain is closer to being healthy than to develop dementia as compared to the first. The decision thresholds are learned through a classifier or by averaging the difference of mean scores obtained for every neighbour classes using a small image subset (e.g. for the NC/MCI/AD problem, th_{anom} and th_{AD} would be the mean of the averaged scores for the NC/MCI and MCI/AD samples,

respectively). In the following subsections, different types of scores are presented, which in turn allow for the label assignment of unknown neuroimages.

4.2.3.A Latent feature space

Latent space detection of anomalous classes can be done in several ways. One can use the regularizer term D_{KL} in equation (4.2) as a parameter of distinction, since an abnormal sample is likely to produce a distribution that diverges more from $p(z)$ when compared to the PDF generated by a NC sample. Besides this, considering that the latent space of a VAE trained on a normal control set is an effective representation of that same training data in this specific space, one can measure parameter distances (for example, mean differences in the Gaussian case), or even compute the minimum Bhattacharyya distance between distributions:

$$D = \min_{x \in X} D_B[q_\phi(z|x_{test}), q_\phi(z|x)], \tag{4.4}$$

in which D_B is the Bhattacharyya distance, a quantitative measure of similarity between distributions formally defined as

$$D_B = -ln[\int \sqrt{q_\phi(z|x_{test}), q_\phi(z|x)} dz]. \tag{4.5}$$

Besides this, a density-based measure is also applicable. For instance, in the normal distribution case, one can compute the value of average of the Gaussian distributions produced by X evaluated at the mean of the Gaussian distribution produced by x_{test}, which mathematically can be expressed as

$$D = q_X(E[q_\phi(z|x_{test})]) \tag{4.6}$$

where

$$q_X = \frac{1}{m} \sum_{x \in X} q_\phi(z|x) \tag{4.7}$$

is the average of all the produced Gaussian distributions. The value calculated by equation (4.6) is expected to be lower if the sample is abnormal when compared to a NC one. This occurs in this case due to the fact that the produced distribution in the latent space will be distant to the average Gaussian produced by the training set, so the value of the latter taken on the mean of the first will therefore be low.

4.2.3.B Original input space

If detecting differences in the original input space, a logical step to take is to evaluate the quality of the reconstruction: since the VAE is only trained with normal samples, it is expected that if a non-normal class sample takes the place as x_{test}, its reconstruction error will be bigger when compared to the error

commited when the input belongs to the normal class. For instance, one can compute a deterministic reconstruction error in the form of

$$D = ||x_{test} - E[p_\theta(x|E[q(z|x_{test})])]||_2^2, \tag{4.8}$$

where $||.||_2^2$ is the L2-norm, evaluated between the input x_{test} and the expected value of the distribution produced by the decoder p_θ as the result of that same input. Another choice could be a fully-stochastic reconstruction error, that uses samples from both distributions produced by the encoder and decoder and computes a mean error between them and x_{test} as shown in 4.9.

$$D = \text{mean}||x_{test} - x_i||_2^2, \;\; x_i \sim p_\theta(x|z_i) \; , \; z_i \sim q_\phi(z|x_{test}) \tag{4.9}$$

Yet another possibility would be applying distance and density based approaches as done for the latent space measurements, now done taken to account p_θ. Finally, one can also take into account the distance to the closest generated sample, which is no other than an optimization problem, finding a latent vector z for:

$$D = \min_z ||x_{test} - E[p_\theta(x|z)]||_2^2 \tag{4.10}$$

4.2.3.C VAE loss

Another metric consists in using the variational autoencoder's loss function(4.2) itself as a metric, which can be seen as a junction of analysis in the original and latent feature spaces. Again, as the VAE has only been exposed to normal control samples, its expected that if the input belongs to another class, it will produce a bigger loss value than the expected one and thus can be used as a measurement tool, using a Monte Carlo estimate to approximate the value of $E_{q_\theta(z|x_{test})}$.

5

Results, Experiments and Discussion

Contents

In this chapter, all results and experiments conducted in the scope of this thesis will be presented. First, the experimental setup in terms of hardware being used and the available images in the dataset are showcased. Afterwards, the VAE's performance is purely measured in terms of reconstruction performance, in which the produced latent space is presented and the generative ability of the model is evaluated. The classifying CNN is presented afterwards, evaluated in the proposed problems in chapter 4, given the original dataset and an augmented version. Finally, the novelty detection classifier results are shown for selected problems of AD detection.

5.1 Dataset and Experimental Setup

The dataset used in this work has been obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) [47], and consists of Fluorodeoxyglucose Positron Emission Topography (FDG-PET) functional neuroimages from subjects which may be characterized by three different stages of the disease: Normal Control (NC), Mild Cognitive Impairment (MCI) and Alzheimer's Disease (AD). Each patient may have a maximum of three brain scans that correspond to different timestamps of a continuous study (Baseline month, 12^{th} month and 24^{th} month), in which their status may change due to the progressive nature of the disease. In table 5.1, a summary of the existing images is given, along it gender information, age and Mini-Mental State Exam (MMSE) average scores and respective standard-deviation (in both, it is assumed that these values assume the form of a Gaussian distribution).

	NC	**MCI**	**AD**
Images	271	526	228
Gender(M/F)	180/91	346/180	136/92
Age - $\mu(\epsilon)$	76.96(4.82)	76.02(7.34)	76.64(7.05)
MMSE - $\mu(\epsilon)$	29.11(1.14)	26.57(2.87)	21.43(4.58)
Subjects (Total)		406	

Table 5.1: FDG-PET dataset summary table

It is rather important to assure that the variation of the age of the subjects is not high when comparing scans of different classes: this is due to the fact that the brain itself experiences natural changes with age growth, and since AD is a disease which is more common with individuals with age equal or bigger than 75 years old [2], it is of this work's best interest to assure that the solution captures differences in between classes due to the disease status and not age differences. This problem also concerns individuals in the same class: if the features captured in scans belonging to the same disease state are too heterogeneous, this class specific patterns will not be correctly learned by the network (something

that, assuming a novelty detection strategy like this work does, becomes critical). One of the goals of this thesis is also to solve MCI-C/MCI-NC problem assuming several time windows. For this problem, the number of images are presented in table 5.2.

	MCI-NC	MCI-C
Images (Immediate)	456	70
Images (12 months)	400	126
Images (24 months)	366	166

Table 5.2: Number of images available for the MCI-NC/MCI-C problem for different time windows.

Before being used, the neuroimages (of original size 128x128x60) are cropped (skull-stripping) and normalized (intensity values in between a range of 0 and 1). In order to facilitate the choice of architecture parameters, the images are afterwards padded with zeros, being its final size of 64x80x64 pixels. Since every patient may have more than one scan (corresponding to different timestamps), the data partitioning follows a subject-separating strategy: this means that it is assured that scans of a different subject are guaranteed to not occupy the ideally independent training and test sets (what would naturally inflate results).

All proposed architectures and models have been implemented using the Python-based Machine Learning library Keras [48] (version 2.2.4) with Google's TensorFlow backend [49] (version 1.13.1). Experiments have been conducted on a workstation with an Intel-i7 8700 3.20GHz CPU, 16GB RAM and a NVIDIA-GTX 1080 11GB GPU.

5.2 Supervised Classifier Network

As a first step, a Convolutional Neural Network (CNN) is built, with a similar structure to what will proposed with the Variational Autoencoder. It is intended to first solve the problem using a supervised framework in order to serve as a baseline performance metric to afterwards compare the scenario in which no labels are used during the network's training, following the method described in section 4.2.3. The network's chosen architecture is characterized in table 5.3 and is composed of 4 convolutional and pooling layers and 3 fully-connected layers, producing an output which corresponds to the input class probability.

