Universidade do Minho
Escola de Engenharia

Ivo Miguel Marques Ramalhosa

Analysis of Brain Connectivity In fMRI

A Deep Learning Approach

Tese de Mestrado

Engenharia Biomédica/Ramo de Informática Médica

Trabalho efetuado sob a orientação de

Professor Doutor Victor Alves

Doutor Paulo Marques

Outubro de 2017

STATEMENT OF INTEGRITY

DECLARATION

Full Name: Ivo Miguel Marques Ramalhosa

Title dissertation: Analysis of Brain Connectivity In fMRI – A Deep Learning Approach

Supervisor: Professor Doctor Victor Manuel Rodrigues Alves

Finish year: 2017

Master's name: Master in Biomedical engineering

Specialization branch: Medical informatics

School: School of engineering

Departments: Department of informatics

Acknowledgements

First of all, I want to thank my supervisor Professor Victor Alves for all knowledge and material support, which made this work possible. It has been a long learning way since the beginning, which without my supervisor help will not be possible. Also, a special thanks to my colleague Dr. Paulo Marques for his experience sharing and precious advices. As future biomedical engineer, it is inspiring to me having both as supervisor and co-supervisor.

Then, I want to thank all team members in the Neuroimaging ICVS lab for the adventures lived.

A special thanks to my mother and father, sisters, they always supported me unconditionally. Unfortunately it will be not possible to share this Thesis with my grandfather, which passed away in last October. Therefore, I want to give a special thanks to my grandfather José Luís, which made me a person that I am today. Thank you all for your support.

The last thanks goes to the most important person on my life, my partner in work and love, my beloved Filipa.

I dedicate this dissertation to my family, my love Filipa and to my grandfather...

ABSTRACT

The brain functional connectivity extracted from rs-fMRI has been used as a powerful tool to study the different networks in the brain. This neuronal network, found in normal condition, can be associated to different cognitive processes. The applicability of these networks in the future is promising, since is a greater technique to study the effects of several diseases or even treatments on normal brain functional connectivity. Firstly, this question should be addressed: *are these networks possible to be described and to be used as features to classify a group or a particular subject?.*

In order to answer this question, it was settled the use of a Machine Learning method, which has been developed great advances in the recent years, due the good performances in the Deep Learning (DL) method. Therefore, it was created a workflow since the beginning, started with data acquisition until the application of DL methods and the process of creation and fine-tune of these models. In the end, several studies using the functional connectivity were done, namely the assessment of the brain functional connectivity to be used as a "fingerprint". Additionally, it were performed some tests regarding the groups' classification.

After settled the correct approach and validate the DL framework, the "fingerprint" study showed a great improvement on impairment classification, even for simple models. We proved that rs-fMRI can be use in research field to identify singular brain patterns as well as the differences between the subjects, which could be applied as group differentiator in a population.

KEYWORDS: Medical Informatics - Neuroimaging – fMRI – Functional Connectivity – Deep Learning.

RESUMO

A conectividade funcional cerebral extraída de imagens de rs-fMRI demonstrou um potencial adquirido no estudo das diferentes redes existentes no cérebro. Relativamente a estas redes, estas são associadas a diferentes processos cognitivos e sensoriais, ou até simplesmente ao funcionamento normal do cérebro. O uso promissor destas redes é acrescido quando aplicado ao estudo de efeitos de diversas doenças, ou até tratamentos que afetam a normal conectividade funcional cerebral. Mas, primeiramente, surge a questão: *são estas redes possíveis de serem descritas e usadas como características para classificar um determinado grupo ou um indivíduo em particular?.*

Assim, para responder à questão proposta foi definido o uso de um método de Machine Learning com grandes avanços nos últimos anos devido ao bom desempenho – o método de Deep Learning (DL). Foi então criado um fluxo de trabalho desde a aquisição de dados à aplicação de métodos de DL, seguido de uma framework que lida com o processo de criação e ajuste dos parâmetros dos modelos DL. Posteriormente foram feitos diversos estudos usando a conectividade funcional, estática e dinâmica, para nomeadamente estudar se a conectividade funcional cerebral pode ser usada como "impressão digital" e se tem melhor desempenho que outros métodos já aplicados neste tipo de estudo. Adicionalmente, foram realizados alguns testes relativamente à classificação e regressão de grupos.

Em conclusão, foi constatado e validado que esta abordagem e esta framework de DL são boas escolhas. No estudo da "impressão digital", os resultados melhoraram imenso, mesmo usando modelos simples. Consequentemente, foi provado que a rs-fMRI pode ser usada para estudos de padrões singulares do cérebro e nas diferenças entre sujeitos, o mesmo aplicado à classificação em grupo na distinção dos mesmos.

PALAVRAS-CHAVE: Informática Médica – Neuroimagem – fMRI – Conectividade Funcional – Deep Learning.

CONTENTS

List of Figures

LIST OF TABLES

List of Abbreviations, Initials and Acronyms

DNN

Deep Neural Networks 28

DSR

Design Science Research 7

DSRP

Design Science Research Process 7, 8

DTI

Diffusion Tensor Imaging 12, 63, 64

EEG

Electroencephalography 13

EPI

Echo Planar Imaging 15, 16, 51, 53, 184

FA

Fractional anisotropy 51, 54

FC

Functional Connectivity........ 2, 3, 4, 5, 6, 12, 16, 17, 18, 67, 73, 74, 75, 77, 78, 82, 84, 85, 86, 87, 92, 116, 161

fMRI

functional Magnetic Resonance Imaging ....I, iii, v, vii, xxv, xxvi, 1, 2, 3, 4, 5, 8, 9, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 48, 49, 50, 51, 52, 53, 56, 63, 64, 65, 67, 68, 69, 70, 71, 87, 120, 122, 159, 161, 177, 178, 179, 180, 182, 183, 186, 189, 190, 191, 192, 193, 194

FoV

Field-of-view 51, 54

GLM

General Linear Model 52, 54, 191, 193

GM

Grey Matter 16, 20, 70

GPU

Graphical Processing Unit .... xxv, 29, 58, 59, 60, 88, 89, 160, 197

HCP

Human Connectome Project 49, 50, 52, 116, 119, 120, 121, 122, 123

HRF

Hemodynamic Response Function 15

ICA

Independent Component Analysis 2, 17, 22, 177, 195

ICVS

Life and Health Sciences Research Institute .... iii, 48, 49, 50, 52, 125

IT

Information Tecnology 7

LDA

Linear Discriminant Analysis.. 27

MEG

Magnetoencephalography 13

ML

Machine Learning 3, 4, 27, 41

MLNN

Multi-Layer Neural Network 30

MRI

Magnetic Resonance Imaging . xxiv, 1, 5, 9, 12, 13, 14, 15, 16, 19, 23, 49, 50, 51, 56, 63, 67, 68, 69, 70, 71, 72, 177, 178, 179, 180, 182, 183, 188, 189, 197, 198

MSE

Mean Squared Error 39

NIfTI

Neuroimaging Informatics Technology Initiative 20, 51, 178, 183

NIRS

Near-Infrared Spectroscopy 13

OECD

Organisation for Economic Co-operation and Development 13

PCA

Principal Component Analysis 17, 22, 27

PET

Positron Emission Tomography 1, 13, 23

RA

Rademacher Average 28

ReLU

Rectified Linear Unit 32, 34, 38

RF

Radio-Frequency 15

RNN

Recurrent Neural Network 30

ROI

Region of interest 2, 15, 17, 18, 21, 159, 178, 183

ROIs

Regions of interest 17, 18, 22, 48, 49, 73

RQ

Research questions 5, 6

RSNs

GLOSSARY

Artificial Neural Networks (ANN) are computational models inspired by the central nervous system, represented as a system of interconnected neurons that compute outputs from inputs when information is fed through the network.

Brain Atlas is a volumetric or surface based description of the geometry of the brain, where each anatomical coordinate is labelled according to some scheme.

Backpropagation is the procedure used in artificial neural networks to obtain error contribution of each network wherein the algorithm propagates the error since the output layer until the input layer.

Blood Oxygenation Level-Dependent (BOLD) is an imaging biomarker, measured by MRI that correlates the signals among different brain regions through the time.

Classification Model is a model that predict categorical class labels.

Cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters.

Connectology Functional is the integration of a set of biological concepts.

Connectome is a network representation of whole brain connectivity, comprising grey matter and axonal connections, which can be mapped by diffusion weighted MRI.

Convolution is the action of extract the features from the input image, such as the spatial relationship in pixels.

Convolutional Neural Networks (CNN) is a 2-dimensional spatial structure of neural network, one of the deep learning technologies.

Correlation Method is a statistical tool to evaluate how close two variables are.

CPU, or Central Processing Unit, is the electronic circuitry within a computer that handles all instructions that it receives from hardware and software running on the computer.

Cross-validation is a technique for estimating the performance of a predictive model.

Dataset is a collection of data.

Deep Learning is a subset of Machine Learning based in ANN that performs representations from the learning data.

Effective Connectivity is the union of structural and functional connectivity.

Feedforward Neural Network is an ANN whose information moves in only one direction - forward, - from the input to the output; there are no cycles or loops.

Fine-tune is a continuous search procedure for the best hyperparameters of a model.

fMRI is a technique that measures brain activity by detecting changes of blood flow among the different areas using the BOLD contrast, which relies on the fact that cerebral flood flow and neuronal activation are coupled, on other words, when an area of the brain is in use the blood flow to that region also increases.

Framework is a set of functions inside a system and how they interrelate.

Functional Connectivity is a statistical measure association among two or more anatomically distinct time-series.

GPU, or Graphical Processing Unit, is a programmable logic chip specialized for display functions.

Grey matter are unmyelinated neurons and other cells of the nervous central system.

Hyperparameter is a model parameter whose value is given to the model before the learning process.

Invasive Procedure is a technique that requires a skin incision or an insertion of an instrument into the body.

Machine Learning is an area of Artificial Intelligence that enables computers to learn without explicit programming.

Model Layer is the second basic element of a Deep Learning model that contains a set of neurons and a specific function.

Model Weights are the parameters that control the connection strengths between the layers.

Neuron Fire occurs when an action potential occurs, generated by voltage-gated ion channels in neural membrane, which allows the propagation of electric signal along the neuronal axon to another neuron through the synapse.

Neuronal Activity/Activation requires energy, which is provided almost exclusively by oxidation of glucose supplied via the blood circulation regulated by energetic demand bold.

Nipype Pipeline is a neuroimaging Python package, used to analyse data using different algorithms.

Parcellation is a subdivision of the brain structures into well-define parcels or regions of meaningful anatomical or functional significance.

Pooling Layer is used to reduce the spatial dimensions but not depth, in order to decrease the spatial information.

Pre-processing are the all the steps made required to raw data in order to get more reliable and cleaned data.

Python Module allows to logically organize the Python code by grouping related code into a module Python Package.

Regression Model is a statistical model used to estimate the relationship between dependent and independent (or predictors) variables, more specifically helps to understand how the typical value of the dependent variable changes when any one of the independent variable is varied.

Resting State Networks (RSN) is the brain activity when a subject is not performing an explicit task, observed though changes in BOLD signal measured by fMRI.

Time-series is a series of data points listed in time order.

UNIX is a family of computer operating system that start in 1970s at the Bell Labs research centre.

Unsupervised Learning is a deep learning method of inferring a function to describe hidden structure from "unlabelled" data.

Voxel, or a volumetric pixel, is a volume element, representing a value on a regular grid in 3-dimensional space.

White Matter are the matter areas of the central nervous system that are mainly made up of myelinated axons, also called tracts.

1. INTRODUCTION

Medical Imaging Informatics is a specific field of Medical Informatics (MI). As it is implied, it is responsible to deal with imaging data that has to be correctly and efficiently processed [1]. Later, the physicians started to use the processed data towards a best practice of medical care. Not only are these data extremely important for the understanding of the human being, but also in the search for new treatments and new ways of diagnostic. This type of research has evolved continually and that is why it has been the major booster in this area [2]. Medical imaging informatics is a simple collection of methodologies and techniques used in order to represent visually and spatially parts of the human body for diagnostic or research's purposes. The Medical Imaging is also a multidisciplinary area, specialty dedicated to the brain, known as neuroimaging. This area comprises several modalities such as Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET) and Computed Tomography (CT) [3].

Neuroimaging has a major clinical application as medical support in neurology, more properly in the diagnosis. But, recently, other areas have been recurring to neuroimaging, such as psychiatry, as a mean of a better understanding of psychiatric disorders. Also, a new dimension called the brain-body medicine merged, focusing in the study and understanding of the interactions between the brain, peripheral pathways and bodily organs. These new advances gives to neuroimaging research a new three new medical research topics: the mind-body connections, the behavioural psychosomatic and the integrative medicine [4]. Besides, the research and knowledge in neuroimaging is continuously changing over the time, we had the highest improvement in the last 20 years, [5].

In neuroimaging, the study of brain structure is not enough, since it doesn't give relevant information for the diagnosis of pathologies where structural alterations are not anatomically detected. Therefore, a method to assess brain function is necessary. This led to the development of functional Magnetic Resonance Imaging (fMRI), which is a technique that monitors hemodynamic events related to changes in neuronal activation in the brain, relying on the Blood Oxygenation Level-Dependent (BOLD) contrast [6];[7]. This modality has countless advantageous that boosts even more its growth, such as non-invasiveness, relative easiness of implementation and high spatial resolution. Moreover, the resulting signal is robust, it is easily reproducible and highly consistent. [5]. fMRI is typically used in the context of a given task, performed within the MRI equipment, in order to identify brain regions associated with the neuronal processes involved in the performance of that task. However, this precludes the interactions between different brain regions (i.e. brain connectivity).

Nowadays the neuroimaging community is changing emphasis from functional specialization towards functional integration [8]. Functional Connectivity (FC) is based on the temporal correlation between spatially remote neurophysiological events in the brain [9];[10]. FC can be measured using a variety of different techniques, but the most commonly used is the resting-state fMRI (rs-fMRI). Using this technique, several spatially distributed patterns were observed across subjects and presented a strong similarity between them [9];[10]. These patterns were denominated as the Resting State Networks (RSNs) [10]. FC can also be exploited from task-based fMRI. More recently, fMRI was also used to determine, what was named by Friston as, the effective connectivity [9]. This term defines the final objective, wherein the fMRI data is used to attempt to identify firstly the brain regions of interest that are mainly active during action, perception and cognition, and secondly the causal relations among activity in these regions.

There are three main methods used by the neuroimaging community to analyse and evaluate FC: Seed-Based Analysis (SBA), Independent Component Analysis (ICA) and connectomic analysis. All of these methods have the same final goal which is the analysis and attainment of brain connectivity networks, although having different approaches.

In the SBA method, the user identifies a Region-of-Interest (ROI) - or “seed”, - and gets the resulting networks from the correlation of time-series of the seed and of all the remaining voxels of the brain.

By contrast, in the ICA method it is not needed any reference region or seed in its application. This data-driven method decomposes the complete fMRI four dimensional dataset into time and associated spatial maps from components presented in the data [13]. The resulting components have the characteristic of being statistically independent. Additionally, ICA is also often used to support the decision of choosing the number of nodes to apply in the parcellation process for connectomic analysis.

Lastly, the connectomic analysis approaches the interactions between every possible pair of brain regions. This method uses the fMRI time-series extracted for each region belonging to a given atlas or for every voxel in the dataset. Then, the correlations between the time-series of all pair of regions are calculated. Thus, this results in a correlation for each pair, quantifying the FC between pairs of regions. The set of all regions and their connections can be visualized as a network or as a graph composed by nodes (brain regions) and the connections between them (FC). Additionally, connectomic data is typically represented as a connectivity matrix. The connectivity matrix is the schematization in matrix of all the values from the correlation between all pairs of regions. So, its size is N x N, where N is the number of regions or nodes resulting from the brain parcellation.

The information obtained in the connectivity matrices has shown a lot of applications. One of its use it is for the study of changings in the (brain functionality) due to a particular disease, such as

Alzheimer's Disease (AD) [14] or autism [15]. Another is for the study of the main networks in the connectome responsible for a specific psychological trait [16] or some mental capacity. Moreover, these matrices are used for classification recurring to Machine Learning (ML) or statistical methods. In *Machine-Learning To Characterise Neonatal functional connectivity in the Preterm Brain* [17], they have been able to discriminate preterm infants at term - equivalent age and healthy term – with born controls with 80% accuracy using a ML method. Plus, it was used to distinguish between healthy and depressed controls with accuracy values of 85.85% and 70,75% in classification between patients with treatment resistant depression, non-treatment resistant depression, and healthy controls [18]. Furthermore, this connectivity matrices have shown to be reliable fingerprints, making possible to identify accurately a specific subject from a large group. The research made in *Functional Connectome Fingerprinting: Identifying Individuals Using Patterns Of Brain Connectivity* [19] demonstrated that applying correlation methods can predict and identify an individual with 92,5% and 94,4% accuracy in resting states. For fMRI acquisition in task conditions, they also got results that presented a great accuracy, rates about 87,3%. Relatively to classifications using train and test data with different fMRI conditions (resting or task) for each one, the final accuracy values achieved were lower, in maximum rates of 50,4%.

The approach to FC in this dissertation is based in techniques of ML, a recent subfield of computer science related to neuroimaging. The main advantage of this technique is the capacity of the models "learning" from data. So, there is always a learning process in order to obtain accurate representation of data and prior knowledge [20]. In neuroimaging, as supporting tools to studies with FC, these techniques come firstly to solve the need for classification models, as referred previously in some examples. But the development and implementation are yet little and insufficient, when exists so much to explore and discover about brain functionality. Besides that, the methods applied are simple and applied in a similar way as has been used in the last years. Nevertheless, a new area recently emerged in ML - the Deep Learning (DL).

DL is nothing more than an improvement of Artificial Neural Networks (ANN), which allows to modulate higher level of data structure abstraction and improve predictions (provides a better classification) [21]. It is only possible due to its characteristics, larger number of layers and more accurate algorithms, which makes easier to find similarities in data and improving the process of "learning". That allows to work with high dimensional datasets. These are the reasons that give an enormous potential to the use of Deep Learning in this work. Since the connectivity values are saved in large dimensions connectivity matrices completed with a lot of features, they allow these features' granularity to be diversified, where the unit in study can be a voxel or a brain region (resulting from brain parcellation) [22];[23].

In addition, DL is a technique that, although being little used in neuroimaging in general, in image processing has expanded quickly and granted the creation of a lot of applications such as object recognition. Even some prodigious technologies companies, like Facebook and Google, are applying DL and have made a great effort towards its development. So, it is a good opportunity to use this information and algorithms, and adapt them to work with FC to create good classification models.

1.1. Motivation

The brain is a complex system and there is a lot to discover and understand about it yet. The brain structure is already well documented, since it has been a trend area of research and there is an enormous quantity of knowledge associated to it, in contrast to brain function. To fix this gap, the field of neuroimaging has been in a continuous searching for classification models using FC. This classification can be, e.g., to classify and distinguish a subject as healthy or non-healthy. Additionally, recent researches have also demonstrated that FC can be used as a "fingerprint" classifier. So, it can be quickly evidenced an increasing demand regarding FC data due to its conceivable applications.

Although in the present it is verified that there are already some ML techniques applied in this field, they are insufficient, and, at the same point, they are simple comparatively to the evolution occurred in the field. Thus, besides of the potential knowledge that FC can give is tremendous, the use of data is very far from what can be achieved. To resolute these opportunities and problems, it is proposed a new approach with a DL framework. This ML method has already achieved good results in fields like image processing, and it is viewed as an incoming and promising "learning" technique. This method has the advantage of working with high dimensional and big data and how much bigger it is, better will be the results. It also presents a great capacity of "learning" and no theoretical limitations. These are the characteristics that make this technique a right tool to work with FC data.

1.2. Objectives

The work begins by identifying the main goal, or more accurately, the research problem, traduced in a question – "*Can the functional connectivity from rs-fMRI be used as feature in subject's classification tasks using a Deep Learning approach?*". In other words, the artefact presented is a DL approach in order to get knowledge using functional connectivity matrices processed from resting-state fMRI (rs-fMRI), to create prediction models to classification tasks. These models are based in individual features for classification of subjects or group features in sex and age classification. As result and to answer the research problem, emerged two secondary objectives: one linked to the MRI and rs-fMRI pre-processing processes to get the FC and another that involved the creation and development of Deep Learning models. During the model development there were continuous adjustment and fine-tune of DL parameters models, finishing with evaluation of the results. To specify each of these objectives, it were created others more specific and a set of research questions (RQ) that emerged during the work elaboration and should be answered. Thus, it was elaborated the next work plan:

1. *Development of a framework that allows the application of DL algorithms to FC data*
   - Study and implementation of the best fMRI processing in order to extract FC

     RQ 1 - Can an implementation of a *Nipype* pipeline improve several points of a processing workflow?

     1.1. And with differences in the time and data organization?

     1.2. What about the data management?

     1.3. And using multi-processing?

     1.4. How to stop failures from affecting the working pipeline?

     1.5. And with integration of several image software?

     RQ 2 – What approach should be used to extract the different types of functional connectivity?

     RQ 3 – Which is the best way and which data to save using a Python environment?

     2.1. How much processing is needed to FC data?

     2.2. And which data will be saved?

   - Development and application of a framework to deal with DL algorithms using FC data already treated before, as intended

     RQ 4 – What is the best architecture to implement in order of fine-tunnelling the DL models and its posterior creation?

     3.1 Which Deep Learning Python libraries are more appropriate to use?

     3.2 Which metrics to use and to compare the models?

     3.3 Which metrics are extracted in the final analysis of the models?

3.4 Was there any limitation in the validation approach used?

2. *Test the framework with real data and evaluate the results*
   - Test and analyse if the FC static and dynamic acquired in different sessions can be used as individual "fingerprint" or group classification by gender.

RQ 4 - Can the results of the article "*Functional connectome fingerprinting: identifying individuals using patterns of brain connectivity*" [19] be reproducible?

4.1. Can be the data acquired in the same research project or other data 'with no such good' quality of resolution acquired in machines with magnetic fields less than 3T reproducible?

4.2. Using the same atlas or another atlas?

4.2.1. Comparison of the results and analyse how the number of nodes influences the performance.

RQ 5 - Is it possible to get better results using the DL approach by the framework created and static FC?

5.1. Can DL increase the performance results?

5.2. Are there any significant differences using different atlas? And how that happens?

5.2.1. Comparison of the results with the ones obtained in 4.2.1.

5.3. Can convolutional networks improve the models?

5.4. Does the data normalization change the results?

5.5. Are there some subjects classified more correctly than others?

5.5.1. Are there any differences in the data?

RQ 6 – Can the previous approach and dynamic FC improve the last results?

6.1. Can DL increase the performance results?

6.2. Are there any significant differences using different atlas? And how that happens?

6.2.1. Comparison of the results with the ones obtained in 4.2.1.

6.3. Can convolutional networks improve the models?

6.4. Does the data normalization change the results?

6.5. Are there some subjects classified more correctly than others?

6.5.1. Are there any differences in the data?

6.6. Can the type of FC change the results? If so, how and why it happens?

RQ 7 – Can the DL approach give good results in the age classification task?

1.3. Research Methodology

Firstly, it is necessary to define correctly the term "research", often mistaken with a simple information gathering from documents or facts, or a merely rummaging for information [24]. Research can be defined as a systematic process of collecting, analysing, and interpreting data in order to give more knowledge in a particular phenomenon [24]. Before anything in the research process, it is necessary to define the research problem. This can be described as the main and comprehensive difficulty that the researcher wants to answer with the proposed work [25]. Then comes the construction of the design's research. This fundamental process where is defined the conceptual structure within will guide the research and wherein is understood the way that the gathering, measurement and analysis are made [25]. In this context appears other term, the "methodology", and summarized it means the mode to systematically answer the research problem using research techniques. This is a general approach, but since the appearance of informatics, it was needed to adapt the existing research methodologies more appropriately for the Information Technology (IT) areas. One of them was the Design Science Research (DSR), methodology that will be followed in the dissertation, once this work covered areas, such as computer science and medical informatics that belong to Information Systems (IS).

DSR is a recent methodology in IT area and, since its appearance, it is continuously gaining attention in the scientific community. Mostly because of its characteristics to resolve some lacks and problems generated by the new technologies that others before couldn't solve [26]. The DSR methodology is built around the artefact term. Object which will be constructed and designed, effectively and efficiently during the methodology with utility in an application environment [27]. The artefact can be constructions, models, frameworks, architectures, methods and instantiations [26]. For the research to be in accordance with this methodology, it should follow the guidelines of the same, known as DSR Process (DSRP), and it resulted from work and investigation of several researchers. Furthermore, it can be traduced in six main activities [28]:

i. **Problem identification and motivation** – The research problem is defined, and the practical and scientific values of the solution presented are justified.
ii. **Objectives** – The qualitative and quantitative objectives inferred from the problem are deduced. In this section it is required a construction of the state of art wherein is defined the state of the problems and of the existing solutions.
iii. **Design and development** – The functionality and architecture of the artefact are defined, ending in its creation.

iv. **Demonstration** – The confirmation that the artefact solves the problem efficiently.

v. **Evaluation** – The results are registered, measured and analysed to conclude if the artefact is really a solution for the problem.

vi. **Communication** – It is the final process, where are communicated the problem, the solution represented by artefact and its results.

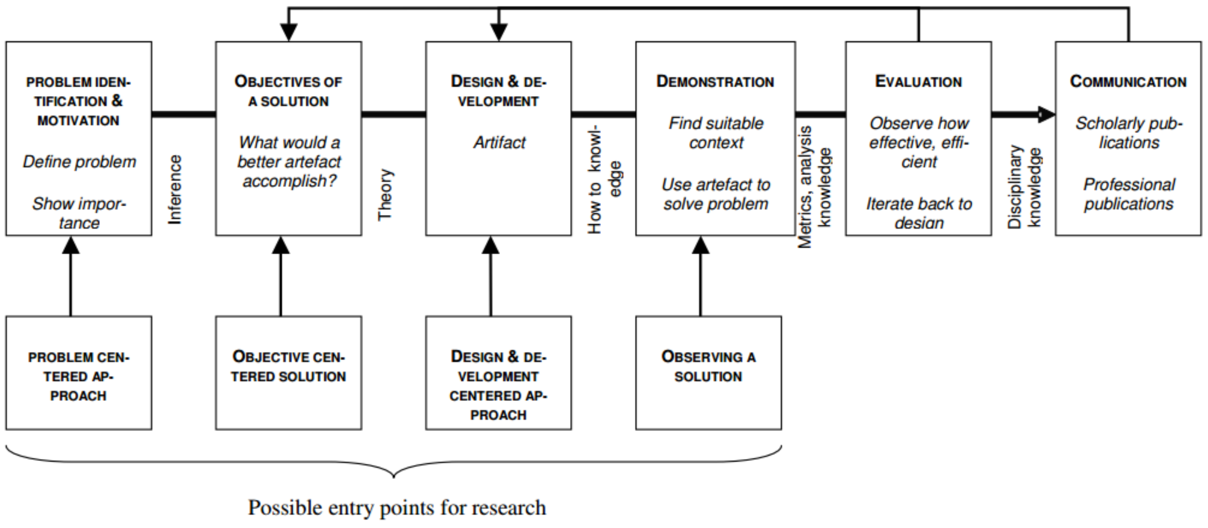

PROBLEM IDENTIFICATION & MOTIVATION
Define problem
Show importance
Inference
OBJECTIVES OF A SOLUTION
What would a better artefact accomplish?
Theory
DESIGN & DEVELOPMENT
Artifact
How to knowledge
DEMONSTRATION
Find suitable context
Use artefact to solve problem
Metrics, analysis knowledge
EVALUATION
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
COMMUNICATION
Scholarly publications
Professional publications
PROBLEM CENTERED APPROACH
OBJECTIVE CENTERED SOLUTION
DESIGN & DEVELOPMENT CENTERED APPROACH
OBSERVING A SOLUTION
Possible entry points for research

Figure 1.1. Schema summarized with the task, resources for each phase of the Design *Science Research Process (DSRP) (from [28]).*

A key point is that the different DSRP parts are not fixed and, so they do not have to follow a restricted sequential order. Subsequently, the start point can be in almost any step forward or backward. In the Figure 1.1 are schematized the different phases, its description, the resources used in each one and possible movements of continuous research to find the best artefact for the defined problem.

1.4. Structure of The Document

The present work comprises, besides this introductory chapter, more six different chapters structured as follows:

Functional Connectivity in fMRI - This chapter resumes and introduces all the important concepts related to Functional Connectivity extracted from fMRI, from the methods, the techniques, the technologies involved and applications.

Deep Learning – Structured as the first chapter about the technique applied in this work**,** it introduces some essential concepts to understand more its operation, its limitations and how fine-tune the possible parameters.

Materials – In this third chapter it is resumed all the important information about the datasets, the technology, the software and other data applied and used in the next chapter, and the methods to get the intended results.

Methods – The chapter four has all the information about the work done in this dissertation to get the results. So, it is explained all the study's architecture, wherein it is presented the pre-processing to fMRI and MRI to get the data used by Deep Learning models and, then all the search methods for the best models with the statistics analysis and models' management resulting from it.

Results and discussion – Over this chapter are exemplified the different results obtained during the research for the best models and the replication of results of other articles. During the process of continuous search for the best model are argued the results obtained and proposed new actions in order to improve the model performance.

Conclusion – In this last chapter it is synthetize all the results accomplish with implications for the main research problem and the conclusions that could be extracted. Also, the research questions done in the elaboration of this dissertation are answered and potential future work to be done is described.

2. Functional Connectivity in fMRI

The Functional Connectivity (FC) is a consequence of the continuous search to understand the human brain. This is a complex organ, responsible for our conscience, personality, sensations and cognitive process and, yet much unknown, reason conjecture that fascinates us as humans. So, it was needed tools and technologies to study the brain connectivity. Thus, it was created a group of techniques, anatomical and functional (Figure 2.1). To study with less risks for human health and in an easier way, it were created techniques less invasive. The more invasive and with more spatial and temporal sensitivity are only applied in animals' models (see Figure 2.1).

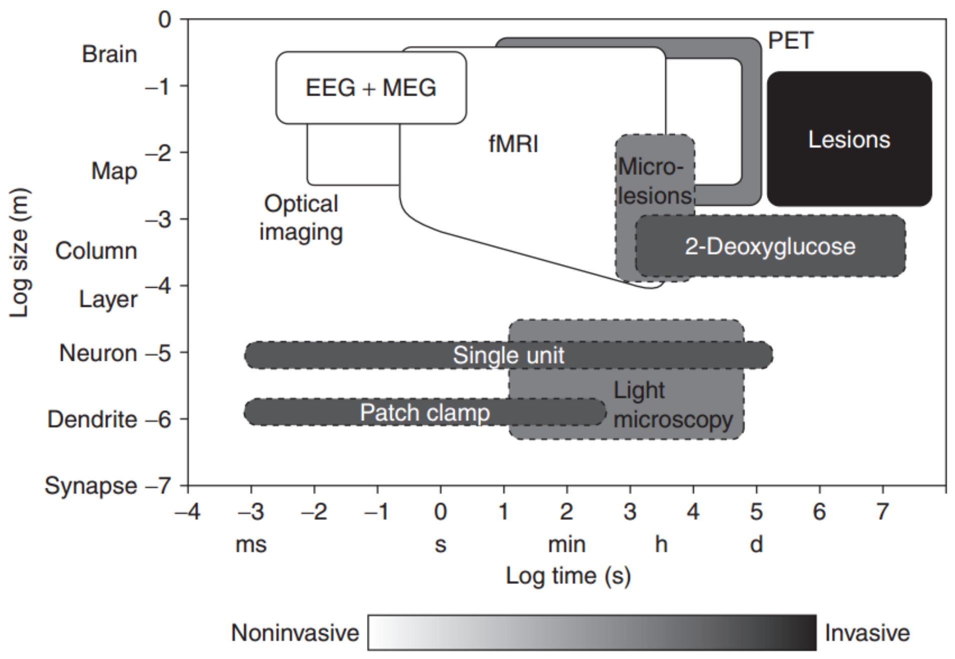


Figure 2.1. The different brain imaging techniques with the relative spatial and temporal sensitivities, and the level of invasiveness of each one (from [29]). The techniques only used in the animals are outlined with a dash line.

The anatomical techniques appear in general first and their use revealed to be insufficient to a deeper analysis. But they were important since they proceed more neurological sophisticated methods. Besides, they maintain to be consequential for research, such as: Magnetic Resonance Imaging (MRI) to give detailed structural information to distinguish the white matter from the grey matter and, Diffusion Tensor Imaging (DTI) to visualize myelinated tracts. Although, the dominant are the functional methods as well as the study of the brain connectivity - the study of the neural activity when a subject performs a specific cognitive task or only the study the normal activity in a subject in a rest state. Moreover, the functional techniques can be categorized by the measuring type, direct or indirect. Inside the direct methods are the Electroencephalography (EEG) and Magnetoencephalography (MEG), and they measure directly the electrical activation associated to neuronal activity. Then, the indirect methods that measure the neuronal activity by the metabolic activity and the blood flow, include the Near-Infrared Spectroscopy (NIRS), the Positron Emission Tomography (PET) and the functional Magnetic Resonance Imaging (fMRI).

The fMRI, used in this work, is the main modality used in clinical neuroimaging since its introduction in the clinical environment and, nowadays, has been become a major tool to research the structure and function of the human brain.

The MRI is based in the physical property *nuclear magnetic resonance*, which measures the protons of hydrogen and atoms of the water molecule. Water is the most abundant element in the human body, and even more in the soft tissues as the brain. During the procedure is created a strong external magnetic field and the proton aligns according to it. Then, it is used radiofrequency pulse to excite the protons and alter the magnetization alignment comparatively to the magnetic field. These changes are quantized by sensors, being possible to create a patient's image [30];[31].

The MRI technology continues in advance. The magnetic field strength used for research has increased over the last years due the improvements in the magnet design and technology. Also, the radiofrequency electronics and magnetic field gradients have improved supporting the use of systems with higher field strength. Nowadays, the 3T fields are already plenty used in the neuroimaging research centres, and institutions are taking improvements to machines with 7T or even higher fields, such as 9.4T, 10.5T, 11.7T and 14T [32]. The consequences are only positive for research in neuroimaging due to a better resolution, sensitivity and contrast in the acquired volumes. But low fields at or below 1.5T continues to be the main fraction worldwide, once the application purpose is merely clinic and it is not necessary a high resolution with more costs.

In 2016, OECD in the United States, was the first company in the worldwide to collect about 121 MRI scans per 1000 of population [33]. They showed that this data has an enormous potential to be study without negative biological effects reported. Recently, even the cardiac pacemakers, cerebral aneurysm and cardio-ventricular-defibrillators have been developed to be MRI-safe following strict protocols. But they are only allowed to submit magnetic fields inferior to 1.5T and it is always needed some care about the implant heating or the ferromagnetic forces. Although this widespread sense of security, there are some researcher concerned about the possible direct and indirect effects over the DNA resulting in genetic damage and in turning carcinogenesis as consequence of the Static Magnetic Field

(SMF). A shared concern due to the continued increase of SMF. Some studies already detected DNA damage as consequence of MRI, however more studies are needed [34].

The fMRI appears to fill the lack of non-invasive methods to study the brain activity and, due to some positive reasons, it became a widely technique in brain connectivity study. First of all, because it has the best spatial and temporal resolutions compared with the others indirect methods. In addition, MRI scanners are widely used, the costs are relatively low per scan comparatively to another methods, and has not scientifically recognized risk for patients. Ogawa was the first to describe Blood Oxygenation Level-Dependent (BOLD) parameter as result of the hemodynamic changes in the brain caused by neuronal activity [7].

The discovery of the possibility of using the fMRI for the study of the functional connectivity, more specifically in how the different regions are related and the identification of the different networks formed by these relations, associated to brain senses, cognitive processes or simple to resting state, opened new fields to explore. What led to a “boom” of the connectivity studies in the last decades, consequence of the large search of it to clinical use and neurologic research, being the neuropsychological research with the major application in this area.