5.2.1 Training Settings and Hyperparameters

The network is trained to minimize the binary cross-entropy loss function (as shown in 3.4 and in turn maximize the training set accuracy. The data used is partitioned following a subject-separation criteria

Layer Type	Specifications	Filters/Units	Output Size
Convolutional	Same Padding, Stride-2, ReLU, MaxPool 2x2x2	5x5x5x32	32x40x32x32
Convolutional	Same Padding, ReLU MaxPool 2x2x2	5x5x5x64	16x20x16x64
Convolutional		5x5x5x128	8x10x8x128
Convolutional		3x3x3x256	4x5x4x256
Fully Connected	ReLU, 50% dropout	512	512x1
Fully Connected	ReLU	128	128x1
Output	Sigmoid	2	2x1

Table 5.3: Architecture of the supervised classification network.

(80% of dataset is used for training, 10% for validation and 10% for testing), and its performance is evaluated using 10 fold cross-validation [50]. In order to train the proposed network, the Adam optimizer [27] is used, with initial learning rate $\eta_0 = 0.0001$,$\beta_1 = 0.9$ and $\beta_2 = 0.999$. The network training lasts for 50 epochs (80 in MCI-C/MCI-NC problems) and has a mini-batch size of 40 images per iteration. An early stopping [51] strategy is adopted, in which the final network parameters correspond to the training epoch that has resulted in the best performance in the validation set. It is also important to underline that the MCI-C/MCI-NC problems suffer, as table 5.2 suggests, of class imbalance [52], an therefore a weighted training strategy is applied, in which samples belonging to the dominant class with N samples are given unitary weight and samples belonging to the smaller class (with M samples) are entitled to an $\frac{N}{M}$ weight [52].

5.2.2 Performance

In order to evaluate the classifiers' performance, it is necessary to introduce some notation. In the considered binary classification problems, one can denote attributed labels as positives and negatives. Since one of the primary aims of this work is to detect the AD phase, the latter will be designated as positive, with its problem counterpart (either NC/MCI-NC) being known as negative. In the case of MCI-NC/MCI-C, the latter will be known as positive and the first as negative. Doing this, one may define the following quantitative measures that will be used to measure the CNN's performance:

- Accuracy (ACC) = $\frac{TP+TN}{TP+TN+FP+FN}$,
- Sensitivity (SENS) = $\frac{TP}{TP+FN}$,
- Specificity (SPEC) = $\frac{TN}{TN+FP}$,
- F1-Score = $2 \cdot \frac{SENS \cdot SPEC}{SENS+SPEC}$,

in which true positives (TP) are correctly predicted positives, false positives (FP) correspond to wrongly labelled samples which are actually negative and *vice-versa* for positive and false negatives (FP and FN, respectively).

5.2.2.A Original Dataset

Using the given dataset which was described in table 5.1, the classification results for the supervised CNN are presented in table 5.4.

Classes	ACC	SPEC	SENS	F1-SCORE
NC/AD	0.8925	0.8700	0.9106	0.8877
MCI/AD	0.8456	0.8780	0.7510	0.8094
MCI-C/MCI-NC (immediate)	0.8400	0.8400	0.8400	0.8400
MCI-C/MCI-NC (12 months)	0.7325	0.7250	0.7372	0.7310
MCI-C/MCI-NC (24 months)	0.6750	0.6700	0.6759	0.6729

Table 5.4: Classifier results for different classification problems.

Although comparing results between different methods in DL problems is a difficult task due to the immeasurable possible configurations, hyperparameters and ammount of resources (dataset size, available GPU which for instance affects the batch size or size of the dataset for instance, etc.), the results shown in chapter 2 (in particular in table 2.1) will serve as comparison metric to the results in table 5.4. Considering the NC/AD problem first, the proposed network reaches a mean accuracy of 89,25% with 91,06% rate of well identified Alzheimer's Disease cases, a value that is comparable to cutting edge results but overall slightly lower when compared for instance with the results of Bäckström et al. in [9] which achieve 90,1% accuracy and 93,26% sensitivity or the ROI aided solution of Zheng et al. in [11] with 97,27% accuracy.

The results in general also confirm the difficulty that is solving the problem of mild cognitive impairment conversion (MCI-NC/C problem) when compared to, for instance, the simpler problem NC/AD, with the CNN being unable to reach accuracy or sensitivity rates above the 84% mark (obtained for an instantaneous conversion). Increasing the conversion time window for 12 months, an accuracy of 73,25% and F1-score of 73,1% are obtained, slightly lower than those of Suk et al. [18]. For the longest term prevision of 2 years (24 months), an accuracy of 67,5% and sensitivity of 67,59% is produced, lower than the results of Choi et al. in [20] that present for a larger conversion period of 36 months an accuracy of 84,2% and specificity of 87,0%.

The major difficulty of identifying conversion cases when compared with simpler class discrimination problems (e.g. AD/NC) can also be confirmed when comparing the training and validation performance plots (obtained for 1 fold) in figures 5.1 that show the train and validation set performances during training for the problems of AD/NC and 12 month MCI-C/MCI-NC: as in the first problem (fig.5.1(a)) a

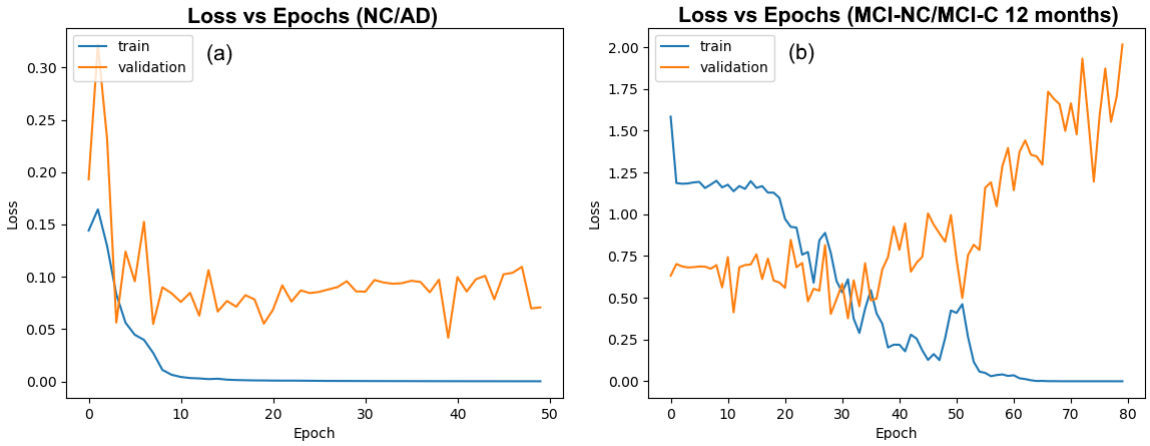

Figure 5.1: Train and validation loss values obtained with one training fold for: **(a)** NC/AD problem; **(b)** MCI-NC/MCI-C 12-month problem. The orange line denotes the loss value for the validation set, while the blue represents the loss calculated for the train set.

clear convergence of the loss curve for both sets is achieved during the learning phase of the network, the same is not verified for the latter (fig.5.1(b)) in which overfitting occurs (shortly after the 50^{th} training epoch, the validation loss starts increasing while the training loss decreases). This can be credited to the fact that the MCI sample features that indicate if a patient is likely to convert to AD or not are more subtle, with converter and non-converter images presenting a smaller feature variance when for instance compared with interclass images from the NC/AD or MCI/AD problems, and thus more difficult to learn.