2.1. Functional Magnetic Resonance Imaging

The fMRI is a variation of Resonance Magnetic Imaging (RMI) that appeared to fill the lack of a method to study the brain activity, becoming the most important technique to serve that purpose. This non-invasive technique measures the hemodynamic change in the blood as consequence of neuronal activity known as vascular hemodynamic response. So, the neuronal activity is related to an increase in the Cerebral Blood Volume (CBV), in the Cerebral Blood Flow (CBF) and in the Cerebral Metabolic Rate of Oxygen (CMRO2). The increase of CBV and CBF explains the vascular response which is positively correlated. Moreover, the increasing of the oxygen rate consumption (CMRO2) in the tissue around the fired neuronal cells leads to decrease the venous oxygenation level. In order to revert this effect, the CBF has to increase but quickly occurs an overcompensation due to CBF exceeds the CMRO2. In another words, it is provided more oxygen than it is needed, and this is demonstrated as a peak in the hemodynamic response function (Figure 2.2). Ogawa in the 90’s discover that the BOLD, as contrast technique, can measure indirectly this phenomena: the CBV, the CBF and CMRO2, as results of the neuronal activity [7]. The haemoglobin (Hb) presented in the blood carries the oxygen molecule, and when the oxygen molecule is out of Hb is it called desoxyHaemoglobin (dHb). Due to the vascular response during the neuronal activity in a tissue there is an oversupply of blood and, consequently, the decrease of quantity of dHB. As dHB presents magnetic properties (is paramagnetic), it changes the static magnetic field in T2-weighted images. It can be used as contrast agent and, thus, be detected and measured by MRI. The BOLD signal is characterized by the same behaviour of the estimated Hemodynamic Response Function (HRF), consequence of its vascular nature. So, after the neuronal activity has a rise of the signal until the maximum value, a peak of ≈ 5 to 6 seconds after the stimulus. Then it returns to the baseline after ≈ 12 seconds, where the stimulus is followed by a negative overshoot before the final signal stabilization about 25 to 30 seconds after the stimulus (Figure 2.2). The delay occurred between the stimulus and the neuronal activity is caused by the slow nature of the hemodynamic response as seen in a HRF, explicating the low sampling rate around 1 Hz [29];[35];[36];[37];[38].

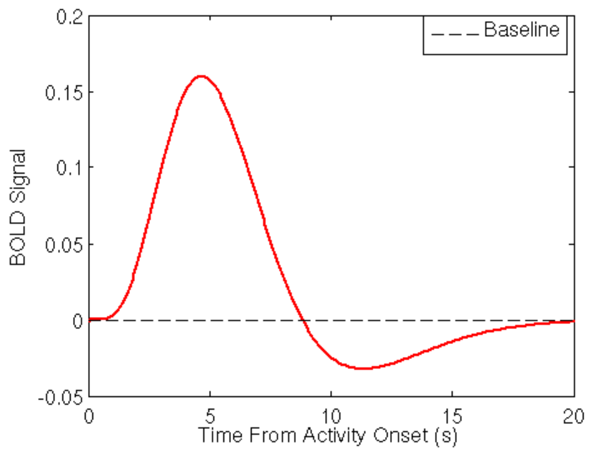


Figure 2.2. Model of Hemodynamic Response Function (HRF) (from [39]).

In the acquisition process are obtained, in a fast sequence, several images with low resolution of the ROI. Each of these images has a volume of the brain in study and are extracted within 2 to 3 seconds, because there is a need to acquire the data in reduced amount of time, leading the Echo Planar Imaging (EPI) to be more used in this type of exams.

The EPI is the fastest practical imaging method and it was proposed in 1977 by Mansfield [40] applied to nuclear MRI. The difference between MRI method and the EPI method resides in the conventional method of creating data for an image from a series of discrete signal samples rather than form a complete volume from a single data sample/shot. This volume is extracted in the interval of time between two Radio-Frequency (RF) excitation pulses, interval known as repetition time (TR). During the signal extraction the TR is used as window and centred in the echo time (TE) in order to maximize the signal to extract. The TE is an acquisition parameter and depends on the target tissue, the target imaging parameter and the magnetic field strength of the MRI machine. In the case of fMRI as the T2* weighted image it is used to measure the BOLD fluctuations, the TE is approximated as possible to the T2* relaxation time of Grey Matter (GM). As the major advantage, the EPI method is way faster than the MRI method: for a normal acquisition with TR=3 second, the first one can collect all the image data in about 40 to 150 milliseconds, while the second, for an image of the same resolution, needs 384 seconds. After the acquisition of the volume's set in a specific interval of time, it is built a time series for each voxel using those volumes, resulting in the final fMRI's signal.

Besides the fMRI technology (e.g. multichannel receiver) and hardware (e.g., higher magnetic strength fields), there is a continuous search for a greater Signal-to-Noise Ratio (SNR). This rate measures the quality of the acquisition being directly proportional. This because the fMRI time-series contains not only the signal of interest, but also other signals as noise (e.g., cardiac function and respiration, subject head motion, thermal noise or hardware interferences) [41]. This is the reason of the need for pre-processing pipelines before any analysis and use. Also, it was created the Multi-Echo (ME) acquisition of fMRI signal to improve the sensibility to BOLD signal and increase the SNR. This method will improve the single echo techniques as EPI, with some alterations. Besides the methods of EPI, the ME measures the time-series for each voxel using different TE, and so the signal differs between each voxel in terms of T2*weighting and thermal noise [42];[43];[44].

A normal study of fMRI has as objective the identification of regions that are activated when a subject receives a stimulus or does a specific task. In these conditions, it is said that is a task-based fMRI. But the study of functional connectivity is gaining importance in the neuroimaging community, extracted specially from fMRI done in resting state, without any stimulus or task. But it is also possible to use task-based for functional connectivity and, in the future, this could be a very promisor approach, moved by the continuous investigations and advances in understanding the different patterns in rs-fMRI, increasing more the knowledge about FC.

2.2. Brain Functional Connectivity

The FC can be defined as "temporal correlations between spatially remote neurophysiological events" [9];[45]. So, it can be simple described as the manner how occurs the communication between brain regions that are anatomically distant.

This is a recent applicability for fMRI discovered in a resting state condition when was concluded that some spontaneous and intrinsic fluctuations occur in the BOLD signal [10];[46]. So, the brain is never idle, and even in rest, there are anatomic separated brain areas that besides the presence of spontaneous neuronal activity, they are linked to others. In addition, the regions where the signal had a high correlation with other region, they had functional similarities already described and known [10];[11];[47]. Moreover, functional studies using rs-fMRI demonstrated the existence of temporally coherent patterns known as the Resting State Networks (RSNs), associated to low frequencies. But other networks with coherent BOLD fluctuations were found associated to cognitive processes, emotions or senses. Although the functional connectivity can also be studied in a task-based fMRI, the large number of variables in this type of acquisition make this too uncertain to use, besides being more complex to design the task conditions, the imaging acquisition and the analysis process [46];[48].

In the present there are some statistical models to analyse rs-fMRI in order to extract FC information. The Independent Component Analysis (ICA), the Principal Component Analysis (PCA), the clustering, and the Seed-Based Analysis (SBA) are the four most popular applied to fMRI, being the first three data-driven models[1].

Beginning with SBA, as the name enunciate, is a statistical model where the first step is the identification by the user of a specific ROI. Then it is created a map of the networks as a result of the computation of the correlation of the signal of this region and the other voxels that remain outside the seed [46]. Regarding the PCA technique, it relies on discovering a group of orthogonal axes that can maximize the interpreted variance of the data and divide the pertinent data from the noise. On the other hand, the ICA technique is an extension of the PCA technique, in which there is a separation of the individual elements into their components. It also models the obtained dataset as a constant amount of spatially/temporally independent components. Finally, the clustering method is based on mathematical algorithms that associates data into clusters, in which the specifications within a cluster are like one another than to different clusters [13];[38].

There is yet the ROI analysis, option used in Machine Learning studies, that is performed based on the SBA. In this method there is *a priori* the selection of ROIs in the brain, and then, for each region, it is determined a value that is correlated with all the other regions. Thus, the result is an association matrix that has the values of the correlation of all ROIs' pairs. The ROIs' structure information and parameters are normally extracted from atlases, other functional studies or statistical models applications, such as ICA [49] or clustering [50]. The fact that in this analysis is usually used the same ROIs for all the acquisitions of different subjects, does not take into account the subjects' functional brain variability. Consequently, it can happen that specific ROIs don't represent correctly the signal of a brain

[1] Data-driven models are models that do not require previous information nor a predefined model.

region, adding some error or loss of information to future use. Thus, it is important to create ROIs that are reproducibly and the number of regions chosen maximize the regions time-series extracted [50]. Afterwards, this type of analysis can also be used by graph theory analysis, where the brain is modelled as a network composed by nodes (ROIs or voxels) and edges (values of time-series correlations). This representation of the brain, as networks that can also be modelled as a graph, can be used in mathematical calculations and graph properties to perform the FC analysis and discover significant patterns [38]. Regarding the FC calculated by the ROI approach, there is a new variant called High-Order Functional Connectivity (HOFC), where the correlation of each region with all other regions is called topographical profile. And using the correlation matrix calculated to the ROI analysis, it does a second correlation between a topographical profile with each of the remaining profiles, resulting in a matrix with the same size. But the difference in this method is dealing with spatial properties presented in BOLD signal, instead of focusing in the temporal correlations, like other methods [51].

However, more recently, emerged a new approach to study the FC. Once a resting state analysis is obtained, the correlations measure over a lengthy periods the fluctuations related to complex, and dynamic interactions patterns that happen in finer time intervals are lost [51];[52];[53];[54];[55]. So, dynamic FC could be a good approach to study and understand the different spontaneous changes in the patterns presented in the rs-fMRI time-series. There are nowadays several papers with different approaches to measure the different variations in the spatiotemporal structure of the fMRI time-series. One of the most common is the sliding window strategy where there is a partition of the whole BOLD signals into different intersecting segments, for a specific window size [51];[54];[56];[57];[58]. But this technique has some issues due to the analysis being done using a temporal window and its limited size, consequently affecting the temporal results and statistical validation of the results [55]. Nevertheless, there is another techniques already applied in dynamic functional connectivity's studies such as: time-frequency analysis [59], single-volume co-activation patterns [60], repeating sequences of BOLD activity [61], multiplication of temporal derivates [62], and phase coherence connectivity also named phase synchronization [63];[64];[65]. There is also another approach where the dynamic functional connectivity is evaluated among subsets named domains. The method was named as dynamic Functional Domain Connectivity (dFDC) and uses the RSNs domains. Thus, the dynamic changes are only considerate for a subset specifically to study, instead of using whole-brain [66].

2.3. From fMRI to Functional Connectivity

2.3.1. Acquisition

MRI is a masterpiece of engineering, physics and informatics that since its creations - in the 1970's - had a great expansion as an imaging method. Until now it comes as one of the most important radiological examinations. The fMRI acquisition is a variation of the MRI, and so the fMRI time-series are also extracted in the same machine. Although existing some variation according to the subject state analysis, task or resting state, the procedures used in the acquisition are very similar to different functional studies. But, in both, the acquisition is acquired in an interval time that will define the number of volumes extracted per acquisition.

The rs-fMRI is recommended to be made with the minimum stimulus possible for the subject. Even though is not consensual, the best approach to take the acquisition is in the resting state, where normally it is required to the subject to stand relaxed, to remain wake and try not to think in anything. But even so, some fluctuations are registered caused by the subjects' behaviour, such as the visual processing, and that is why it is also required to the subjects to close their eyes or to fix them in a crosshair.

On the other hand, there is stimulus in the task-related fMRI during the acquisition in order to acquire the respective BOLD responses. The temporal allocation of these stimulus is called paradigm, and the design of the different paradigms will bear on the study's precision and effectiveness. The task-related fMRI uses two different paradigm designs: the block design and the event-related design. The block design uses blocks with equal temporal range, repeated activity and rest to excel the neuronal activation. In the event-related design, the stimulus is presented with different temporal ranges, and with one or more specific cognitive events in test. Therefore, the block design is used to discover the response to a stimulus as patterns activations, and event-related discovers the characteristics and properties of the response to a stimulus [67].

Additionally, during the acquisition procedure, in both cases, the patient should be informed to be as still as possible. The motion, or more properly the head motion, adds a lot of noise and artefacts in the acquisition, decreasing the data quality. So, it is important during the acquisition to prevent from happening head or body motion, taking some measures as head fixation or using a comfortable padding for head and body.

2.3.2. Pre-processing

The pre-processing is a fundamental step in any area that work with signals or images to improve the data quality, as well as remove possible noise to later be conceivable to extract valid information.

The fMRI acquisition it is not perfect and there are a lot of interferences due to a substantial number of variables to control. Thus, to the normal fMRI signal are added some unwanted fluctuations, consequence of hardware-related artefacts or physiological events from the participants. The ones caused by the hardware are specially interferences in sensors, thermal noise normal of electronic components and hardware instabilities [41]. Although in the acquisition configuration are taken some precautions to diminish these problems, they continue to have a great impact in the quality of the final time-series. The artefacts related to participants are above all resulting of the head motion, but also from the motion caused by the cardiac function and respiration and consequent vascular effects. They also have a negative outcome in the acquired signal [41];[68];[69]. This is also proves that low-frequency drifts (artefact) are responsible for 8,4% of the signal variation, being the visual cortex the area most affected when compared to whole-brain Grey Matter (GM) [70]. Therefore, if nothing is done, the artefacts will definitely reduce the sensibility in catching the true neuronal activity. Therefore, it causes low reproducibly of the results in re-tests, increases the difficulty, and adds errors in data interpretability what impair the scientific or clinic value of the studies. And that is why the data needs to be pre-processed, to eliminate or diminish the effect of noise and increase the rate between the signal and the noise – SNR -, growing the quality of the data. Nonetheless, before the pre-processing itself, there are some processes that should be done.

The verification of the quality of the data before the pre-processing is crucial. The verification, if exists corrupted data or if the protocol was respected for all the subjects in a group analysis (e.g., acquisition time, number of slices, repetition time, and others), is a good start. Then, it confers if the brain volumes don't have any artefact or a brain lesion, using available Digital Imaging and Communications in Medicine (DICOM) viewers like Osirix, RadiAnt, MRIcon or ImageJ. The viewers have to be compatible with the format DICOM because this is the format of the data resulting from the machine acquisition; DICOM is a format used to manage and view medical images [71]. Nevertheless, the pre-processing tools work mainly with the NIfTI (Neuroimaging Informatics Technology Initiative) format, an adaptation of Analyze 7.5 (uses the header to save metadata) [72]. So, the conversion is a must-have step before the pre-processing. Most of the pre-processing packages already have converters tools but there are also others upright dedicated converters. The most used are the dcm2nii and the MRIconvert (see more in Appendix A).

A normal pre-processing fMRI pipeline follows the next steps (see more detailed description, with a rs-fMRI pre-processing example, in Appendix A):

1. **Initial stabilization** – Removal of the first volumes of the acquisition in order to allow the signal stabilization;
2. **Slice-timing correction** – Once the volumes are acquired with a set of slices (2D images) and, although the repetition time is small, there is a minor delay between the real and the expected acquisition times. So, this step has in account all the slices of the volume, interpolating each voxel time-course of each slice with a reference slice, normally the first or the most near with the half of the TR;
3. **Motion correction** – As already mentioned, the head motion is a significant artefact to remove. There are different approaches, but the most popular is realign each volume using the rigid body alignment transformation using a reference, e.g., first volume, last volume or the mean volume;
4. **Skull stripping** – Removal of the skull maintaining only the ROI, the brain;
5. **Spatial transformations** – The volumes of different subjects are spatially different (variability of size, shapes and orientation), so this method aligns the different images originated from a subject with others from a different modality or subject (registration), or from common standard space (normalization). In the application of these methods are used templates of standard space, in which the most applied is the Montreal Neurological Institute (MNI) template (preferably the MNI152);
6. **Spatial smoothing** – In this step each voxel in the volume is averaged with the neighbours, working like a band-pass filter where the low frequencies are inhibited, and high frequencies maximized. The normal procedure is the convolution of the time-series with a Gaussian function with a specific width according to the data application;
7. **Temporal filtering** – Removal of cofounding signals with specific frequencies caused by the several artefacts, and normally belonging to low frequencies or to high frequencies ranges when compared to frequencies of a normal signal of fMRI. Thus, in rs-fMRI is common the use of a band-pass filter for the interval 0.01 to 0.08 Hz. Although, in task-based it is only used a high band-pass to eliminate noise of low frequency due to the signals of interest belonging to more frequency ranges.

2.3.3. Functional Connectivity as Static and Dynamic Connectivity Matrix

This approach of analysis and extraction of functional connectivity from the time-series signal is common used for applications of Machine Learning or statistics applications, and, as was already mentioned, the regions of interest analysis. As time-series are for each voxel, and normally the ROIs are regions of the brain, it's made the brain parcellation. Although, if the approach is directed to voxel analysis, there is no needed to perform this step.

The brain parcellation is the division of spatial domain into a set of non-overlapping nodes or regions. These regions are constituted by a set of voxels that show somehow a homogenous behaviour or properties in one or various modalities (e.g., functional connectivity, anatomical connectivity, task-related activation). Thus, the different voxel time-series signals in the same region are average, and so, the resulting data becomes in a set of time-series values wherein the set size is equal to the number of regions chosen. The most common approaches of brain parcellation to study the functional connectivity are: the use of specific anatomical or functional regions (ROIs), the use of brain atlas of recognized studies, and the use data-driven statistical methods. The analysis of functional connectivity based in specific ROIs aims to focus the analysis in these areas [73], and usually the regions' selection is based on previous studies and tests [74]. The brain atlases have a well-defined set of ROIs of whole-brain volume and their spatial structure is based on the anatomy, functionality or connectivity of the brain. These atlases are available online to use, such as AAL atlas [75], the Harvard-Oxford atlas [76] or, more recently, a 268 nodes atlas [50]. This approach is good to compare and reproduce studies, once the study is based on the same structures, though the labels could not maximize the resulting signals and lost some valuable information. That is the reason it is better to use the data-driven techniques to find the best model that adjusts to the data in study. The number of methods to use are larger and they could be mixed. It also includes variants of k-means algorithms [77];[78], variants of clustering [79];[80] (e.g., hierarchical clustering, spectral clustering and dense clustering), Independent Component Analysis (ICA) [81];[82], and variants of Principal Components Analysis (PCA) [83].

After the parcellation process, it is possible to construct the connectivity matrix. It can be of two different types of connectivity matrix: the static and the dynamic. Both are a mathematical/statistical measure between the different regions pair-wise of the data, resulting in a symmetric matrix with the number of columns and lines equal to number of regions. The differences are in the measurement and in the number of final matrices. The static uses correlation measurement between the different regions and results in a correlation matrix. On the other hand, the dynamic compares the temporal variations and properties of each pair wise regions time-courses signals, in an instant or interval of time, as already described in the brain functional connectivity section. Thus, it results in several dynamic connectivity matrices, where the number depends on the temporal analysis chosen. If the measurement is done for each repetition time, e.g. each volume, the number of matrices is equal to the number of TRs of the acquisition.

2.4. Neuroimaging Software

In the Medical Imaging informatics, more precisely in neuroimaging, it is available some software for analysis and processing of fMRI images, and other types of images, such as images from MRI and PET, to whom exist three software that are more known: SPM, Freesurfer and FSL.

The Statistical Parametric Mapping (SPM) software is an open source software designed to functional and structural analysis of images of neuroimaging. This program is based in spatially statistical methods in order to test possible hypothesis about regionally specific effects. So, all the process proceeds by analysing each voxel using any statistical parametric mapping. SPM was originally conceived by Karl Friston, but it has been suffering updates over the years, wherein it were added new tools and features, and all the versions are implemented in Matlab [84];[85];[86].

The second software, the Freesurfer, is also open source and it was foremost developed by Anders Dale and Marty Sereno, who were studying applications to the construction of cortical surface models, turning it in the main motivation to the development of this software [87];[88]. Since then, much more features have being added with contributions of several people. And, in sum, Freesurfer is a package of powerful tools that provide a vast, full and automated analysis of key features in images of neuroimaging [89].

Lastly, the FSL software, also named FMRIB Software Library, was created by FMRIB (Oxford centre for Functional MRI of the Brain), although since then it has received a lot of contributions by members of other neuroimaging investigation sites. Hence, this software happens to be a library of different and independent tools that can be used separately or joint with others. These tools are for analysis of functional, structural and diffusion of brain imaging data, likewise the other programs. The implementation of this software occurs in C++ and scripts within Unix environment [90].

A brief analysis of these software it's enough to conclude that all are addressed for image processing. Nevertheless, FSL is implemented in the UNIX environment and allows to modify the different variables in the processing, being possible to create automated scripts with different commands. In addition, this software is the one that has the best results to study brain connectivity, which is what is intended in this dissertation.

2.5. Applications

Brain functional connectivity is in the present used in two major fields. One related to understanding the neural connections networks and brain's organization that exists across a group of subjects in a set of acquisitions - also denominated as connectome. And a clinical application that has the objective of investigate the effects of a pathology in the functional networks that distinguish a sick subject from a healthy subject, or as tool to study a disease's evolution or a response to treatments. The diseases of these cases are mostly of the psychological forum.

In this work it is given a particular attention to studies wherein the functional connectivity is used as a fingerprint in order to identify a specific subject or even a group. This approach can demonstrate that there are intrinsic features presented in the rs-fMRI that help to classify a subject, if the models achieve good classifications results. Then, there are some debriefings, like what are these distinctive features presented in the functional connectivity, and what they mean, that will have to be answered. In addition, the perspective of a clinician can lead a creation of methods that take into account more of the patient connectivity characteristics to a better treatment or analysis. Also, it is possible to use these models to classify a subject as healthy or unhealthy, using a group classification approach.

2.5.1. Human Fingerprint

There are investigations that demonstrate the brain connectivity can be applied as a fingerprint because identifies accurately a specific subject from a large group. Furthermore, they demonstrate this can be used to predict a cognitive behaviour or, more appropriately, the fluid intelligence [19].

In this work we used functional data from 126 subjects. Each subject did two rest sessions (one per day) and four task sessions (two per day). The task sessions approached the following areas: working memory, emotion, motor and language. The identification of subject, made after the construction of the connectivity matrices, was performed through an iterative process, in each one matrix was compared with all other matrices in the database, in order to find the one more similar. In this process there was one requirement: the matrices in comparison had to be scanned in different days. To the application of similarity was defined the Pearson correlation between vectors edge values of the two matrix in each process [19].

The results, applied to the whole-brain using 268 nodes Atlas, showed a success rate of 92,5% and 94,4% when used as target-database the pair Rest1-Rest2 or the inverse, respectively. Using the task sessions, when compared to resting states, one of the best rates was 54%, and between task-to-task the one of the rate was of 87,3%. And, finally, it was studied the combination of networks to see whether they increase the identification accuracy. Two networks emerged successfully resulting in the frontoparietal-based identification that between resting state 1 and resting state 2 had an accuracy of 98 to 99% [19].

2.5.2. Monitoring A Disease Evolution or A Treatment Progress

There are studies where it was used the Bayesian Hierarchical Model (BHM) to predict a future rs-fMRI. The approach was made individually for each subject or group. To create a model, it was used a baseline for rs-fMRI of a subject, or a group, and significant clinical information, or even demographic characteristics. The application objective is a tool to predict the changes in the functional networks over time, caused by a disease or a normal aging. Thus, it could be used to predict the final result of a treatment regimen and so adjust the best treatment for the patient in an individualized way. Besides, it is very important to study when is more appropriate an intervention analysing the predicted disease progression avoiding worst outcomes. The proposed approach used two datasets, rs-fMRI from Alzheimer's Disease Neuroimaging Initiative 2 (ADNI2) study and fMRI from kirby21 study [91]. The achieved results indicated that this approach has a better performance than other methods and, in a scientific way, that are differences in the functional connectivity of Alzheimer Disease (AD), Mild Cognitive Impairment (MCI) and normal control subjects' baselines, wherein the two first are more similar. Also, they demonstrated that the changes in functional connectivity over time are much more pronounced in AD subjects [92].

2.5.3. Study Psychiatric or Other Diseases

Currently it is the main use of brain connectivity in the neuroimaging community. The approach can be done by studying rs-fMRI, where is researched if a pathology creates distinctive patterns of functional connectivity in relation to a normal group. Examples to study are the schizophrenia and others brain diseases [93], addictive substance users [94], or patients with bipolar and a major depressive disorders [95].

3. Deep Learning

Artificial Intelligence (AI) is the parent field of Deep Learning (DL), once the last one is a research field of Machine Learning (ML). Since the appearance of AI, there was a need for the systems to acquire their knowledge in order to take some action or task. This lack led to the development of ML, which brought the capability of extracting knowledge from data analysis and patterns' identification. So, the performance of these methods is intrinsic related to the representation of the data and the generalization of the knowledge learnt for future use in new or real data. Thus, a simpler machine can perform better than a complex one, due to better representation and so, there is none correlation with the methods' complexity. For this, the solution of many AI tasks it's the classical approach, wherein the problem is solved by a simple feature selection before providing the chosen features to the ML algorithm. Approach that continues to evolve in order to improve or create new pipelines of pre-processing data transformations, to move towards more effective algorithms. This role is taken by feature engineering, fundamental in many applications of ML, which is associated to a difficult and expensive practise. This process is also very dependent of human input and low automated characteristics that deviate the system from the concept of AI. Therefore, it was important to create learning algorithms less dependent of feature engineering, which could easier the applicability of ML and increase the range of applications, moving towards the AI. And it was in this line that appeared another approach in ML, the Representation Learning.

Once, it is very difficult to make an optimal representation of the features underlying the data, the Representation Learning not only discovers the representation that explains the output but also construct the map representation. This mapping is done by capturing factors of variation that explain the input data. Although this approach could seem recent, the first data representation learning methods have been developed for more than 100 years. More specifically, the Principal Component Analysis (PCA) by K. Pearson in 1901 [96], a unsupervised method, and then in 1936 by R. Fisher [97] a supervised method, the Linear Discriminant Analysis (LDA).

The AI tasks have a tendency to be used more in the real-world data, associated to many factors of deviation with high-level of abstraction, therefore new methods are needed such as the Deep Learning [98];[99];[100];[101];[102].

Deep Learning is a type of representation learning method that can automatically extract complex representations from raw data with any feature selection *a priori*. The model is built on a set of layers that respect a hierarchical learning architecture that allows the representation of higher-level features using simpler representations, such as non-linear transformations. This is a resulting method of continuous development and conjugation described firstly by the Gartner Hype [103]. The first area of DL development appeared in the 1940-1960s, named as cybernetics, created the first models inspired in the biological neurons, as the Adaline and the Perceptron models [104];[105];[106]. Then appeared the concept of Connectionism in the 1980s-1990s, very related to the creation of backpropagation algorithm to train one or true hidden layers [107], an algorithm that continues to be today the main one to train and optimize Deep Neural Networks. Lastly, and more recently, the wave of research that began in 2006 known as Deep Learning with the development of more complex architectures related to the depth of the model. Since then, Deep Learning Networks have shown a great performance in different Machine Learning tasks, such as image classification [108];[109], speech recognition [110] and natural language processing [111]. The medical field was not an exception, specially in the medical image analysis research with several applications on the detection of structures as organs, tissues or even cells. Thus, other applications are possible such segmentation of organs like the brain, and then more abstract as tool for computer-aided detection and prognosis [112].

The characteristic more differentiating that was already mentioned is the greater depth of the model. As this characteristic is pointed as consequence of DNN success, several researchers are trying to understand more its role. A first hypothesis that appeared is that the depth is intrinsic correlated with the generalization bound [102]. But this hypothesis can only be used to explain shallow fully connected networks and cases of binary classification. So, more recently, appeared a more supported explanation, mentioned during this introduction, that depth models are able to represent more complex functions [113]. Increasing continually the depth did not means that the model performance will continue to improve. Although deeper networks have larger representation power that decreases the margin error of the model, the use of Rademacher Average (RA) to measure the margin bound showed that the margin bound increases with the increasing of model depth, affecting the final test error. So, there is a trade-off between this two points that explains the normal behaviour with increase of models depth, wherein first happens a decreasing and then increasing of test error [102].

The appearance of the term Big Data in the companies caused by the needed of collect useful information from a large amount of data, it was also a recent booster to DL. The quantity, the complexity of structures and the diverse types of data makes the DL a good approach. The Big Data can provide large amount of examples that to training this models is not a problem; on the opposite, it makes possible to find more reliable patterns and hence get models with better performance [114]. Moreover, due to the recent advances in the hardware with the appearance of computing resources of high performance, such as GPU supported by friendly software of easy applicability, the use of bigger and deeper models was possible. As consequence these models with billions of parameters with great representation improves even more the accuracy of state of art in multiple tasks. Besides, the successive improvements in learning algorithms with new methods to optimized deep models helped the application of the models [115];[116];[117].

3.1. The Basic Concept – The Neuron

The first concept that precedes all the models related the Deep Learning is the neuron, which in Deep Learning is also frequently denominated as node. This structure was developed based on the basic computational unit of the brain, the neuron (see Figure 3.1 (a)). The human nervous system has about 86 million neurons that are connected to approximately 10^{14} to 10^{15} synapses. So, each neuron receives several input signals in its multiples dendrites and produces the respective output along the single axon. This axon then suffers the division in several branches that connect other neurons' dendrites through synapses. Looking this from a computational point of view, the synapse controls the signal that pass to another neuron dendrites, depending on an intended inhibitory or excitatory response. Thus, there is a control of signal strength that in DL are known as weights, and once they are adjustable made feasible the capability of learning. Therefore, the signal passes to the cell body depends of the weight $(x_0 w_0)$. Then this resulting signal, like all other, results from the different dendrites that are received in the cell body. The approach in this basic model is summed of the different signals, resulted of neuronal fire if the final value is over a threshold. For the computational model it is important to know the firing rate of this neuron, by the frequency of its spikes Based on this, the firing rate is modelled using an activation function, as illustrated as is in the Figure 3.1 (b).

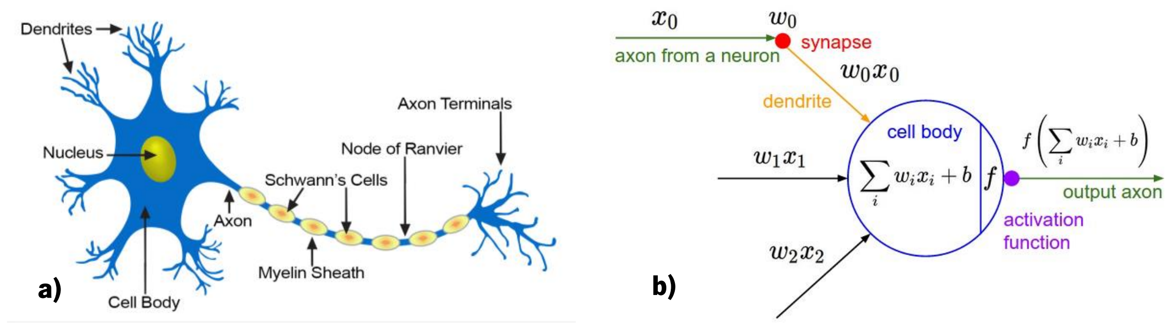


Figure 3.1 - Illustration of a biological neuron (a) (from [118]) and its corresponding basic computational model (b) (from [119]).

So, the mathematical model summarized for a set of inputs as a vector with size I, it's the sum of the multiplication of each input (X_j) with the weights (W) added to bias (b) of each neuron (i) in the layer, to which is applied an activation function (f), also known as a non-linearity function (represented in Equation (1).

$$y = f\left(\sum_{j=1}^{I}\left(W_{i,j} \times X_j\right) + b_i\right) \tag{1}$$

3.2. Deep Learning Models

The brain, the well-organized system known, can process information from different senses as hearing, sigh, touch, smell and taste. And the key to process such high-level of information is the collaboration of a large number of single neurons and its connections. Once the first models were construct based in this idea, they were called as Artificial Neural Networks (ANN). The ANN are the base concept of DL models wherein the neurons are organized layer by layer with a specific function and are stacked in several layers, in order to increase the complexity of the model and be possible to learn more high-level features. Besides, these models can be of two types according how the information passes thought the model. If the information passes only forward all along the model, where the cycles do not exist, the model is a feedforward network; if not, the model is a Recurrent Neural Network (RNN). In this work it will be used the feedforward approach, since the RNN are in the present very difficult to train and more difficult to implement, further as the objective is modelling relationships between a set of input variables and a set of output variables. So, the feedforward approach is the more appropriate, even though the Deep Learning can be implemented using the two types of networks.

Among the feedforward networks there are diverse types of models. The first network to appear known as Perceptron is constituted by a single feedforward layer and it was capable of classify patterns linearly separable [120]. Once this model was very limited, because could only describe data with linearity properties, the models grow in the number of layers, emerging the term Multi-Layer Neural Network (MLNN). MLNN is characterized by a model with more than two layers - the input and output layer, leading to appear the term hidden layer to define layers between the last two mentioned. This also can be named as Multilayer Perceptron since the hidden layers are non-linearity layers. Besides, this type of layers is in Deep Learning called as fully connected layers since all pairs of nodes in two consecutive layers are connected, as shown in Figure 3.2. More in detail, the model architecture has always an input layer where the input is the vector (X) with the values $\{x_1, x_2, x_3\}$ of each example input, and an output layer where the size depends on the number of classification classes used in case the classification task, or on the number of values to predict in the regression task, vector (Y). Moreover, inside the model there are hidden layers responsible for making the different representation learning, where each layer has the respective resulting: an activation vector (Z, U). It is also important to emphasize that in the layer; the neurons are not connected between each other. The number of hidden layers is what define the depth of the model. Normally, a model with more than two hidden layers (so four in total) can be labelled as a Deep Learning model, because only this way the layers can learn more complex representations. In addition, these neural networks can be used together with other type of architectures as the Convolutional Neural Networks (CNN), to get models with better performance.

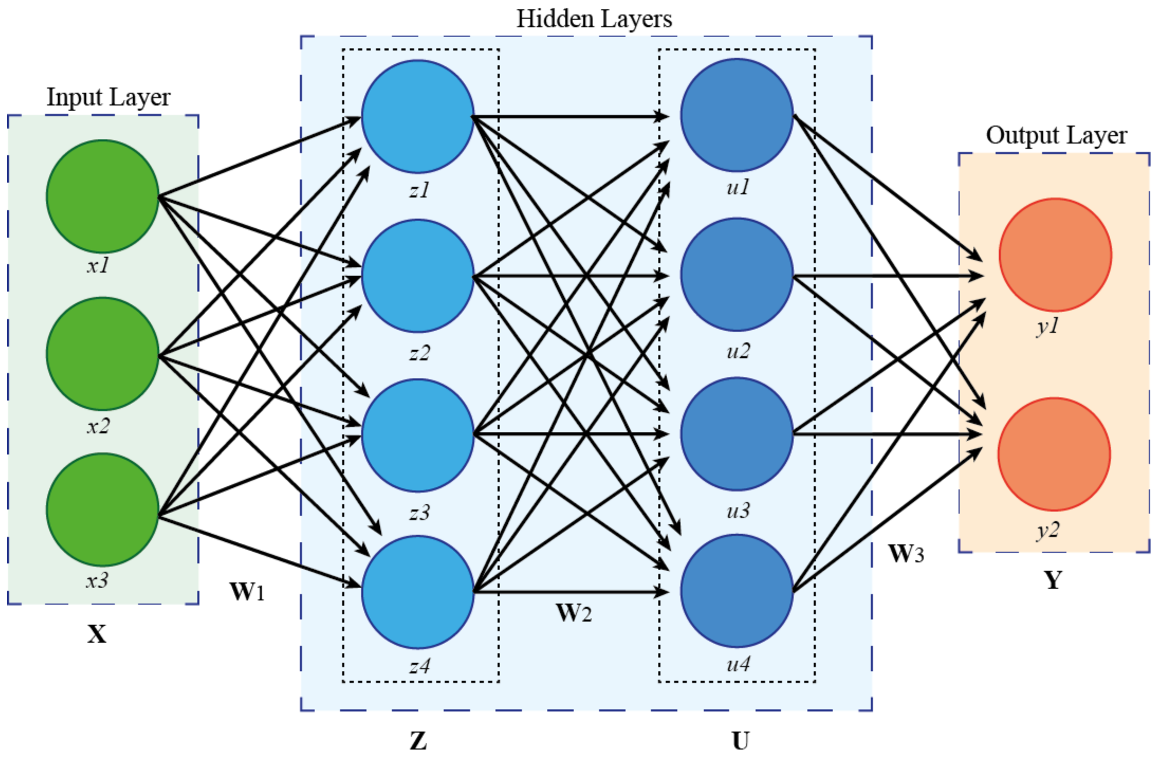


Figure 3.2 - A feedforward and multi-layer neural network with 4 layers (2 hidden layers).