Also, increasing the conversion timewindow causes the network to experience a performance drop, as seen in table 5.2.2 when comparing results obtained for different conversion time windows. This is expected due to the fact that samples corresponding to earlier timestamps show more subtle signs of an eventual development into Alzheimer's dementia when compared to images that are taken in time periods closer to the actual conversion. Basically, growing this time frame can be seen as increasing the variance of the neuroimages belonging to the MCI-C class, thus becoming more difficult to learn for a neural network.

5.2.2.B Augmented Dataset

In order to improve the quality of the network, one can perform data augmentation to the dataset. Since AD is a disease that affects the brain as a whole and due to the symmetry the brain presents, flipping left and right hemispheres may work as a variation of the original neuroimages and thus improve performance, increasing the number of training examples (in this case, doubling the number of available images). So, in this section, the results presented in table 5.5 correspond to the CNN performance when trained with the augmented dataset.

In comparison with table 5.4, one can immediately spot that performance increases. For instance,

Classes	ACC	SPEC	SENS	F1-SCORE
NC/AD	0.9100	0.9050	0.9132	0.9090
MCI/AD	0.8600	0.8875	0.8352	0.8601
MCI-C/MCI-NC (immediate)	0.8475	0.8400	0.8459	0.8479
MCI-C/MCI-NC (12 months)	0.7875	0.7850	0.7891	0.7870
MCI-C/MCI-NC (24 months)	0.7675	0.7650	0.7678	0.7664

Table 5.5: Classifier results for different classification problems with an augmented dataset.

the NC/AD labelling problem now achieves a 91% accuracy, 91,32% sensitivity, 90,5% sensitivity and 90,9% F1-score, a results that slightly tops the one achieved by Bäckström et al. in [9] but still inferior to other solutions like the automatic CNN of Hon et al. in [10] or the ROI based solution of Zheng et al. in [11]. The conversion problems also increase their performance metrics, with the 24-month case seing the biggest increase, reaching an 76,75% accuracy and 76,64% F1-score, barely topping the 18 month conversion classifications of Suk et al. [18] and Li et al. in [19] but still slightly below the approach proposed by Choi et al. [20]. In this problem specifically, data augmentation also tackles directly the class imbalance problem by increasing the number of samples that represent the less populated class (MCI-C), and thus can be seen as one of the possible reasons for improving the obtained results.

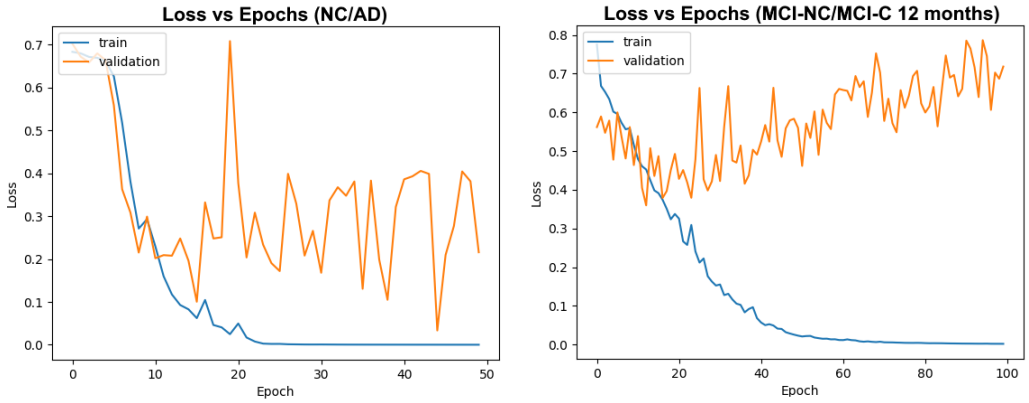

Figure 5.2: Train and validation loss values obtained with one training fold using the augmented dataset for: **(a)** the NC/AD problem; **(b)** the MCI-NC/MCI-C 12-month problem. The orange line denotes the loss value for the validation set, while the blue represents the loss calculated for the train set.

Finally, one may inspect the training and validation performances for the augmented dataset, with the plots obtained for the NC/AD and MCI-NC/MCI-C 12-month being presented in figure 5.2, where again it's noticeable that the conversion problem does not show a noticeable convergence and again presents overfitting after some training epochs. Comparing with the obtained performances for the original dataset in fig. 5.1, one can see that for both problems the minimum loss obtained values for the augmented dataset are slightly lower than those in the original. However, there are some inconsistencies during training, specifically in the NC/AD problem and for the same fold, where the validation loss shows a much higher variance with the augmented dataset than before.

5.3 Variational Autoencoder

In this part, the unsupervised variational autoencoder (VAE) is presented and the results are showcased. The initial idea was to train the VAE using only NC samples, and then to classify unknown images using a novelty detection process, which is no other than exposing in a test phase anomalous samples (either MCI or AD), analyze the various representation spaces produced by the VAE, define scores and thresholds based on a small subset of images (belonging to all possible classes) and test. However, there were problems in the latent space disentanglement and the *trade-off* between this and the reconstruction ability, which will be further explained, along with experiences and scenarios that tried to (unsuccessfully) solve the issue.

5.3.1 Training Settings and Hyperparameters

The chosen architecture of the VAE is described in table 5.6, and essentially is a copy of the classifier's architecture (described in 5.2.1) in the encoder, with the decoder assuming a symmetric shape with transposed convolutional and upsampling layers (forming a transposed convolutional neural network).

Inference Network (Encoder)				**Generative Network (Decoder)**			
Layer Type	**Specifications**	**Filters/Units**	**Output Size**	**Layer Type**	**Specifications**	**Filters/Units**	**Output Size**
Convolutional	Same Padding, Stride-2, ReLU, MaxPool 2x2x2	5x5x5x32	32x40x32x32	**Sampling**	Gaussian PDF Sampling Layer	—	128x1
Convolutional	Same Padding, ReLU MaxPool 2x2x2	5x5x5x64	16x20x16x64	**Fully Connected**	ReLU	128	128x1
Convolutional		5x5x5x128	8x10x8x128	**Fully Connected**	ReLU	512	512x1
Convolutional		3x3x3x256	4x5x4x256	**Deconvolutional**	Same Padding, ReLU MaxPool 2x2x2	3x3x3x256	8x10x8x128
Fully Connected	ReLU	512	512x1	**Deconvolutional**		5x5x5x128	16x20x16x128
Fully Connected	ReLU	128	128x1	**Deconvolutional**		5x5x5x64	32x40x32x64
Output (μ, ϵ)	Linear	128	128x2	**Deconvolutional + Compression**	ReLU (deconv) Sigmoid (compression)	5x5x5x32 5x5x5x1	64x80x64x1

Table 5.6: Architecture of the unsupervised Variational Autoencoder

Again, the data split follows a subject separation criteria, in which only normal control (NC) images are considered during training. From the datapoints available in this class, 80% are saved for the training set, 10% for the validation set and 10% for testing. The train set is also augmented by rotating the brain (flipping left and right hemispheres, similar to the method in 5.2.2.B). The Adam optimizer [27] is used during training, again with the configuration used previously in 5.2.1 (initial learning rate $\eta_0 = 0.0001$,$\beta_1 = 0.9$ and $\beta_2 = 0.999$). Training lasts for 450 epochs and the mini batch is of size of 12. Again, early stopping is applied with the best performance in the validation set (smallest loss value) as criteria. The latent space has size $k = 128$ and the parameter β responsible for the balance between the reconstruction term and KL-divergence follows an annealing schedule, reaching the final value of 0.01 in the 100^{th} epoch which is kept constant for the rest of the training iterations.

5.3.2 Results

The training performance for the test set[1] using the configuration described in subsection 5.3.1 is presented in figure 5.3. Performance-wise, its clear that the VAE converges to an assumed optimal solution that minimizes the cost function, so the next step is to analyze the several representation spaces in which the data is mapped and generated in the variational autoencoder.

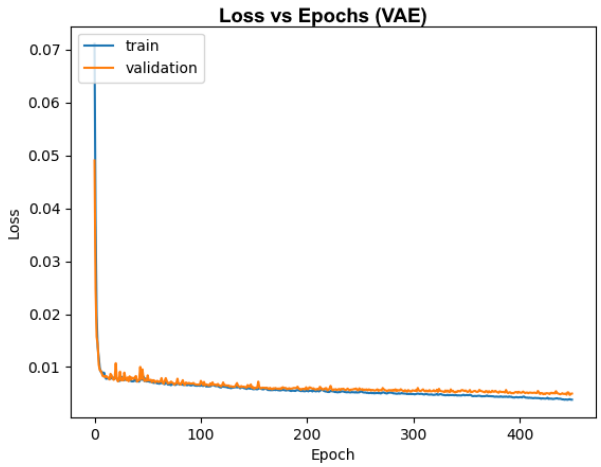

Figure 5.3: Training performance for the Variational Autoencoder. The blue line indicates the training set performance by epoch while the orange line represents the same quantity generated for the validation set.

5.3.2.A Original Feature Space

The next step is now to input images from the test set, which include samples belonging to anomalous classes (MCI and AD) and analyze not only the reconstruction performance but also the latent distribution parameters originated by the encoder. The reconstruction errors are calculated using a pixel mean squared error (MSE) between input and network output, and the results are presented in figure 5.4.

By training the VAE only with NC samples, the network is becoming proeficient in generating images that are likely similar to those in the training set. Taking into account that the disease is progressive, it would be expected that both anomalous classes (MCI and AD) would present higher reconstruction errors and that MCI images would present a MSE smaller than its counterpart AD (due to being an intermediate phase between a healthy and a state characterized by the fully developed disease). In figure 5.4(b), it can be seen that the class average MSE's follow what would be expected, with the average MSE for the AD class being the biggest and the NC the smallest. One can confirm this through the use of heatmaps that indicate the mean absolute error (MAE) obtained in each pixel for every class in the test set, as shown in figure 5.5 with three two-dimensional planes of the brain evaluated per class.

[1] Since a novelty detection approach is being used, the vast majority of NC samples was used to train this network due to its size. In order to avoid lack of NC examples in the results shown in this section, samples belonging to the validation set were added to the results.

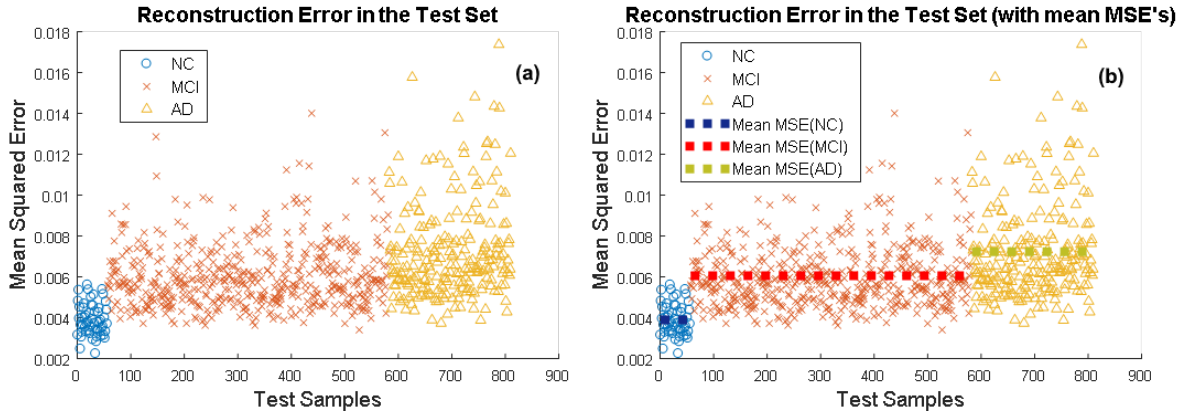

Figure 5.4: Reconstruction error obtained with the test set: **(a)** individual sample mean squared error(MSE); **(b)** individual sample MSE with the average class MSE signed with a line. Blue data represents the NC samples, red MCI and yellow AD.

In the different 2D slices of the heatmaps produced for each class in figure 5.5, areas with a bigger error are more noticeable in the anomalous classes (MCI and AD) due to the fact that training was only done with NC samples, thus implying that the network will not be able to proficiently generate anomalous features in the decoder after compressing the input. In figure 5.6, regions of interest (ROI) which are medically relevant for the diagnosis of AD are shown in the same three 2D planes where the heatmaps are represented.

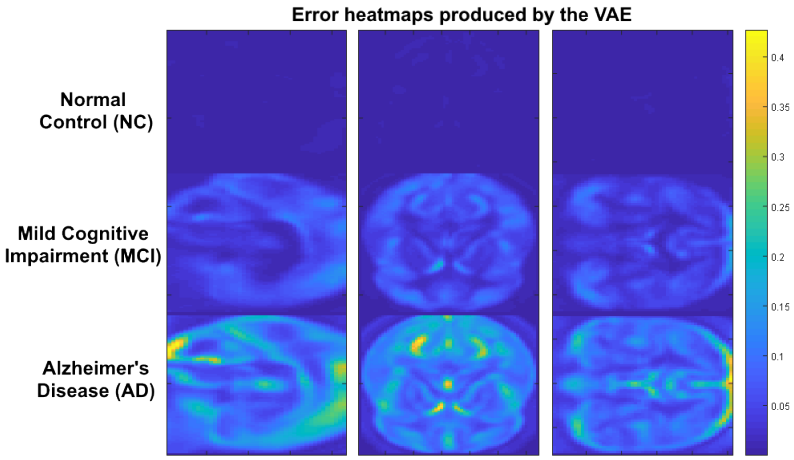

Figure 5.5: Error heatmaps produced for every class by the VAE in three different brain slices, with the decimal scale that indicates the range of committed mean absolute error.

By comparing the ROIs and heatmaps, it's visible that some areas with bigger error that were captured by the reconstruction heatmaps are indeed medically relevant. With this, both results in figures 5.4 and 5.5 show two clear advantages in practical terms:

- Although training the network in a complete unsupervised way, one is able to distinguish anomalous classes and their degree of severity, thus not requiring any labels to train the main network;

- Through the use of heatmaps, one can obtain specific information about which areas failed to reconstruct or approximate to the intensity values shown in the input, which can be associated to disease related pathologies.

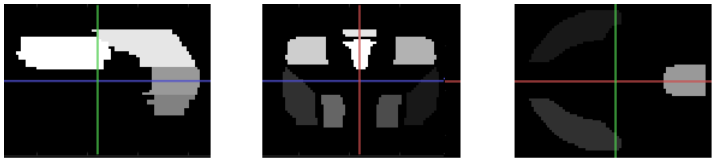

Figure 5.6: Regions of Interest (ROI) for Alzheimer's Dementia Diagnosis. The three slices correspond to the same three planes where the heatmaps are depicted in figure 5.5.