The learning of the fully connected layers is preferential done using the supervised learning to solve classification and regression problems. Therefore, there is the knowledge of the input variables and the respective output for the learning algorithm learn iteratively, making predictions on a training data and adjusting its parameters. But there are also others Deep models used in applications for unsupervised feature representation learning. Particularly, in this kind of learning, the data has no output and the algorithm must properly adjust to the structure of the data in the end, it will be possible to extract a set of features that better represent without the necessity of a feature extractor (computer program or hand designed). The most used for this end is the stacked auto-encoder [121];[122], the deep belief network [123];[124], and the deep Boltzmann machine [125].

In all the types of models already mentioned, the input must have a vector form. The location of the pixels (2D) or voxels (3D) and its relationship with the structures neighbours in the image files, has important structural information that is loosed if it is vectorised. Therefore, Convolutional Neural Networks (CNN's) were created in the Deep Learning to use this spatial and configured information receiving as input 2D or 3D images. The CNN's are characterized structurally by convolutional layers combined with pooling layers followed by a multilayer neural network [126]. The CNN's are used with both supervised [109] and unsupervised learning [127].

3.3. Layers

The Deep Learning models are characterized by a set of connected and hierarchical layers. Layers that could have distinct functions in the learning process. They are, after the neuron, the second basic unity of the model, and its adjustment and finding its best location in the network affects completely the model performance. They also characterize the model in two different aspects: the depth and the thickness. The depth is related to the number of layers and the thickness is the number of nodes by layer.

3.3.1. Linear

The linear layer is the simple model that applies a linear transformation for an input vector x resulting in an output vector y, as showed in the Equation (2).

$$y = A * x + b \tag{2}$$

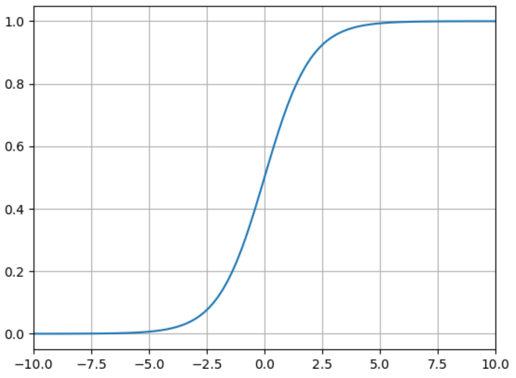
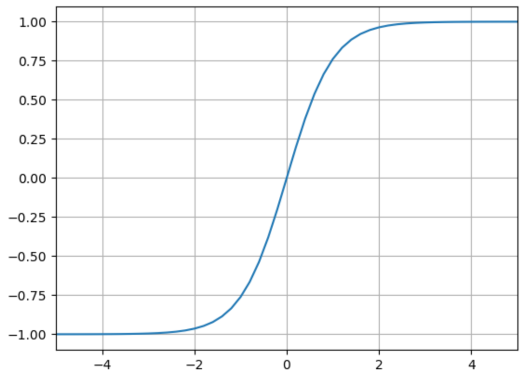
In these cases, the activation function is the identity resulting in the output of the real values.

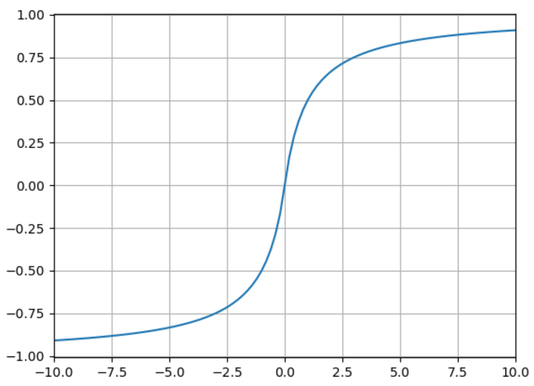
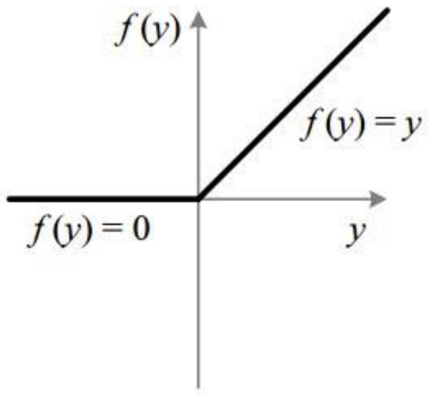
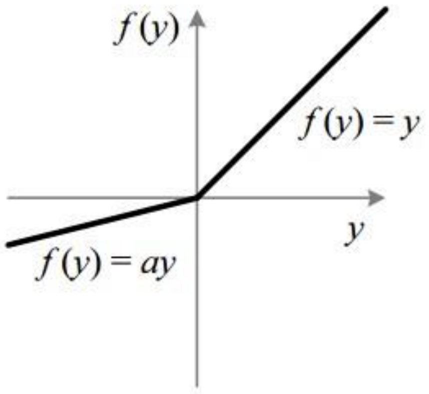
3.3.2. Activation or Non-Linearity

The non-linearity existing in the layers is the reason to neural networks having the capability to create models that approximate to any possible function. For that reason, the application of non-linear functions between the principles are the sigmoid, the hyperbolic function and Softsign. More recently the ReLU functions and some extensions have been imposing in the Deep Learning application since they have better performance in deeper models comparatively to others, like sigmoid that has the problem of have lower learning velocities in the deeper hidden layers. The main functions are in the Table 3.1 with the equation function and the respective plot.

The sigmoid non-linearity function, as can see in the Equation (3) of Table 3.1, is the function that transforms the real values in a range between 0 and 1. Due to the sigmoid behaviour, large values tends to become 1 and low values 0, leading the sigmoid to enter in a phenomenal called saturating state. And consequently, the resulting gradients are almost null, reducing heavily the learning process. Also, there is special attention to the initialization, once great neurons weights can easily saturate the function preclude of happening any learning. The hyperbolic tangent is a function that derivate from the sigmoid function, but in this case the values are squashed between -1 and 1. Once it has the same behaviour as the sigmoid, it suffers from the same problem of saturating for high and low values but, in this case, the output is not zero-centred resulting in less instable gradients. The Softsign function is nonlinear and is an alternative to hyperbolic tangent since it is not so easily saturated as can be seen in the plot of the corresponding function.

Table 3.1 - The principal activations functions used in Deep Learning models, with respective equation and plot.

Activation function	**Equation function**	**Function plot**
Sigmoid	$\sigma(y) = 1/(1 + e^{-y})$ *(3)*	
Hyperbolic Tangent (tanh)	$\tanh(y) = 2\sigma(2y) - 1$ *(4)*	

Softsign	$softsign(x) = \frac{y}{1+\lvert y \rvert}$ *(5)*	
ReLU	$f(y) = \max(y, 0)$ *(6)*	
Leaky ReLU	$f(y) = \max(y, y \times k)$, $where\ 0 < k < 1$ *(7)*	

The ReLU (Rectified Linear Unit) function has become very popular in computer vision applied to deeper networks since it is not saturated for larger values. The negative values are put aside from the learning process, so it depends in the data and in the application if the negative data has not importance and some weight in the learning. As mentioned before, the tradicional ReLU does not use non-positive values, so Leaky Relu is a modification that allows to control the error that propagates backwards when the value y is negative.

The softmax activation is typical used in classification tasks in the output layer ensuring that the outputs are probabilities, and all summed up are one. So, the softmax function squashes a vector of real-values scores in a vector of values between zero and one whose sum is one. When this activation function is the output, the final output is the neuron where the probability is higher.

3.3.3. Spatial Convolution

The spatial convolution layers come in the application of Deep Learning in images classification. Once this method is intended to not have any feature selection or data transformation, the use of images in regular neural networks in linear or nonlinearity layers was impossible. The term image appears here not it the broad sense of something visual but in the form. An image normally consists in three channels (red, green, blue, e.g., RGB) with a matrix associated to each channel where the number of rows and columns correspond to the width and height in number of pixels of the image. So, an image with 3 channels and square with width and height equals to 64 pixels provides 3 x 64 x 64 = 1228 inputs, giving 12289 final parameters (with bias) for each single node. Consequently, the model will be too much big with too much parameters, affecting the velocity of training and increasing the probability of lack of generalization, e.g. overfitting problem. The spatial convolution comes to reduce the number of parameters taking the advantage that input has spatial relationships, once neighbour pixels carry normally similar information. Further, it finds a pattern in the images or other type of data. In this work, convolutional layers are used to find patterns in the correlation matrices. In order to find these spatial relationships, the convolution layer learns an intended number of filters for each channel that, when convoluted spatially with the input image, produces 2D features maps equal to number of filters. The feature maps size depends in the size of the filter, and the number of convolutions that happens between each filter and each region, results in all the possible displacements of the filter over the matrix. The advantage of this layer comparatively to a conventional layer is that the weights are shared over the full image decreasing the number of parameters.

3.3.4. Spatial Pooling

The spatial pooling layer is a layer normally presented in the convolutional models to reduce the size of features maps and, consequently, the number of parameters, as well as the computation of the same during the training process. Besides, it gives to the model more invariance in very similar images, reducing the dimensionality of intermediate representations improving the generalization model ability.

The pooling is done iteratively by a filter with a specific size that moves all long the features maps and, for each process, it results in a value. The spatial intended can be controlled by the filter size, as well as the dislocation in each iteration, and the intended function applied in the filter to choose the resulting value.

3.4. Training

3.4.1. Process

In order to train any DL model, it is necessary a set of methods and algorithms with a specific function towards the same objective: the creation of a model with good performance in the required task, as for the classification (predict a class) or the regression (predict a value). The main parts presented in the training are the initialization, the backpropagation and optimization.

The initialization is the first part of the training process, and it is only performed one time in the beginning to initialize the weights and bias from the model. Normally the weights parameters are initialized as Gaussian random variables with mean 0 and standard deviation equal to 1 divided by the number of inputs, and bias is initialized to zero. But there are several approaches resulting of the researches in this field that will be mentioned in the initialization point.

After the initialization, begins the training process that iteratively uses one or more examples in each training iteration. The training with only one example is usually entitled as online learning, and in the case of various inputs, the set is more known as batch. During the training, it is used a loss function in order to quantify the capacity of the network to predict the ground truth labels or values for all the training inputs. This loss will be the variable of study of the training and consequently in the backpropagation and optimization. The training process starts to compute the prediction for the inputs given and all the associated losses to obtain the final cost of the model, and then it is applied the backpropagation algorithm. The algorithm propagates the error since the output layer until the input layer, where for each layer is computed all the partial derivatives, e.g. $\frac{\partial E}{\partial w}$ and $\frac{\partial E}{\partial b}$ of the loss function (E) for all weights (w) and bias (b). The partial derivative, as it is known in the mathematical field, gives the slope of the study variable in the function, which gives the information about how the slightly-scale changes in the input affects the output. Thus, with the application of the derivatives applications it is formed a gradient and it is known how the bias and weights affects the full error. This gradient is essential for the next step: the optimization. This step uses algorithms to optimize the loss of the model searching for the convergence of the value in order to find a minimum that ensures good predictions. Minimum that in the best occasions is the global minimum, but in the major of the situations a local minimum that performs nearly as well as the global is enough and not easy to achieve. The basic concept of optimization takes the gradient as a vector where are used the negative values in order to move in the direction of negative gradient – gradient descendent – converging to a minimum. This can be described as Equation (8) to calculate new parameters calculated (x'), using the gradient of the function $\nabla_x f(x)$, the old values (x) and the learning rate (μ).

$$x' = x - \mu \nabla_x f(x) \tag{8}$$

As can be seen in the Equation (8), the learning rate has a great role in how the optimization process runs and, so it has to be fine-tuned. If the learning rate is too large, the descending steps made in the gradient will be overdone. That will increase the probability of the process never reach an optimal minimum point, making for example the values oscillating from one side to another in a valley of the loss function. But if the learning rate was too small, they could cease the gradient descendent in a local minimum that performs poorly and, due to small steps, it is not able to exit from the minimum valley. Besides, it increases the training time too much. The optimizers have also improved and have been added new approaches besides the gradient descendent to get the function converging to the optimal minimum.

3.4.2. Initialization

The initialization is an important step that shall not be discarded. It can differentiate a network that easily converges from an optimal minimum from a network that does not go anywhere, even with an application of a large amount of training iterations. One of the problems is in the case of the layers weighs being too small, once the variance of input value diminishes when passes through each layer in the network. So, the weight value drops to a very small value making it no more useful for the learning process. Also, it depends on the activation function applied, since its behaviour will determine which values variance that are useful or not to network. In the case of the sigmoid function that has a flattened comportment in high and low values, if the initialization produces too small or too high initial parameters, their gradients will quickly tend to zero stopping the learning process.

Thus, there are different approaches to initialization. Krizhevsky et al. [109] in his research work of ImageNet classification, initialized the weighs parameters in each layer with a zero-mean Gaussian distribution with a standard deviation of 0,01. And the bias was initialized as one in the first CNN's layers and fully connected layers, being equal to zero in the remaining. This was the best approach according to model architecture and non-linearity function applied. The most popular approach is from Xavier et al., whose weights are initialized following a distribution with zero mean and variance 1 divided by N, where N is the average of number of input and output neurons [128]. The objective is to maintain the variance through the layer of the network to not impair the learning process. But this initialization does not work so well in the non-linearity ReLU layers [129]. The Xavier's initializer is also known as Glorot's initializer, and there are some adaptations for uniform and normal distributions to the different activations functions. Due to the effectiveness of Xavier initialization in the ReLU activation functions, He et al.[130] proposed a new approach with better results to this type of functions, whose unique modification was in the standard deviation - in this case is $\sqrt{2}$ by the number of the inputs nodes. In addition, Saxe et al. created a new initialization method that instead of using random Gaussian matrices initialization, recommends initialization by random orthogonal matrices [131].

3.4.3. Loss Functions

The loss functions that measure the performance of the models relatively to error are diverse, but it will only be mentioned three of the most used. The cross-entropy and hinge functions are the most applied in classification tasks, while others, like mean square error, although could be used in classification tasks, are more efficient in regression tasks.

The cross-entropy function is very popular in the Deep Learning community, consequence of the good performances related to posterior probability in diverse problems even in the cases where there are problems with limited data [132]. Also, the fact that the function uses logarithmic describes better the error, more specifically when the model output is close to the desired output for all training inputs, being the error almost zero. The cross-entropy cost function for a total number of n items in the training data, for all training inputs (x) and for a desired output (y) and the actual output (a) can be described as seen in Equation (9).

$$E(a, y) = -\frac{1}{n}\sum_{n=1}^{n}[yln(a) + (1 - y)\ln(1 - a)] \tag{9}$$

The Hinge loss function measures the loss separating the positive and negative inputs and so is used for the maximum-margin classification, very popular also in Support Vector Machines (SVMs). For n items in training data, a desired output $x \in \{-1, 1\}$, e.g. case of binary classification, the predicted output (y) and a margin (m) are part of the hinge loss of the prediction, as defined in the Equation (10).

$$E(x,y) = -\frac{1}{n}\sum_{n=1}^{n} \max(0,1.m - y.x) \qquad (10)$$

In the basic Hinge loss function, the margin is one, but it could be customized. As verified by the expression, when the desired and the predicted outputs have the same signal, - correctly classified, - the loss is 0, but on the opposite the loss increases linearly depending of the predict output. This made the loss function convex useful in the optimization process making easier to minimize. Although, it has the problem of not being differentiable, but this limitation was solved by computing its gradient locally.

The Mean Squared Error (MSE), or quadratic, is the loss function usually used in linear regression to measure the models' performance. But it is also applied in classification tasks in DL even for deeper architectures with some sophistications [133]. Moreover, the standard form of MSE for n examples and for an input (x) and a predicted output (y), is defined as in the Equation (11). The optimization target of this loss function is to minimize the residual sum of squares.

$$E(x,y) = -\frac{1}{n}\sum_{i=1}^{n} (x - y)^2 \qquad (11)$$

3.4.4. Optimization Algorithms

The optimization has the leading role in the final model performance, so there was, since the beginning, a search for effective algorithms in discovering the best parameters in the training model without affecting too much the efficiency, once the gradients grew tremendous with the application of deeper models and made the process of optimization more difficult (more layers interconnected).

The main optimizer algorithm used in DL applications is the Stochastic Gradient Descendent (SGD), based in the gradient descendent algorithm with a simple simplification in order to speed up the learning. The idea is to estimate the gradient by computing it for a small and randomly sample of training inputs. Thus, averaging the gradient over this small sample makes the process consume less time. Besides, it reduces the variance in the parameter update leading to more stable convergence. The fine-tuning of the learning rate used in the SGD is difficult, but with approaches as learning decay can help to occur a good convergence to a local optimum. Moreover, in many applications it is used a momentum associated to algorithm, which is added to the dislocation vector, increasing the values updated all the time. This can help the SGD in situations that do not perform so well, as for example areas wherein the surfaces curves much more sharply in one dimension than in another usually presented around the local optimum [134]. That makes the SGD oscillating more across the ravine than towards a local optimum. The momentum will increase the parameters updates in case the gradients point in the same direction and the opposing if they are contradictory, resulting in less variation and faster convergence [135]. Furthermore, Nesterov has created a variant of momentum known as Nesterov's Accelerated Gradient (NAG) that helps the choice of the best gradient associated to the momentum in order to best learning [136]. But, besides this, there are other variations applied to SGD that result from new approaches in order to improve the optimizer performance for a particular application [137];[138];[139].

The Adagrad is a gradient-based optimization algorithm that during the training adapts the learning rate to the parameters, wherein the updates occur less frequently, suffering larger updates and more frequent parameters smaller updates [140]. So, it's suitable for cases where applications with few examples in the data. This optimizer has showed to be a good optimizer option by the google approaches for training large networks [141]. From the Adagrad appeared some extensions, as Adadelta and RMSProp, mainly to reduce the radical form of decreasing continually the learning rate during the training. In opposite of Adagrad that accumulates all the past squared gradients, the Adadelta make the sum value continuously increase while the learning rate shrink more and more causing the learning rate too small, [142]. The RMSprop is another approach to Adagrad, yet unpublished, proposed by Geoff Hinton, that although has created independently, the concept is similar to the first vector update in Adadelta. The learning rate is divided by an exponentially decaying average of squared gradients [143].

The Adam algorithm from Adaptive Moment Estimation is another approach that computes the learning rates for each parameter, in addition to use the exponentially decaying of the past square gradients as Adadelta and RMSprop, also uses the exponentially decaying of the past gradients that correspond respectively to uncentered variance (second moment) and mean (first moment) of the gradients. Applications using this optimizer confirmed that this is a robust and suited optimizer for non-convex optimization problems associated to low-consumption of memory hardware resources [144]. There is also an extension of the Adam algorithm that incorporates the Nesterov accelerated gradient and, so it is known as Nadam optimizer, in order to improve the speed of convergence and the performance of the learned models [145].

The different optimizers described above are only a subset of the optimizers available and applied in the training DL, but these are the most important. The optimizer performance in a problem depending also in other factors. Firstly, it depends in the data, in how distributed the values are and its size; secondly, it depends on the task type and loss function used; and lastly, depends on the architecture model, the number of layers, the number of nodes by layer and the function of each layer.

3.5. Generalization – Objective

The main objective and the challenge in Deep Learning as in other techniques of Machine Learning is that the model learned must perform so well on new and unseen data as was for the training data. This ability of the model is called generalization and the unseen data is called the test data.

In the training process of any machine learning method the error optimized and computed is the training error, but this is only useful for measuring the optimization process. So, it is used the test data to get the test error representing the generalization error. This test data is a set of examples that was grouped separately from training data. The creation of the training and test data is based in the probability distribution over the datasets, so there are some assumptions that should be respected. One assumption is that each example in both datasets are independent from the others; the second assumption is that both datasets are identically distributed, being characterized by the same probability distribution. Given that, if it is used any model, the training and test errors for each dataset should be the same. But, after the training process where the model parameters are chosen to reduce the training error, the test error will be greater, or in the best situations equal, to the training error. So, there are two aspects that will be determined in how a ML model, as DL, will have the ability to perform well. Primary, if during the training of the model, the error converges to a small value, and secondary, if, as a measure of the generalization, makes the gap between the training and the test data smaller as possible. These two factors will determine how the model performs, whether it is appropriated to perform a task application, or whether it is in a situation of underfitting or overfitting. The model is in underfitting when it is not able to converge enough the test error, and so it is not a good representation in the learning of the inputs. In the case of the overfitting, it occurs when the gap between the test and the training error becomes too large, making precluding the generalization process. This trade-off between underfit and overfit relies in the model capacity as is shown in Figure 3.3.

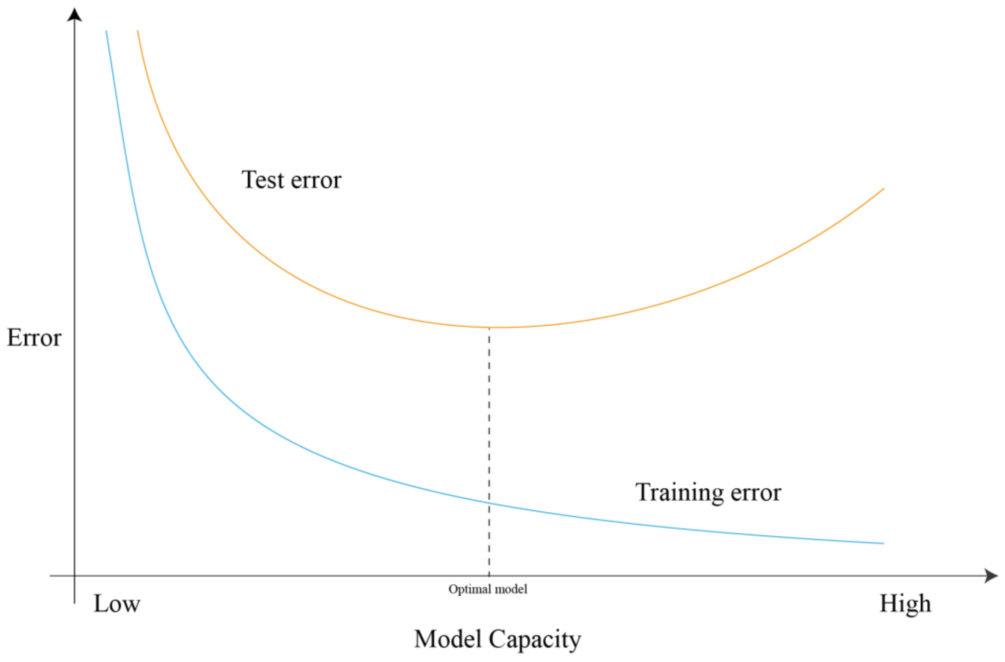


Figure 3.3 – Training and test curve changing the model capacity.

The capacity of the model is related to the ability of the model to fit in distinct functions of different tasks and applications. A higher capacity gives the possibility to the model to learn more complex tasks, but this has to be controlled in order for the capacity not being higher than needed, impairing the generalization and making the model enter in overfitting. If the capacity is too low, the model is unable to solve complex tasks, being characterized as a model in underfitting. The capacity of a model is intrinsically related to the model's architecture as the number of layers (depth) and the number of nodes by layer, increasing the number of parameters in the model and, consequently, its representational power. Nevertheless, the capacity is not only dependent on the chosen model topology, but it is also related to the optimizer used in the learning process and its parameters. The fact of the learning be affected by the optimization, which is dependent of the data and the error function to optimize, the term to refer the capacity of the model is effective capacity. Besides, there are other variables that can be used to control the behaviour of the learning algorithm - the hyperparameters. These values are not adapted by the model itself, so they had to be hand tuned or other learning algorithm used to learn and identify which are the best parameters. Once these hyperparameters control and directly affect the model capacity, the best fine-tuning of these parameters will always result in the overfitting, if it has learnt using the training data. And it is why that in Deep Learning and in other techniques a validation data is used, a set of unseen examples not used in the training and test data. Thus, it is avoided the overfitting; the best parameters are selected by the validation data and then it is made a final test to get the generalization of the model. The fact that the validation data is used in the selection of the hyperparameters, this will underestimate the true generalization error. So, the test data should not be used to select the hyperparameters, therefore the validation data is constructed using a subset of the training set, typically 80% of training data is used for training and 20% is used for validation. These fixed divisions into the different datasets have the problem of adding statistical uncertainty around the estimated average test error, specially when the test data has not the enough size. Hence, it were created new procedures to overcome this problem, based in the fact that this statistical uncertainty decreases if the training and the test are repeated several times for different and randomly chosen subsets or random splits of the full dataset. The most widely used procedure is the cross-validation, which allows to train and test the model k-times on different subsets of the original data giving an estimation of the performance of the model in unseen data.

As the capacity shows a great importance in the performance of the model, a lot of researchers investigated ways to measure it, such as VC dimension [146], uniform stability [147];[148] and Rademacher complexity [149]. The different theories were also used to proof the assumption that capacity is the variable that explained the good generalization of the Deep Learning models. More specific, they showed that the gap between the training and the test error increases as the model capacity grows as was explained before and is shown in Figure 3.3, and, on the other hand, the curves tends to approximate when the number of examples used is increased. But once the models in Deep Learning continue to increase the number of parameters caused by application of more and more deeper models with a considerable number of nodes by layer (example the Inception or Alexnet), the different theories showed inability to measure the real model capacity. Besides, the understanding of learning algorithms related to non-convex algorithms continues to be a mystery and they affect the model capacity. Moreover, recently Zhang et. al [150] showed with different and recent Deep Learning models that it is capable of memorize random labels with none training error and even with small test error for some natural datasets as CIFAR-10 (large image dataset available online [151]). This means that these models are in real overfitting, once the different training examples were completely memorised rather than learn predictive features from the data. And although during the tests the full model (architecture, size, hyperparameters or optimizer) has been maintained, the generalization error was increasing directly with the noise in the training data (percentage of data randomize). So, the capacity of the model was maintained but the generalization error changed. Thus, the answer to the question "Why the Deep Learning generalizes so well?"- continues to be a mystery, and there are new methods to measure the generalization ability, as already was began in [152]. Besides, the Zhang et al demonstrated that regularization on the norm of the weights is not necessary to obtain small test errors [150].

3.6. Reduce overfitting problem

The overfitting that was already mentioned is the contrary of the intended - the generalization of the model. This is the major problem in neural networks, especially in the modern networks associated to more parameters, and so more representation power. Therefore, it is important to detect when overfitting happens to not over train the model and have some techniques that reduce the effects of overfitting.

To detect the over train of the model it is used a validation data formed by examples not used in the training that for each training will show how the model performs in unseen data, giving a feedback of the training process. There is also a popular resampling technique to estimate the model generalization, the cross validation. Additionally, there are some techniques to reduce the overfitting problem that will be explored, such as early stopping, regularization, dropout, batch normalization and data augmentation. A simple and good approach will be increasing the size of the training datasets, in order to increase the differences through the dataset, but in the research areas this data is normally expensive to obtain.

3.6.1. Early Stopping

The early stopping is a basic and logic way to reduce the overfitting problem, wherein the method stops the training process when the validation accuracy is saturated. Moment that is characterized by the increase continuous of the validation error and not as a result of some fluctuations.

3.6.2. Regularization

The regularization is another method to reduce the overfitting problem in the model in which the idea is modify the learning algorithm of a model in order to reduce the generalization error without affecting too much the training error. The weight decay and, L1 and L2 regularization are examples of different techniques available. But all of them use the same idea that the regularization of the model is done by adding a penalty (regularizer). As example it is shown in the Equation (12) the final error function (E) using the cross-entropy loss function and the L2 regularization for a layer (L) with j neurons, with the respective activation (a) and the desired output (y).

$$E(a,y) = -\frac{1}{n}\sum_{nj}\left[y_j ln\left(a_j^L\right) + \left(1-y_j\right)\ln\left(1-a_j^L\right)\right] + \frac{\lambda}{2n}\sum_{w} w^2 \,, \qquad where\ \lambda > 0 \tag{12}$$

As we can see in the Equation (12), the regularization term can be controlled by λ, the regularization parameter that is scaled by the size of the training size n. The addition of this regularization penalty over the weighs shows that the model will be prefer, during training, to learn from small weights. So, the large weights only be important in the learning if they achieve improvement from the first part of the resulting function cost, in this case the cross-entropy cost function. Thus, during the training will not be given so much weight in the features that are easier to learn to put also the weight in fewer features, improving the generalization [98].

3.6.3. Dropout

The dropout [116] is a recent technique (2012) that showed to be very effective to prevent the overfitting problem. As the name denotes, during the training there are nodes that are dropped from the network, so they are not used in the training process. This dropping is only temporary and for each training iteration the nodes dropped are chosen randomly. And the rate of the nodes dropped is chosen as intended as if it were other training hyperparameter. The final intention is during the training the neurons learn better representations that explain the all data and not representations based in some informative complex features of the data, and so not good for generalization.

3.6.4. Batch Normalization Layer

In this method it is added the batch normalization inside the network by a layer, making the normalization as a part of the model architecture and performing the normalization for each training mini-batch independently of the batch size used. The layer is normally applied before the non-linearity layer and do not affect the representation ability of the network. It adds a normalization step, so there is a shifting of the inputs to zero mean and unit variance, removing possible internal covariate shift. Problem related to the fact that the distributions are continuously changing during the training in the inputs of each layer that affects the learning. So, this makes the optimization process easier to converge and gives the possibility of using higher learning rates, less careful with the initialization and, in some cases, eliminate the need for dropout. The fact that this helps the overfitting is that the model improves its performance in the test set, so the generalization error decreases. Besides with this layer it is possible to get the same performance with a model with less capacity as other with more, - it is used a model with less probability to enter in overfitting [153].

3.6.5. Data Augmentation

This technique is used to increase the training set, which was already mentioned as the preferable way to eliminate the overfitting problem. There are some data there is difficult to collect, more particularly, to relate to the medical field, where there is access to specialized data, such as images or videos that is expensive and has always privacy concerns. And this happens in other AI areas. So, there was a search for different approaches to create effective augmentation techniques, approaches that were already shown to be very effective in image classification [154];[155].

4. MATERIALS

The materials of this work are related to the research problem "*Can the functional connectivity from rs-fMRI be used as feature in subject's classification tasks using a Deep Learning approach?*". As is mentioned in the question, the material in study and used as features for the model was functional connectivity, which values was extracted from resting-state fMRI from different subjects. Once the learning process of the models is based in the supervised learning, was used some information as the subject as identity, sex or age. There was also the necessity of atlases, resources essential for get the functional connectivity by the pretended and defined brain regions.

Part of the developed worked and research questions are related to the results and work performed by Shen et al. [19], wherein the functional connectivity provided from rs-fMRI is used as subjects' fingerprint. Therefore, the approach follow in this work was based in the article, which was used two different sessions of rs-fMRI acquired in different days for a subjects' set, wherein each correlation matrix with functional connectivity for each session was correlated by Pearson's correlation with all the other correlation matrices in the other session. And for each correlation matrix (subject), if the better correlation value of the pair-wise belongs to same subject the classification was made successfully.

4.1. Atlases

In this work the brain parcellation was done using atlas with nodes well defined and applied in other studies. Thus, was selected 6 brain atlases. The AAL [75] atlas is the anatomical atlas the most widely used in functional brain networks studies that divides the subcortical structures and cortex using the gyrus and sulcus boundaries. The full atlas comprehends 116 brains regions, but in the majority of the researches the cerebellum is omitted resulting in 90 ROIs. The Freesurfer atlas, is a combination of two atlas based in different segmentations, available by Freesurfer software. This atlas is formed by 146 regions form the 'Desikan-Killiany' cortical atlas [156] and 14 regions from the 'Destrieux' cortical atlas [157], creating a final atlas with 160 nodes. As is based in structural segmentation of brain structures this atlas is also anatomical, and it was applied to get more knowledge about it. Since was already used in studies in the Neuroimaging lab of the ICVS [158]. The other 4 atlases was provided from the same study with different number of nodes [50]. These are recent atlas but already has a great importance in studies with particular importance in functional connectivity studies. Besides this was the atlas used in the "fingerprint" that was reproduce and compared in this work. But despite their use only the 268 nodes atlases, in this work are test other atlas from the same study. The reason is to test the different parcellations and see how they influence the results. In the study of they conclude that 268 nodes are the best atlas to use once the reproducibility cross-subjects was good without compromising the acquisition resolution and the anatomic variance of the subjects. They also conclude that the time-series of resting state resulted from your atlas was more coherent that the obtained by AAL atlas.

The different atlas has different number of ROIs and in the atlas process creation the approach based the study was different (Table 4.1). The name of Shen Et al. atlas don't coincide with the number of total of nodes, because the name is associated to number of "seeds" used in each cerebral hemisphere in the creation process. Besides some nodes was eliminated due not represent correctly the brain region functionality.

Table 4.1 - Table of atlases used in this work with the number of total nodes, descending order, and the brain approach used to creation of the atlas.

Parcellation Name	**Total regions**	**Brain Based approach**
150 nodes	278	Functional
268 nodes	268	Functional
100 nodes	184	Functional
Freesurfer	160	Anatomical
Aal	116	Anatomical
50 nodes	93	Functional

4.2. Data

The resting state fMRI data used in this work was obtained from two different sources and named according to it. The named In-House data was got from a study for the SWITCHBOX Consortium project (http://www.switchbox-online.eu/) realized in the ICVS institution and given amicably. The other named HCP comes from a global initiative called WU-Minn Human Connectome Project (HCP) consortium [159]; [160]. Their aim is characterizing the human brain and function of 1200 healthy adults and analyse their characteristics at individual and level group. So, in this initiative it was collected a very large of samples of imaging data such as MRI, rs-fMRI, task fMRI or diffusion data. The project has an open data policy and the public disclosure data format provided is since a raw until a data with a specific pre-processing pipeline implemented in the project to a particular goal. As this initiative there are others like: the UK biobank [161]; [162], Japan's Brain Mapping [163], China Brain Project [164] and Human Brain Project [165].

The data used in this work are volumes with the temporal data saved from rs-fMRI. Initial, the data is a set of DICOM slices brain images of different temporal points of the acquisition that are converted in a single volume file. Then they suffer an own pre-processing pipeline to get the value of functional connectivity to a few brain regions chosen *a priori* in the parcellation step in processing. The resulting file of processing is only one file with the values of correlation of intensities of two brain regions.

In this work the data used was obtained of two independent rs-fMRI datasets, a HCP dataset publicly available online and an In-House dataset from an old study done in ICVS. Both datasets are composed by two rest sessions acquired in different days. The interval time between the two sessions was different for each situation. For HCP that time was only about one day; however, for In-House data that was about one year. Another difference was the number of subjects in each study: HCP dataset has data from 100 subjects and in In-House data from 76 subjects.

4.2.1. In-House Dataset

This dataset is set of rs-fMRI data of 76 different subjects, of two different sessions. It is sample of the subjects recruited for the SWITCHBOX Consortium project (www.switchbox-online.eu/). The study respected and was done according the principles expressed in the Declaration of Helsinki and as approved by the Ethics Committee of Hospital de Braga (Portugal). Besides all the subjects gave an informed written consent.

Relatively to image data, the session 1 had more subjects with rs-fMRI, about 107, however the number was reduced to the subjects that was present in both sessions. Once the classification uses one session to train and the other to test, there is no needed add data to the model not applied in the test, or otherwise, train a model and test the model with data unknown. The data quality is poor than the HCP data, once the significant differences between the strength of the magnetic fields used in the acquisitions, 1.5T comparatively to 7T.

Finally, the dataset has 76 subjects, 39 male and 37 female. The mean age is 64,86 ± 7,96 years, wherein the max age is 82 years and minimum age is 51 years.