However, these conclusions are all made out of average measurements. Again, analyzing the sample-reconstruction errors in figure 5.4(a), one can notice that a considerable quantity of NC images produce reconstruction error values that are very close to the mean MSE produced by samples of the MCI class. The same conclusion can be applied to some MCI class samples that also present values equal or above the average AD sample mean squared error, something that jeopardizes the efficiency of a classifier whose task is defining decision thresholds for the errors committed in every class and eventually labelling an unknown neuroimage.

5.3.2.B Latent Feature Space

In order to improve the results in 5.3.2.A, it's important to remember that the decoder uses a multidimensional variable generated by the parametrized PDF (encoder output) as a starting point to generate an image similar to the output. Additionally, and as said previously in chapter 4.2.3, one can also analyze the latent space and the distributions produced by the test inputs to detect anomaly patterns, since the VAE learns how to encode NC images into a specific *a priori* chosen PDF (in this case, a multivariate unit Gaussian). Thus, it is expected that anomalous samples will fail at matching this distribution, producing parameters which are distant from the values normally produced in a training phase, and that will imply that the network cannot generate an image similar to the input.

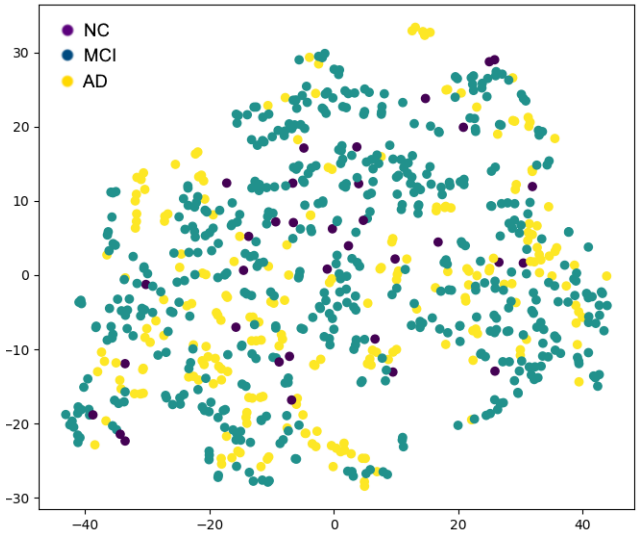

Figure 5.7: Visualization of the high-dimensional latent generated means using t-SNE. Purple, navy blue and yellow dots are generated by NC, MCI and AD image, respectively.

To evaluate this (with the same settings described in 5.6), and with the aid of dimensional reduction algorithm t-Distributed Stochastic Neighbor Embedding (t-SNE) [53] one can visualize the produced latent-space PDF parameters for the test set, as shown in figure where the means are plotted in a 2D space.

The figure 5.7 shows an unstructured latent space, revealing that the network is unable to form clusters or to, when exposed to a different class of images compared to the training set (in this case, MCI or AD), produce mean values that are substantially different from those which are generated by NC images. This suggests that the reconstruction errors shown in figures 5.4 and 5.5 are affected by the natural variance existing between healthy and anomalous brains. Also, the progressive nature of the disease implies that MCI neuroimages will present intensity values which are less distant to healthy ones as in comparison to AD datapoints, so it is expected that their average MSE is lower than the latter.

5.3.3 The VAE Objective Trade-off

Looking back at the VAE objective in eq. 4.2, and as mentioned earlier in section 4.2, the parameter β plays a key role in assuring a balance between the trade-off of input reconstruction and latent space structure. Since in 5.3.2.B it can be seen that the latent space suffers a lack of disentanglement, the value of β is increased from 0.01 to 0.1, in order to try to achieve better latent space disentanglement and consequently, increase the reconstruction errors in anomalous classes. In figure 5.8, the visualization of the latent space and reconstruction errors measured in the original feature space are presented for a value of $\beta = 0.1$.

Increasing the value of β actually had the opposite effect to what would be expected: as seen in figure

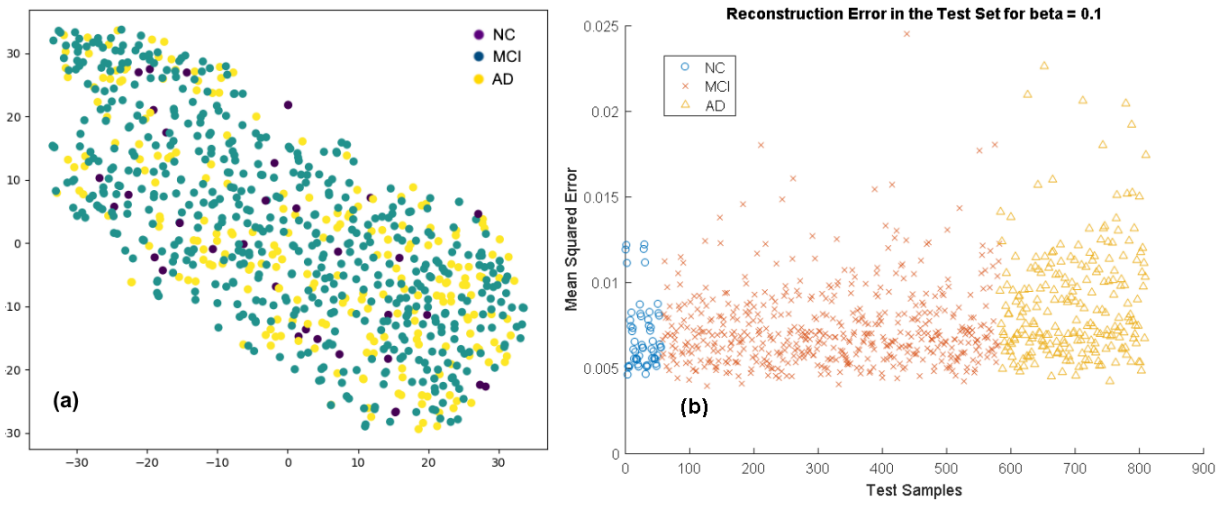

Figure 5.8: Results for VAE with $\beta = 0.1$: visualization of the high-dimensional latent generated means using t-SNE(a) and the reconstruction MSE(b)

5.8(a), the originated means got closer to each other but without significant latent disentanglement. And, as expected due to increasing the significance of the KL-divergence term, reconstruction MSE's are now very similar in between classes: this happens due to the fact that the constraint put in the latent space being raised, causing the latent PDF parameters to be input independent and generate very similar outputs.

An example is given in figures 5.9, in which same 2D cuts of three different class neuroimages originate very similar and almost indistinguishable outputs, what can be perceived as underfitting.

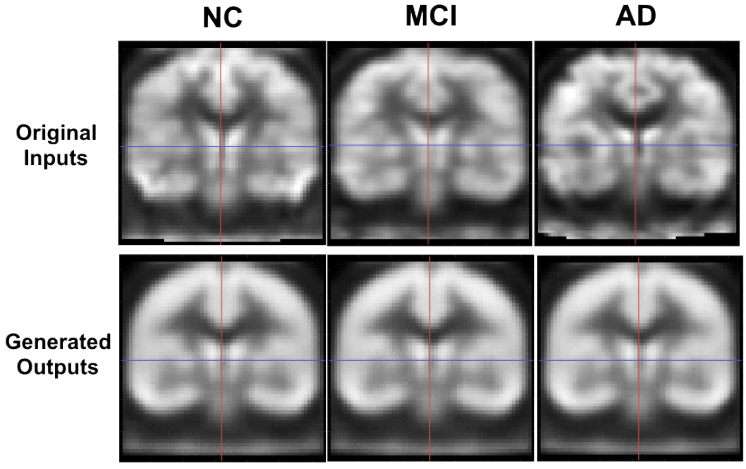

Figure 5.9: Example of three different 2D slices of generated neuroimages with different inputs, where underfitting is apparent.

5.3.4 Latent Space Dimensionality

Another option to try and increase latent space meaningfulness would be to decrease the dimensions k of the latent space, a hyperparameter whose value deeply affects a variational autoencoder and mainly its ability to reconstruct the input, something that is crucial for a novelty detection problem:

- if k is too big, reconstructions may become class independent: this means that anomalous images can be reconstructed as well as the healthy ones due to the lack of feature selection made by the encoder;
- if k is too small, the data compression enforced in the encoder is too harsh and can lead to sub-optimal reconstructions, which would cause underfitting and homogeneous outputs regardless of the input.

One may find this trade-off similar to the one defined by the hyperparameter β, that in turn controls the regularization term in the latent space as shown in subsection 5.3.3.

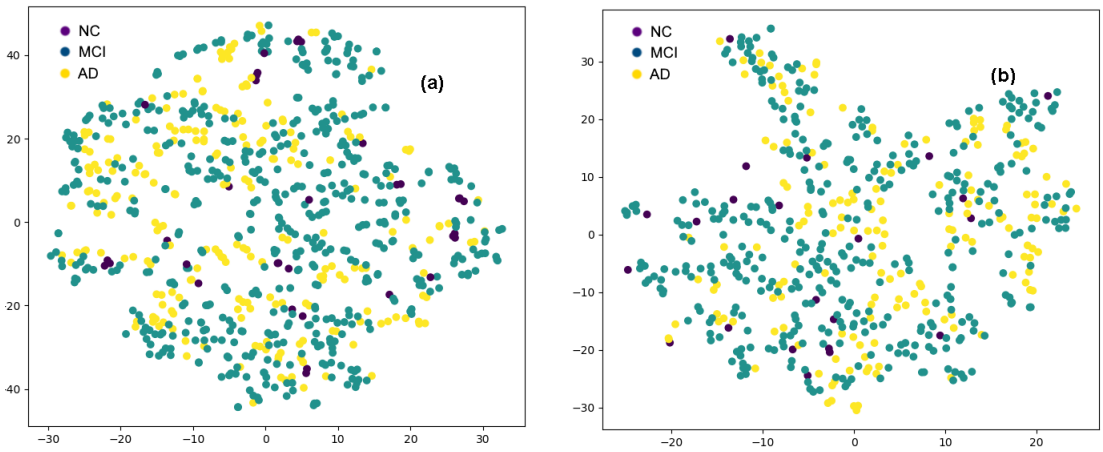

Figure 5.10: Latent means generated by the encoder for latent space dimensions of $k = 32$ (a) and $k = 256$ (b)

However, the bigger concern is achieving good latent disentanglement as to force the decoder to reconstruct healthy samples and fail when the image is anomalous, while still maintaining a good approximation to the target PDF. Since good disentanglement has not been achieved yet, it's logical to try and vary the latent space dimensions, so maintaining the original $\beta = 0.01$ several values of k were experimented. For the sake of brevity, only two examples with the latent means generated for $k = 32$ and $k = 256$[2] are shown (again in a smaller dimension plane using t-SNE) in figure 5.10.

From the plots presented in figure 5.10, no major differences in terms of disentanglement are achieved. There are several combinations of β and k values which were for the most tested but all presented results

[2]For k=256 the final dense layers of size 128 are replaced with dense layers of 256 units each

are similar to what was been shown earlier and do not add anything relevant.

5.3.5 Network Dimensions

Aside from the results obtained, from a practical standpoint it's important to compare the memory and time consumption between the proposed supervised and unsupervised networks. In table 5.7, a summary table that compares the number of layers (the ones available in tables 5.3 and 5.6 plus the input and output layers), number of trainable parameters (weights and biases), memory requirement for every sample during training (that takes into account every intermediate tensor produced in the environment for the backpropagation algorithm to take place) and the average training time necessary to train the network in the experiments shown before.

	Convolutional Neural Network (CNN)	**Variational Autoencoder (VAE)**
Number of Layers	12	24
Number of Parameters	6 954 242	22 578 689
Memory Required per Image during Training (megabytes)	44 MB	183 MB
Average Training Time	1 hour	9 hours

Table 5.7: Comparison of the two networks in terms of size, memory required (per image) and the average training time using the horizontal mirroring augmented dataset.

From table 5.7, it's noticeable that the VAE drains more resources than the CNN, by doubling the number of layers, having more than the triple of the parameters and almost quadrupling the required memory for every input image, which can harm the performance of the network and require a much larger amount of training samples. This also has drawbacks in training, which lowers the possible batch size for the VAE (12 when opposed to 50 used in CNN) and increases the elapsed training time (average approx. 9 hours as compared to only 1 using a CNN). As for the memory, it also limits how much one can actually increase the latent space dimensions, something that would mean more training time and demand more training examples in order to work.

6

Conclusions and Future Work

Contents

In this work a final summary of the work and main contributions is given, with advantages and disadvantages of the proposed approach, main difficulties found during the process and future work ideas that can improve the quality of the proposed solution.

6.1 Main Difficulties and Overall Conclusions

Automatic diagnosis of Alzheimer's Disease using Deep Learning methods is a problem far from being trivial. Dataset complexity, deep networks and low computational efficiency are usual challenges that this branch of ML usually faces.

Comparing the classic CNN approach (very popular in current state of the art approaches as shown in chapter 2) and the convolutional VAE-based approach, one can argue that the first is superior in terms of performance. The VAE struggles a lot with hyperparameter tuning, being sensible to changes with the latent space dimensionality k and the regularizing parameter β which control the structuring of the latent space as opposed to the rate of error committed in reconstructions, whose balance is key to achieve success using a similar method to this one.

This balance poses as the main problem, since the VAE's main goal is to encode input information into an *a priori* chosen PDF, whose samples may not be adequate or informative enough to serve as an input for the generative part of the network (decoder), which may be the cause for the unstructured latent spaces shown in the results. In fact, recent studies point out that problems with uninformative latent spaces and vanishing KL-terms are the reason behind unstructured PDF parameters and sub-optimal reconstructions [43] [44]. There also have been studies that state that the decoder may ignore the latent code and learn during the training process to rebuild samples from any input, concatenating the encoding-decoding path [44].

To make matters worse, the relationship described in the VAE's objective function is still poorly misunderstood in practice, with this hyperparameter fixed values being a lot of times defined in an empirical way rather than analytically, and to the added cause that KL-divergence and usual reconstruction metrics may have different value ranges and whose balance needs to be taken into account when choosing β.

Also, an additional difficulty is the huge computational load and very high training and testing time per simulation, caused by the low batch size, which is due to GPU limitations triggered by the networks considerable size. This is a natural consequence of the curse of dimensionality, caused by the huge and very complex image set that is the ADNI PET dataset.

However, this approach brings a smaller dependency on labelled datasets as the majority of state of the art solutions present, due to the unsupervised nature of the VAE. In terms of an user-friendly tool, supervised networks also fail to indicate (in a trivial way) in which brain regions the disease is noticeable,

something that may aid diagnosis in an intuitive way (although work in the DL field of interpretable CNN's is being made to solve this issue [54] [55], which itself may not be easy to interpret by someone not familiar with this kind of techniques), while VAE's may be used as a tool that indicates areas that may be (or not) associated to the disease. Additionally, the VAE is a generative network: with proper tuning, it can serve also as an augmentation mechanism to increase the dataset's size in an unsupervised way and exponentialize performance in supervised solutions or serve as tool to tackle data inbalance issues, like the ones that happen in MCI-C/MCI-NC problems and that may cause issues in its performance. The structural problems faced during this work may be a challenge to those who decide to follow this approach, but the rewards in a practical application are bigger than any supervised solution.