4.2.1.1. Magnetic Resonance Imaging Acquisition

The two imaging sessions were acquired on clinical approved Siemens Magnetom Avanto 1.5 T MRI scanner in addition with a 12-channel receive-only head-coil. For each session it was extracted one structural and one rs-fMRI acquisitions. Relatively to the structural acquisition, the T1-weighted magnetization prepared rapid gradient echo (MPRAGE) was prepared with the following parameters:

- 176 sagittal slices
- TR/TE = 2,730s/3,48 ms
- FA = 7°
- slice thickness = 1 mm
- slice gap = 0 mm
- voxel size = 1 x 3 x 1 mm^2
- FoV = 256 mm

In the rs-fMRI was used the Blood Oxygen Level Dependent (BOLD) sensitive Echo-Planar Imaging (EPI) sequence with the next parameters:

- 30 axial slices
- TR/TE = 2,000s/30 ms
- FA = 90°
- slice thickness = 3,5
- slice gap = 0,48 mm
- voxel size = 3,5 x 3 x 3,5 mm^2
- FoV = 1,344 mm
- 180 volumes

Several resting state acquisition requirements were asked to the subjects, such as, remain still, eyes closed and try to think of nothing in particular and awake, condition verified by the subject in final of acquisition.

4.2.1.2. Data Conversion, Pre-Processing and Parcellation

Before the pre-processing process itself, it was done the conversion of the images DICOM format for NIfTI file using the converting tool dcm2nii (https://www.nitrc.org/plugins/mwiki/index.php/dcm2nii:MainPage). In the pre-processing once the structural MRI data was used to help the spatial normalization step of the fMRI, was needed apply pre-processing steps to get the final volume with spatial coefficients. So, the pre-processing begins for the structural data, and first was done the skull stripping using the Brain Extraction Tool (BET) from FSL. In this step is important choice the correct fractional intensity threshold, to extract correctly the brain. The default value used in our Neuroimaging lab is 0,2, but the different tests with this data showed better results with 0,18. Then was done the linear and non-linear registration from the structural space with a standard space finishing with the resampling to 2 mm isotropic voxel size. After it proceeded to the fMRI pre-processing, beginning with the signal stabilization, removing the first 10 seconds of the acquisition. As the TR is equal to 2 seconds, the number of volumes removed was 5. Then the slice timing correction and the motion correction, using the MCFLIRT applying the rigid body alignment of each volume with the mean volume. After this was extracted the brain. In this process, first is calculated the again the mean image, obtained a mask to extract the brain using a variant of BET tool and then applied this mask was applied to all volumes resulting in the acquisition only with the brain. Afterwards was applied to the most recent version of functional acquisition a non-linear normalization rigid-body registration using the FLIRT and with the structural data as reference. Followed by other non-linear normalization from the native structural acquisition for the MNI (152) standard space and a resampling to 2 mm isotropic voxel size. The motion scrubbing and removal of the cofounding factors was the next step. In this was used a mask with the WM and CSF regions in order to calculate the mean values of this regions. Besides was search time-points that was corrupted due to much motion. Then using the General Linear Model (GLM) was did the linear regression of motion parameters for removed confounding factors. For last was done the band-pass temporal filtering for (0.01-0.08 HZ) of the residuals of the GLM application finishing the pre-processing.

The final result is a file with 355 lines, one for each of the volumes of the acquisition with the mean BOLD signal for each region of the brain. Note that in the pre-processing are the removal of 5 initial volumes to 360 volumes of the initial acquisition. Thus, it is ready for the next process, the construction of the correlation matrices.

4.2.2. HCP Dataset

The HCP dataset is a functional imaging data available by Human Connectome Project. This data is named by the organization as HCP100, because contains data from 100 different subjects. The HCP dataset used by Fin et al [19], named Q2 in the present did already not exist. This because, the HCP is continually gather more data, and eliminate some old data with worse quality. So, the Q2 and other Q datasets was unpacked and was a mixture of the different data (not eliminated) in the main dataset, the more recent is named HCP 1200 with 1113 subjects. Although existed yet some subjects from the Q2 data, they were not equal to number 126, used by Fin et al., and they were present in a reduced number for a study. Thus, it was decided to use the HCP100 for this work and possible future work in ICVS. The study responsible for the data acquisition was approved by the Institutional Review Board at Washington University in St. Louis and all participants provided written informed consent.

As mentioned the dataset is constituted by 100 unrelated subjects wherein 54 are female and 46 are male. The age given by the project was categorized nominal, and so for each category (time interval) the dispersion as it is in Table 4.2.

Table 4.2. - Subjects by age group.

Age Interval	**Total of subjects**
22-25	17
26-30	40
31-35	42
36+	1

The available data has two different resting state sessions, mentioned in the work as session 1 and session 2, acquire in two different and consecutive days.

The data download from the online HCP database was provided separately for session and for each one there was a folder by subject with MSM-Sulc and MSM-ALL registered versions of Resting State fMRI data pre-processed with the functional pipeline v3.1, the fMRIVolume and fMRISurface pipeline outputs, and the motion parameters [160]. The size of the session1 data is 594,15 GB and session2 594,4GB.

4.2.2.1. Magnetic Resonance Imaging Acquisition

All subjects and the two sessions were scanned in a customized Siemens Magnetom Connectome Skyra 3T at Washington University in St. Louis (WashU) with a 32-channel Siemens receive head coil. The acquisition was made in two days, and for each day it did 2 hour-long sessions that contains resting state and task conditions. In each day, first are scanned two resting state of 15 minutes long with the subject as relaxed as possible and was asked to maintain the eyes open and fixated in a crosshair. Two different acquisitions because they are acquired with opposite phase encoding directions, one left to right and vice versa. Then is acquired two sessions of 30 min of task-fMRI where are made the 7 tasks split by the two sessions [167];[168]. Each task also run two times because to be acquired in the opposite phase encoding. But in this work, it is used only the rs-fMRI acquisitions of the 100 subjects therefore only will be mentioned the parameters resting states images.

The BOLD signal was extracted using the gradient-echo EPI with the following parameters [168]:

- 72 slices
- TR/TE = 720/33.1 ms

- FA = 52°
- slice thickness = 2 mm
- voxel size = 2 x 2 x 2 mm^2
- FOV = 208x180 (RO x PE) mm
- Multiband factor = 8
- 1200 volumes
- Total acquisition time = 14 minutes and 33 seconds

4.2.2.2. Data Processing and Parcellation

The data download has already some of the processing done, more specifically all steps until the spatial smoothing. And as the In-House data the HCP data was normalized for the MNI (152) standard space. More, there was available the motion parameters from the motion correction and needed for the next process, the removal of the confounding factors. So, the pre-processing process began in this step, and using a mask with the WM and CSF region was calculated the mean value for each volume of this regions, besides, it was searched for motion outliers. In the final, the three files, one with the motion parameters, other with the motion outliers and with the mean values of WM and CSF was applied the General Linear Model (GLM) to get the BOLD signal without the cofounding factors. Finally, was applied a band-pass temporal filtering with the same band applied in the In-House data, so signals with frequencies in the range of 0.01 Hz and 0.08 Hz. Then is applied the several atlases applied in this work, using the atlas as mask and calculating the mean BOLD signal for each region and volume.

In the final of the process is obtained a file with 1200 rows and a number of columns equal to number of nodes of the parcellation used.

5. METHODS

In this chapter are describe all the different methods used to support and achieve the answer to the research problem and the different research questions. So, it comprehends all the technology, tools and work implemented to do it. All the different methods are summarized and schematized in the *Figure 5.1* as well the different materials and resources needed or produced by the methods. Thus, the principal parts of the *Figure 5.1* can be divided as resources needed or produced by the methods and the methods itself. Finishing with the analysis of the final results. In the first part are included the image resources, the other resources (module A and B, respectively) – Brain Atlases and MRI and fMRI acquisitions already described in the materials chapter. In the case of the methods, it comprehends the development environment, the processing, the datasets creation and to conclude the Deep Learning application (module D, E and G, respectively). There is also the different datasets with the functional connectivity values resulting from the modules datasets creation (module E) that will be used to feed the models in the Deep Learning application (module G).

The development environment is an essential part for computer program and software product development and can be described as the set of processes and, programming and technologies applied to create the program or a software product. Once this dissertation is about medical informatics based specially in computer science, this is a fundamental part, not to create a computer or software product but to support other methods in developing of the research work. The development environment as can be seen in the work architecture (*Figure 5.1*) was used since the data processing until the application of the Deep Learning module with the production of the best models with respective results. Thus, it was possible better automation of processes, easier and efficient data management and creation and a construction of a workflow to deal with the functional connectivity information and use it as features for Deep Learning in the construction of models for subject's classification tasks. The processing, as already mentioned, it's an indispensable method for extract reliable functional connectivity from rs-fMRI, in which is used a recent library Nipy based in the Python programming language with several advantages for the workflow. The datasets creation which includes the analysis of the values by dataset and subject or class in the dataset, is when is created all the materials, with or not with some feature design, preparing the final data to be ready to use in the training, validation and test of Deep Learning models. Finally, the major objective, the Deep Learning application over the functional connectivity to subject's classification tasks, which includes the fine-tuning of the models, the selection of the best models and the analysis and management of the different results.

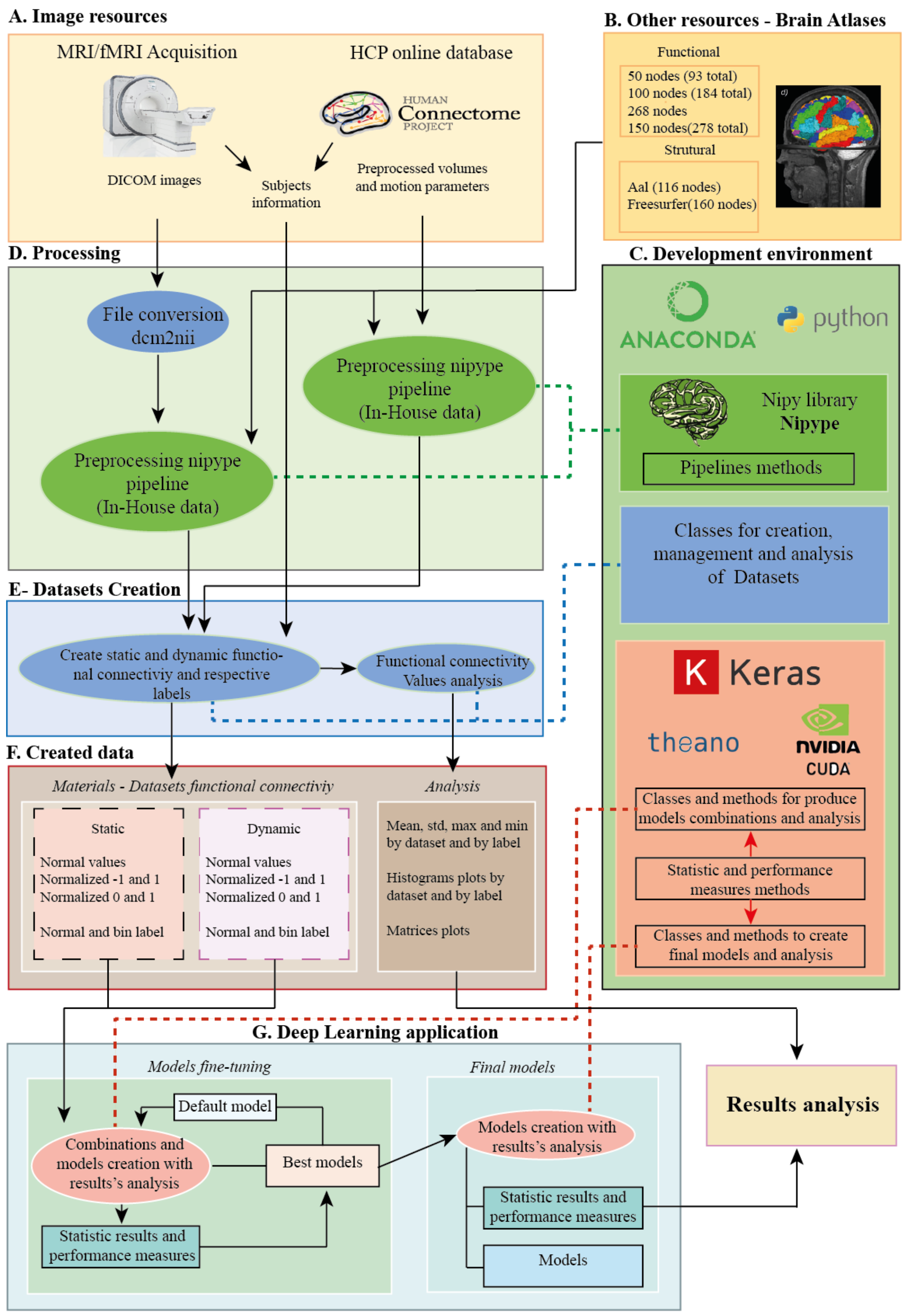

A. Image resources
MRI/fMRI Acquisition
HCP online database
HUMAN Connectome PROJECT
DICOM images
Subjects information
Preprocessed volumes and motion parameters
B. Other resources - Brain Atlases
Functional
50 nodes (93 total)
100 nodes (184 total)
268 nodes
150 nodes(278 total)
Strutural
Aal (116 nodes)
Freesurfer(160 nodes)
D. Processing
File conversion dcm2nii
Preprocessing nipype pipeline (In-House data)
Preprocessing nipype pipeline (In-House data)
C. Development environment
ANACONDA
python
Nipy library
Nipype
Pipelines methods
Classes for creation, management and analysis of Datasets
Keras
theano
NVIDIA CUDA
Classes and methods for produce models combinations and analysis
Statistic and performance measures methods
Classes and methods to create final models and analysis
E- Datasets Creation
Create static and dynamic functional connectiviy and respective labels
Functional connectivity Values analysis
F. Created data
Materials - Datasets functional connectiviy
Static
Normal values
Normalized -1 and 1
Normalized 0 and 1
Normal and bin label
Dynamic
Normal values
Normalized -1 and 1
Normalized 0 and 1
Normal and bin label
Analysis
Mean, std, max and min by dataset and by label
Histograms plots by dataset and by label
Matrices plots
G. Deep Learning application
Models fine-tuning
Default model
Combinations and models creation with results's analysis
Best models
Statistic results and performance measures
Final models
Models creation with results's analysis
Statistic results and performance measures
Models
Results analysis

Figure 5.1 – Schema of the materials and methods used in the dissertation.

5.1. Development Environment

The development and work of this dissertation was developed in two machines running in the open source Ubuntu operating system, a Linux distribution based in the Debian architecture. One of the machines was provided to pre-processing of data and development of Deep Learning models and other is my own machine. They will be named abstractly and respectively as university-PC and Home-PC.

Some specs of this machines, as CPU, GPU and Memory ram that are important to mention due its importance in the work development. In the *Table 5.1* are presented some information about this specs.

Table 5.1 - Summary of the hardware specifications of the computers used in this work.

Specs	**University-PC**	**Home-PC**
Machine type	Desktop	Notebook
Ubuntu version	14.04.5 LTS	16.04.3LTS
CPU	Intel(R) Xeon(R) E5-1650 v2 @ 3.50GHz	Intel(R) Core(TM) i7-4720HQ @ 2.60GHZ
Number of processors	12	8
Memory Ram	64 GB	8 GB
GPU	GeForce GTX 660	GeForce GTX 950M
GPU memory	2 GB	2 GB
GPU capacity	3.0	5.0

In addition, to compare the performance of the two-different pc's using the CPU and GPU was realized a set of five tests for each case and extracted the time of each one and calculated the mean and standard deviation values. The test applied the Theano library, and test the velocity in work with tensors. The results are shown in Appendix B – Theano GPU/CPU.

Table 5.2 – Mean times in seconds and the standard deviation values in the Theano Python test using the university-PC and the Home-PC

	CPU average time	**GPU average time**
University-PC	2.4300 ± 0.0020	0.3970 ± 0.0600
Home-PC	29.7000 ± 0.0870	0.3000 ± 0.0040

All development work was based in the Python programming language, an object-oriented, high-level and interpreted with dynamics semantics with several dynamic semantics for web and app development [169]. Also, it's a simple and easy to learn and the syntax implemented is very friendly, being focuses on the readability. So, it's easy to read and faster to implement any method or class, increasing the productivity. Besides Python supports the use of different modules and packages, so all the projects and code developed can be created in a modular style and later be easy reuse across different projects. Besides it is easy use modules built-in the Python environment or from external sources created for other people, and is always important to emphasize that is free to use. Besides the Python is one of the most widely programming language used world wield and the expectation is to continue to increase. This year for example, the Python in Github only stay behind JavaScript has in the list of the most used programming languages[170]. To download and manage the different packages used in this work was used anaconda, a free and open-source distribution of Python specially designed for data processing, predictive analysis and scientific computing applications[171]. Besides to support the development of the different parts of the work was used the PyDev, a Python Integrated Development Environment (IDE), for Eclipse, another IDE developed to support the development of Java. But great part of the work was developed and tested only by using the bash console or the interactive console of Python (iPython).

Moreover, this work used the CUDA, a parallel computing platform and a programming model developed by NVIDIA [172] for use graphical processing units (GPUs) in different computation tasks. It allows increase the computation velocity, so it's very useful for more heavy computations as is the case of the training Deep Learning models, where are so many parameters to update in each training epoch. A practical example can be seen in the *Table 5.2*, where the velocity increases approximately 6 and 100 times for the University-Pc and Home-PC respectively. So as can be understood the CUDA is a platform that will permit the use of GPU as computation unity in the Deep Learning application. The CUDA is not a default platform embedded in the system ready to use, so it is needed to install the required drivers. During the install is also provided an optional toolkit, which has different GPU-accelerated libraries, debugging and optimization tools that can be deployed in different applications.