6.2 Future Work

For any future work related with this thesis, one has to assess the main problems explained before that this approach faces (especially when exposed to neuroimages). Some future lines of work may be:

- Applying adaptable methods that change the value of β to this problem may function to increase meaningfulness in the latent code [43] instead of empirically adjusting it, since it has been proven that the architecture may ignore the latent code intentionally [44];
- Changing the chosen distribution from the usually chosen multivariate unit Gaussian (that has also been pointed out as cause of VAE sub-optimal performance due to its simple and uninformative nature). Other variations of the VAE using different PDFs have been proposed, for instance, making use of spherical [56] or t-Student [57] distributions.
- Altering or varying the objective loss function of the network, like Ladder [58] or Info-VAE's [59].
- Using a semi-supervised version of the problem, which basically consists of adding a classifier network to the encoder and use the produced labels as a part of the latent space to increase latent code relevancy [60].

Bibliography

[1] Organization for Economic Co-operation and Development (2018): ”Care Needed - Improving the Lives of People with Dementia”. OECD Health Policy Studies, pp. 19-21

[2] Alzheimer’s Association (2018): ”2018 Alzheimer’s disease facts and figures”. *Alzheimer’s Dementia*, *14*(3), pp. 367-429.

[3] B.C. Dickerson, T.R. Stoub, R.C. Shah, R.A. Sperling, R.J. Killiany, M.S. Albert, B.T. Hyman, D. Blacker and L. deToledo-Morrell(2011): ”Alzheimer-signature MRI biomarker predicts AD dementia in cognitively normal adults”. *Neurology*, *76*(16), pp. 1395-1402

[4] B. Mwangi, T.S. Tian, and J.C. Soares (2014): ”A review of feature reduction techniques in neuroimaging.” *Neuroinformatics*, *12*(2), pp. 229-244.

[5] I. Goodfellow, Y. Bengio, A. Courville (2016): ”Deep learning”. Cambridge: MIT press

[6] F. Chollet (2017): ”Deep learning with python”. Manning Publications Co.

[7] P.J. Rosch(2016): ”Stress, Alzheimer’s and Memory Loss”. *The American Institute of Stress.* `http://www.stress.org/stress-alzheimers-and-memory-loss/`

[8] Chandola, V., Banerjee, A., Kumar, V. (2009): ”Anomaly detection: A survey”. *ACM computing surveys (CSUR)*, *41*(3), 15.

[9] K. Bäckström, M. Nazari, I. Yu-Hu Gu and A.S. Jakola(2018): “An efficient 3D deep convolutional network for Alzheimer’s disease diagnosis using MR images”. *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 149–153. IEEE.

[10] M.Hon and N.Khan (2017): ”Towards Alzheimer’s Disease Classification through Transfer Learning”. *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1166-1169. IEEE.

[11] X. Zheng, J. Shi, Y. Li, X. Liu, and Q. Zhang(2016): ”Multi-modality Stacked Deep Polynomial Network based Feature Learning for Alzheimer’s Disease Diagnosis”. 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI), pp. 851-854. IEEE.

[12] S. Sarraf and G. Tofighi (2016): ”DeepAD: Alzheimer′ s disease classification via deep convolutional neural networks using MRI and fMRI”. *BioRxiv*, 070441.

[13] T.D. Vu, H.J. Yang, V. Q. Nguyen, A-Ran Oh and M.S. Kim(2017): ”Multimodal learning using convolution neural network and Sparse Autoencoder”. *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 309-312. IEEE.

[14] K. Aderghal, A. Khvostikov, A. Krylov, J. Benois-Pineau, K. Afdel and G. Catheline(2018): ”Classification of Alzheimer Disease on Imaging Modalities with Deep CNNs Using Cross-Modal Transfer Learning”. *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*, pp. 345-350. IEEE.

[15] M. Liu, D. Cheng, and W. Yan (2018): ”Classification of Alzheimer’s Disease by Combination of Convolutional and Recurrent Neural Networks Using FDG-PET images”. Frontiers in neuroinformatics, 12(35)

[16] E. Jabason, M.O. Ahmad and M.S. Swamy (2018): ”Deep Structural and Clinical Feature Learning for Semi-Supervised Multiclass Prediction of Alzheimer’s Disease.” *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 791-794. IEEE.

[17] M. Liu, J. Zhang, E. Adeli and D.Shen(2017): ”Deep Multi-Task Multi-Channel Learning for Joint Classification and Regression of Brain Status”. *2017 Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp 3-11. Springer, Cham.

[18] H.I. Suk, S.W. Lee, D. Shen and the Alzheimer’s Disease Neuroimaging Initiative(2014): ”Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis”. *NeuroImage, 101*, pp. 569-582.

[19] F. Li, L. Tran, K.H. Thung, S. Ji, D. Shen and J. Li(2015): ”A robust deep model for improved classification of AD/MCI patients”. *IEEE Journal of Biomedical and Health Informatics, 19*(5), pp. 1610-1616. IEEE.

[20] H. Choi, K.H. Jin and Alzheimer’s Disease Neuroimaging Initiative (2018): ”Predicting cognitive decline with deep learning of brain metabolism and amyloid imaging.” *Behavioural brain research*, *344*, 103-109.

[21] B. Lim and M. Schaar (2018): ”Forecasting Disease Trajectories in Alzheimer’s Disease Using Deep Learning.” arXiv preprint, arXiv:1807.03159.

[22] B.Lim and M. Schaar (2018): ”Disease-Atlas: Navigating Disease Trajectories using Deep Learning.” Machine Learning for Healthcare Conference 2018, pp. 137-160.

[23] M. Nguyen, N. Sun, D.C. Alexander, J. Feng, and B.T. Yeo (2018): "Modeling Alzheimer's disease progression using deep recurrent neural networks." *2018 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, pp. 1-4. IEEE.

[24] C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall (2018): "Activation Functions: Comparison of trends in Practice and Research for Deep Learning." *arXiv preprint*, arXiv:1811.03378.

[25] D.E. Rumelhart, G.E. Hinton and R.J. Williams (1988): "Learning representations by back-propagating errors." *Cognitive modeling*, *5*(3), 1.

[26] M.D. Zeiler (2012): "ADADELTA: an adaptive learning rate method." *arXiv preprint* arXiv:1212.5701.

[27] D.P. Kingma and J. Ba (2014): "Adam: A method for stochastic optimization." *arXiv preprint* arXiv:1412.6980.

[28] K. Fukushima (1988): "Neocognitron: A hierarchical neural network capable of visual pattern recognition." *Neural networks*, *1*(2), pp. 119-130.

[29] A. Krizhevsky, I. Sutskever and G.E. Hinton (2012): "Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp. 1097-1105.

[30] K. Simonyan and A. Zisserman (2014): "Very deep convolutional networks for large-scale image recognition." *arXiv preprint* arXiv:1409.1556.

[31] K. Simonyan and A. Zisserman (2014): "Two-stream convolutional networks for action recognition in videos." *Advances in neural information processing systems* pp. 568-576.

[32] E. Grefenstette, P. Blunsom, N. de Freitas and K.M. Hermann (2014): "A deep architecture for semantic parsing." *arXiv preprint* arXiv:1404.7296.

[33] D. Scherer, A. Müller and S. Behnke (2010): "Evaluation of pooling operations in convolutional architectures for object recognition." *2010 International conference on artificial neural networks (ICANN 2010)*, pp. 92-101.

[34] J.T. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller (2014): "Striving for simplicity: The all convolutional net." arXiv preprint arXiv:1412.6806.

[35] M. D. Zeiler, D. Krishnan, G.W. Taylor and R. Fergus (2010): "Deconvolutional networks." *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, pp. 2528-2535. IEEE.

[36] P. Vincent, H. Larochelle, Y. Bengio and P.A. Manzagol (2008): "Extracting and composing robust features with denoising autoencoders." *Proceedings of the 25th international conference on Machine learning (ICML 2008)*, pp. 1096-1103. ACM.

[37] D.P. Kingma and M. Welling (2013): "Auto-Encoding Variational Bayes." *arXiv preprint* arXiv:1312.6114.

[38] A. Ng (2011): "Sparse autoencoder." *CS294A Lecture notes*, 72(2011), pp. 1-19.

[39] J. Masci, U. Meier, D. Cireşan and J. Schmidhuber (2011): "Stacked convolutional auto-encoders for hierarchical feature extraction." *International Conference on Artificial Neural Networks (ICANN 2011)*, pp. 52-59. Springer, Berlin, Heidelberg.

[40] D.P. Kingma (2017): "Variational inference deep learning: A new synthesis." PhD Thesis, University of Amsterdam

[41] C. Doersch (2016): "Tutorial on variational autoencoders." *arXiv preprint* arXiv:1606.05908.

[42] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, N. Watters, G. Desjardins and A. Lerchner(2017): "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". *5th International Conference on Learning Representations (ICLR)*, *2*(5), 6.

[43] S.R. Bowman, L. Vilnis, O. Vinyals, A.M. Dai, R. Jozefowicz and S. Bengio (2015): "Generating sentences from a continuous space". *arXiv preprint* arXiv:1511.06349.

[44] X. Liu, J. Gao, A. Celikyilmaz and L. Carin (2019): "Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing." *arXiv preprint* arXiv:1903.10145.

[45] A. Vasilev, V. Golkov, I. Lipp, E. Sgarlata, V. Tomassini, D.K. Jones and D. Cremers(2018): "q-Space Novelty Detection with Variational Autoencoders". arXiv preprint, arXiv:1806.02997.

[46] Y. Lu and X. Peng(2018): "Anomaly detection for skin disease images using variational autoencoder". *arXiv preprint*, arXiv:1807.01349.

[47] Alzheimer's Disease Neuroimaging Initiative (ADNI).
`http://adni.loni.usc.edu/`

[48] F. Chollet (2015): "Keras".
`https://keras.io/`

[49] M.Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng (2016): "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint* arXiv:1603.04467.
`https://www.tensorflow.org/`

[50] R. Kohavi(1995): ”A study of cross-validation and bootstrap for accuracy estimation and model selection.” *Ijcai*, *14*(2), pp. 1137-1145.

[51] Y. Yao, L. Rosasco and A. Caponnetto (2007): “On early stopping in gradient descent learning”. *Constructive Approximation*, *26*(2), pp. 289–315.

[52] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng and P.J. Kennedy(2016): ”Training deep neural networks on imbalanced data sets.” *International Joint Conference On Neural Networks (IJCNN 2016)*, pp. 4368-4374. IEEE.

[53] L. Maaten and G. Hinton (2008): ”Visualizing data using t-SNE.” *Journal of Machine Learning Research*, *9*(Nov), pp. 2579-2605.

[54] D. Garcia-Gasulla, F. Parés, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta and T. Suzumura (2018): ”On the behavior of convolutional nets for feature extraction.” *Journal of Artificial Intelligence Research*, *61*, pp. 563-592.

[55] Q. Zhang, Y. Nian Wu and S.C. Zhu (2018): ”Interpretable convolutional neural networks.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8827-8836.

[56] J. Xu and G. Durrett (2018): ”Spherical latent spaces for stable variational autoencoders.” *arXiv preprint* arXiv:1808.10805.

[57] H. Takahashi, T. Iwata, Y. Yamanaka, M. Yamada and S. Yagi (2018): ”Student-t Variational Autoencoder for Robust Density Estimation.” *International Joint Conferences on Artificial Intelligence (IJCAI)*, pp. 2696-2702.

[58] C.K. Sønderby, T. Raiko, L. Maaløe, S.K. Sønderby and O. Winther (2016): ”Ladder variational autoencoders.” *Advances in neural information processing systems*, pp. 3738-3746.

[59] Zhao, S., Song, J., Ermon, S. (2017). ”Infovae: Information maximizing variational autoencoders.” *arXiv preprint* arXiv:1706.02262.

[60] D.P. Kingma, S. Mohamed, D.J. Rezende and M. Welling (2014): ”Semi-supervised learning with deep generative models.” *Advances in neural information processing systems*, pp. 3581-3589.

A

Evidence Lower Bound and Kullback-Leibler Divergence

This appendix gives mathematical proof for the decomposition made in the optimization problem 3.15 regarding variational inference in chapter 3 section 3.4.2.C, in which the Kullback-Leibler (KL) Divergence is decomposed into the sum of a constant factor and the Evidence Lower Bound (ELBO). This demonstration is based in [40].

Given an inference model $q_\theta(z|x)$ which approximates the true posterior $p_\phi(z|x)$ according to the optimization problem

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, D_{KL}(q_\theta(z|x)||p_\phi(z|x)), \tag{A.1}$$

one can have

$$\begin{aligned} log\ p_\phi(x) &= E_{q_\theta(z|x)}[log p_\phi(x)] \\ &= E_{q_\theta(z|x)}\left[log\ \frac{p_\phi(x,z)}{p_\phi(z|x)}\right] \\ &= E_{q_\theta(z|x)}\left[log\ \frac{p_\phi(x,z)}{q_\theta(z|x)}\frac{q_\theta(z|x)}{p_\phi(z|x)}\right] \\ &= \underbrace{E_{q_\theta(z|x)}\left[log\ \frac{p_\phi(x,z)}{q_\theta(z|x)}\right]}_{-\mathcal{L}_{ELBO}} + \underbrace{E_{q_\theta(z|x)}\left[log\ \frac{q_\theta(z|x)}{p_\phi(z|x)}\right]}_{D_{KL}(q_\theta(z|x)||p_\phi(z|x)\geq 0)} \end{aligned} \tag{A.2}$$

In equation A.2, the first term acts as the lower bound of the likelihood of the data, due to the fact that the second term corresponds to a KL divergence between the inferred and real posteriors, which by definition is always positive. This bound is known as variational lower bound or Evidence Lower Bound

(ELBO), and is written as

$$ELBO = -\mathcal{L}_{ELBO} = E_{q_\theta(z|x)}[logp_\phi(x|z) - D_{KL}(q_\theta(z|x)||p_\phi(z)) \tag{A.3}$$

Taking into account the result in equation A.2, it's trivial to notice that the optimization problem in A.1 may be re-written as

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[log\ p_\phi(x) + ELBO \right], \tag{A.4}$$

meaning that minimizing the KL divergence between the inferred and real posteriors is achieved by maximizing the ELBO, and thus minimizing the objective function

$$\mathcal{L}_{ELBO} = -E_{q_\theta(z|x)}[logp_\phi(x|z)] + D_{KL}(q_\theta(z|x)||p_\phi(z)), \tag{A.5}$$

thus proving the validity of the statement made in section 3.4.2.B.