So, until now, it was summarized some parts of the technology support used in the development of the different methods. Between them are the Ubuntu operating system that supports all the other software and tools to create the methods. In the next level is the Python programming language and Anaconda to support the Python packages management. So, two levels were already mentioned as can be seen in the *Figure 5.2*. Below these two levels are the principal packages used to create the different Python files in each method part, which includes: Nipype, scikit-learn, NumPy, Keras and Matplotlib

(*Figure 5.2*). Of course, many more packages were used in the development, specially belonging to the Python standard library, but this package set was essential to the development work in different ways. The five packages are third-party and open-source, downloaded and managed with anaconda that add other functionalities that the Python's extensive standard library not covers. Thus, it reduces the quantity of programming and besides offer different tools and applications already created. The packages are a form of collecting several modules within a single tree-like hierarchy. Each module is a simple Python file as the files created in this work for each method that contains Python definitions, functions and statements. These modules are very useful once they are easily imported as functional module to a Python script or to an interactive Python console (iPython). So, returning to the packages group they are different functions that can be summarized for each in the next points:

- **Matplotlib** [173] – It is a Python 2D plotting library that offers various types of plots in several formats with good quality. So, this package was very useful in creation of the plots of the data and results analysis.
- **Scikit-learn** [174] – It is a machine learning library that has different classification, regression and clustering algorithms, built with other packages used in this work such as NumPy and Matplotlib. But was the different data analysis and data mining tools provided in this package that was used to measure some parameters of the classifications results.
- **NumPy** [175] – It is a package that permit scientific computing with Python, and it's crucial for the work developed, once all the matrices and values with functional connectivity are saved in N-dimensional NumPy array and it made easier the different computations needed to do all long to work.
- **ipype** [176] – It is a Python package that has functions to help processing workflows with different neuroimaging analysis software that belongs to greater project, nipy, which has other packages to support neuroimaging processing. This package was essential in the creation of the different pipelines once all were created based in this library, so it will be further explored in the Processing Method point.
- **Keras** [177]– It is an neural network package that can run over other different backends such as CNTK, TensorFlow and Theano. This package gives different methods for rapidly construct a deep learning model and test. And its focuses on being minimal, modular and extensible. Keras also works with different models, with different complexity and can made the computation needed in CPU and GPU. This package also was fundamental to create the models, train, validate, test and save them in this work, so it will be further explored in the Deep Learning application point.

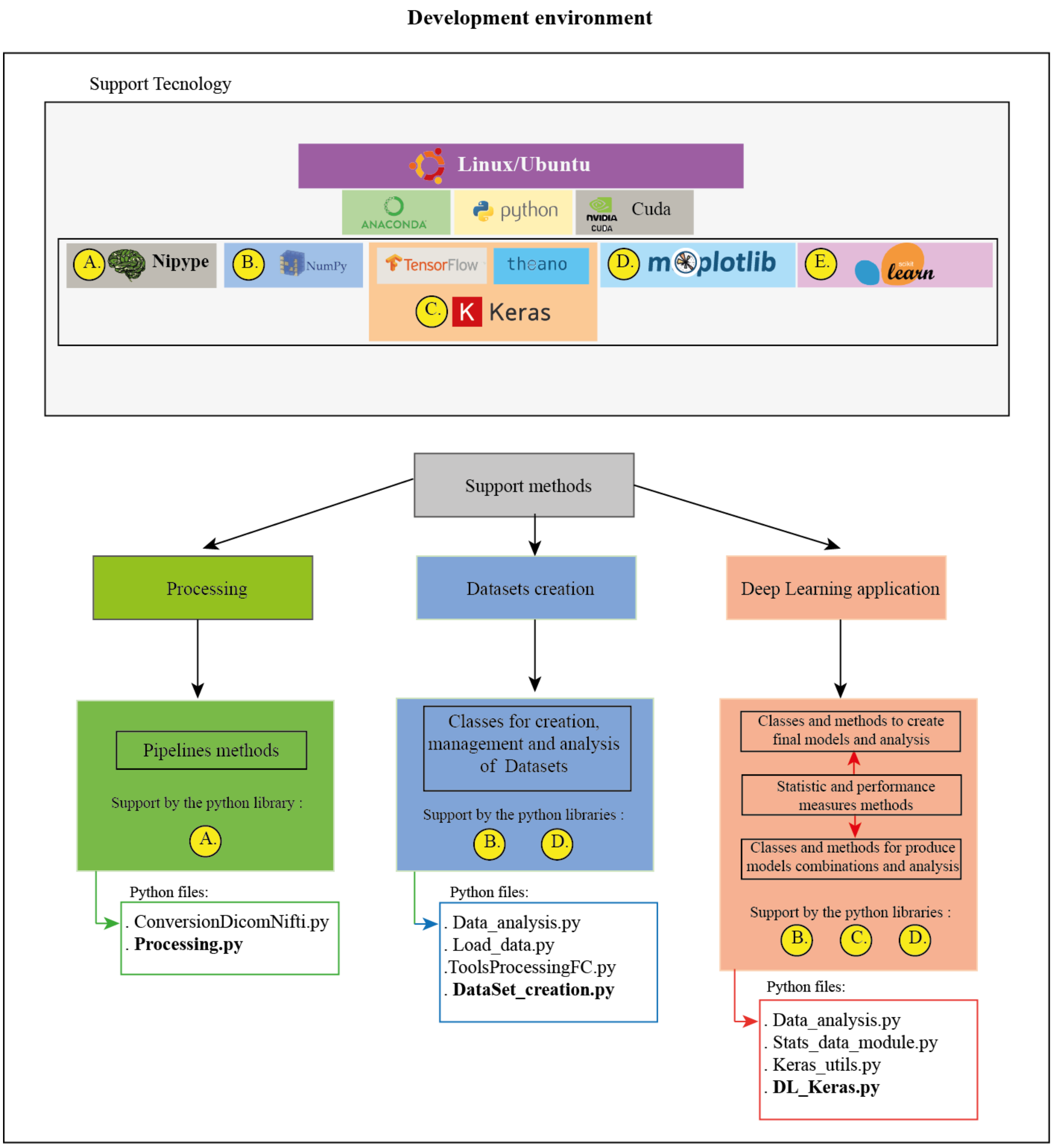


Figure 5.2 – Development environment schema with the support technology used in the development of the different methods, the method itself and their respective Python files created.

The different packages, as was described in each point, have specific functions and they were used during the development phase to support each part of the work, as represent in the *Figure 5.2*. With the different supportive packages, a modules set were created for each method with specific functions. But when the methods can be reused they are repeated in a method module part, since there is no need of repetitive work. So, a total of twelve modules were created with different functions, as is below:

- *ConversionDicomNifti* – It is necessary method to make the conversion of files from DICOM to Nifti format. It contains the functions to create the conversion configuration files, the methods to structure raw data and the conversion itself. This package then will be used by the processing module to make the conversion of the image data.

- *Processing* – This package has all the methods to create the final structures with the pre-processed data and the Nipype pipelines with the different workflows. This is also the main module responsible for the processing method, once they have the pipelines functions ready to use.
- *Data_analysis* – It has several methods to compute statistical values, create different types of analysis plots such as histograms or confusion matrices and to support the stats values management. This modules' analysis is specially used in analysis of the datasets created although one method is used in the analysis of the final modules in the Deep Learning application.
- *Load_data* – It has the methods to load the files with functional connectivity information.
- *ToolsProcessing* – It has the functions needed to create the static and dynamic functional connectivity data in different formats and other methods to get other information intended about the one or several functional connectivity values.
- *Dataset_creation* – It is the main module of the datasets creation method part and it contains the class used to create the static and dynamic functional connectivity datasets as well as several methods to support the creation and analysis of datasets. In this module are imported multiple functions from the *Data_analysis*, *Load_data* and *ToolsProcessing* modules.
- *Stats_data_module* – This module has the function applied to obtain the different analysis metrics about the classifications results.
- *Keras_utils* – It is a module created to specially to support the final main module *DL_Keras*. It contains different classes to save and manage models and results, methods to create the files with performance measures plus the different plots to help the results analysis. Once it need some functions to results analysis it imports the offer method offer by the *Stats_data_module*. Besides once this module and the *DL_Keras* shared different methods, this imports some functions created in the *DL_Keras*.
- *DL_Keras* - It is the module that implements the principal methods used in the Deep Learning application work and used widely functions from the *Keras_utils* module. But it also imported a function from the Data_analysis module. Summarized, this module contains the class of the final object created in the Deep Learning application and the respective functions needed to train the DL model and manage the different class attributes. Then it has the different methods to implement the different combinations tests and the respective results.

These modules will not be analysed in a code level but in each methods part, that includes the data processing, the functional connectivity datasets creation and the Deep learning application, some methods will be deeper explained due their importance to the application of the method.

5.2. Data Processing

The initial data obtained directly from the acquisition project and the Human Connectome Project needed some pre-processing steps to eliminate noise, interferences and cofounding values. So, it is important to improve the quality of the data and to prevent in the final false conclusions. Although the data pre-processing steps applied in the In-House and HCP data has already mentioned in the materials chapter, in this method will be described the approach to improve the processing workflow. This workflow is the sequential pre-processing steps made since the data in analysis until obtain the final data pre-processed or a specific analysis.

The pre-processing of fMRI is in general made with recourse to one or more neuroimaging designed software that apply different pre-processing steps to the images in the different stages of a workflow. Furthermore, they provide several tools to do statistical data analysis and unsupervised techniques. In this research field there are several software available and used in different research neuroimaging centres as mention in 2.4, such as FSL [90], Freesurfer [89] and Statistical Parametric Mapping [86]. The choice of the software will depend of the data format and quality, as well as of the final result intended and the type of the scheduled analysis. We used FSL, one of the most widely used in neuroimaging pre-processing. FSL is a comprehensive library of analysis tools for fMRI, MRI and DTI brain imaging data, designed for Linux and Mac OS, based in the Unix environment. Since these Operating System (OS), Linux and Mac OS, are the most common inside the Neuroimaging research the FSL becomes pointed out tool to be used. In 2011, an internet survey demonstrates where this two OS are used by over 70% in the neuroimaging researchers, being the GNU/Linux the most popular platform [178]. Additionally, the FSL also is a light software and offer a great miscellaneous of tools.

All the tools provided by FSL can be used by a command in the system console mentioning the different parameters of the intended process. It is an easy and fast way to work with a case study, but it becomes impractical when it's intended to do a set of pre-processing steps for a set of subjects. In order to solve the problem, the commands can be simple stacked in a bash script, doing the processes sequential and running one time for each subject. This bash script is a simple plain text file which contains a series of commands runnable in systems with Linux and Mac OS. But this is too rudimental taking a long time and a great deal of effort when the datasets are too large. Has been a tendency to use new approaches based in Machine Learning methods that increases the necessity of larger datasets. Large datasets get more reliable results. The pre-processing pipeline is a complex workflow that need more than one Neuroimaging software with different and specific functions. The pre-processing pipeline should be the most autonomous as possible to decrease the time consumption and the human work that indirectly decrease the human error associated to the normal mistakes or errors in the parametrization of the different methods. The parameters and the order in the workflow of the pre-processing steps should be standardized since will always be dependent of the research objective and work developed. Actually there are already some approaches, created and proposed by Paulo Marques et al.: the BrainCAT, a tool for an intuitive multimodal fMRI/DTI analysis using the graphical user interface (GUI) [179]. A more recent sophisticated approach is a pipeline for task-based fMRI that evaluate and optimized the different steps of the pre-processing, optimizing some data-driven metrics of task prediction and spatial reproducibility [180].

Relatively to this work, our main goal was to create a processing pipeline to support the pre-processing of resting-sates fMRI from the two sources. In addition, our last intention is to use it in the future in another image datasets, maintaining the reproducibility. Moreover, the design and the implementation were thought in order to be possible deal with large amount of data. We used 176 subjects and in the case of HCP there is two acquisition phases, so the pre-processing workflow was needed to perform over 500 times. Also, other important characteristics were considered during the construction and implementation of the pipeline, mostly relevant points already mentioned and explained previously:

- Automatization
- Robustness
- Possibility of customization of the pre-processing tools parameters
- Multi-processing
- Easy management of the data and results

From the points mentioned there is three yet not discusses. The robustness is the capacity of the pipeline deal with failures and the multi-processing is the capacity of the pipeline implement parallel computation. Relatively to the last point it is normal that all the data and results management should be easy and the most autonomous as possible to decrease the work time consumption and effort.

Although in this case the study is not multimodal since there is only the objective of study the functional connectivity from resting state fMRI. In the future, the pipeline should be adapted to other application, mostly because compatibility problems. Finally, and maybe the most important point, the pipeline should be implemented in the Python programming language, because all the development environment was created based in it.

In the search of a solution that checks all requirements pointed to the pipeline was found the Nipype (Neuroimaging in Python: Pipelines and Interfaces) software package tools [176]. This open-source community developed a Python-based software package as part of the Nipy project, which built an independent Python-based platform for the analysis of functional brain imaging data with different software packages for pre-processing, analysis, statistical analysis [181].

5.2.1. Nipype Package

The Nipype is Python-based, so all the pre-processing modules as well as their inputs and outputs are described in an object-oriented manner, easier to integrate in the development environment and other methods in this work. The flexibility in the interface of different software packages, since using the Nipype environment is possible integrate methods and algorithms of different packages (e.g., SPM, FSL, Freesurfer, AFNI, MRtrix, Slicer, ANTS) in the same workflow, becomes the design and management easily to work with. As the Nipype is an open-source, the different methods can be adapted and even optimized for a specific project. For all these reasons mentioned above it was used the Nipype in the study to pre-process rs-fMRI with the goal of compute the Functional Connectivity values.

The neuroimaging processing pipelines are implemented normally in Bash, MATLAB or Python, but Nipype implements the pipeline as a graph. Thus, it is easier to follow the pre-processing steps executed and the order of modifications done during the development in the pipeline (e.g., insert and remove Nodes). To give fine control of each step during the workflow is used an interface that function as a “wrapper” of underlying software. They are the core of the Nipype, which allow a uniform mechanism for use the tools of the different neuroimaging software and can be used as a Python object. Briefly, the creative process can be divided in four parts: first the inputs parameters, their types and dependencies, second the outputs and their types, third the way that will be executed the underlying software and last the mapping where is defined the outputs that are produced for a particular set of inputs. The Nipype package is already available with several interfaces for use tools of different Software packages, but is simple create new ones by a Python class. During the construction of the workflow, the interfaces have to be connected, therefore they are encapsulated as a Node or a MapNode objects. Only in that point they are functionally capable to deal with the different inputs and outputs. After all the interfaces be defined, with the inputs and outputs as well as the tools parameters are used by the workflow object. This workflow engine creates the pipeline by connecting the inputs and outputs of the interfaces, as a Directed Acyclic Graph (DAG). It is important to note that a workflow can be a node of another workflow. The fact that the workflow provides all information about the processing steps and the movement of data along the interfaces, being even possible save the workflow graph in four types of graphs and in a variety of file formats, which help to share the workflows between different neuroimaging centres and the reproducibility of articles or projects results.

Relatively to execution of the workflow it can be locally or on load-balanced grid-computing clusters through an extensible plug-in interface, and there is no need of changing the workflow to switch between the different executions modes. There is only needed change the argument of the plug-in to the mode desired. But in this work that the workflow was executed locally, the Nipype offer a great advantage. The fact of be able to execute the workflow in parallel using a local multi-processing plug-in that not requires any additional software. This saves a lot of time to the work and an unnecessary computation releasing the resources to other pre-processing or computations. It's even easier manage some hardware resources, since there is the option of select the number of processors used in the execution of the pipeline.

After analysis of this package we concluded that has advantages due to the fact be Python-based and to be easily integrated in the development environment, which make the Nipype a suitable package to use in the construction of the pipelines. These characteristics are summarized in the next points:

- Easily to integrate different software packages and combine them in the pipeline for different endings;
- Easier to create and reuse older workflows;
- Make the workflow easy to share and improve the data reproducibility;
- Faster pre-processing computation with the use of multi-processing using different cores/machines;
- The pre-processing checks the inputs and outputs of the workflow and in case of involuntary stoppage the process restart in the last step made.
- Open-source

It is important to highlight that the use of this package offer tools that are developer dependent, namely in the methods. It is possible to rise some problems with the updates, due the possible changes in the methods, in their names or in their variables. However, with Anaconda the downgrade is easy to perform in case of being necessary.

As conclusion, due to the different benefits found in this package, we used it in the development of the pipeline for the pre-processing, with the goal of create an automated and efficient pipeline. In addition, it will be obtained reproducible results that can be easier implemented by other user in the future, as an application to apply in pre-processing to study FC data.

5.2.2. Pipeline Design and Functioning

The principal pipeline was design for a full pre-processing from a raw format until to get the files used in the construction of the correlation matrices with functional connectivity values. The pre-processing steps contains processes done over MRI and fMRI data as mentioned in the Data conversion of In-House data in the Materials Chapter, as summarized in *Figure 5.3*, and described better in the

Appendix A – Processing rs-fMRI. The MRI final files resulted from the MRI data pre-processing will be necessary to improve the spatial normalization in the fMRI data. In the design of the pipeline all the pre-processing steps, management of inputs and outputs, was taken into account in order to have a pipeline automated and simple to be used and adjusted to this type of data.

In order to support and to make easier the management of different folders with the inputs, outputs and files produced was planned to create a configuration file for the customizer adjust for itself. Once the HCP project was some differences (e.g., two acquisition phases) comparatively to In-house data, different configuration files for each situation were created.

The raw format of an acquisition is the Digital Imaging and Communications in Medicine format (DICOM) wherein the acquisition is a set of DICOM images, a standard for distributing and viewing medical images in medical institutions. Because of their format, which are not compatible to computation in the pre-processing, the DICOM images are converter to NIFTI, to a single file named by volume. The NIFTI is based in the ANALYZE 7.5 format, a single file that contains a header to save meta-information and the data. The header has a size of 348 bytes, and contains useful information about the acquisition, such as the phase encoding, information such as the dimensions of the voxel – the unity of regular grid in 3-dimensional space, repetition time (TR) and others. The creation of this format also has the goal of increase the interoperability in the file-exchange between analysis software packages. The dcm2nii as conversion tool [182], since was the one with more and reliable meta-information in the header that can be useful during the processing. It is important to be careful with meta-information about the subject, because in case of inserted information being wrong also the DICOM files will be. In our case the acquisitions were anonymised, so this was not a problem. The conversion step was not inserted in the Nipype pipeline because it can be useful in the future for other applications. Thus, it was created the

ConversionDicomNifti module in Python with several functions in order to make the conversion more automatized as possible. Once again before initiate the conversion itself is produced a configuration file, where is possible to change the input and output folders as well as the name of the first DICOM MRI and FMRI image, as can be seen in the Appendix B – Development Environment in the Processing, configurations file. Before the execution of the conversion, the configuration file should be adjust as intended. The conversion method was constructed in order to convert FMRI and MRI data before the beginning of the major pre-processing pipeline. Additionally, it was created a method to structure the data before the conversion and anonymize the subjects during the process using information in a csv.

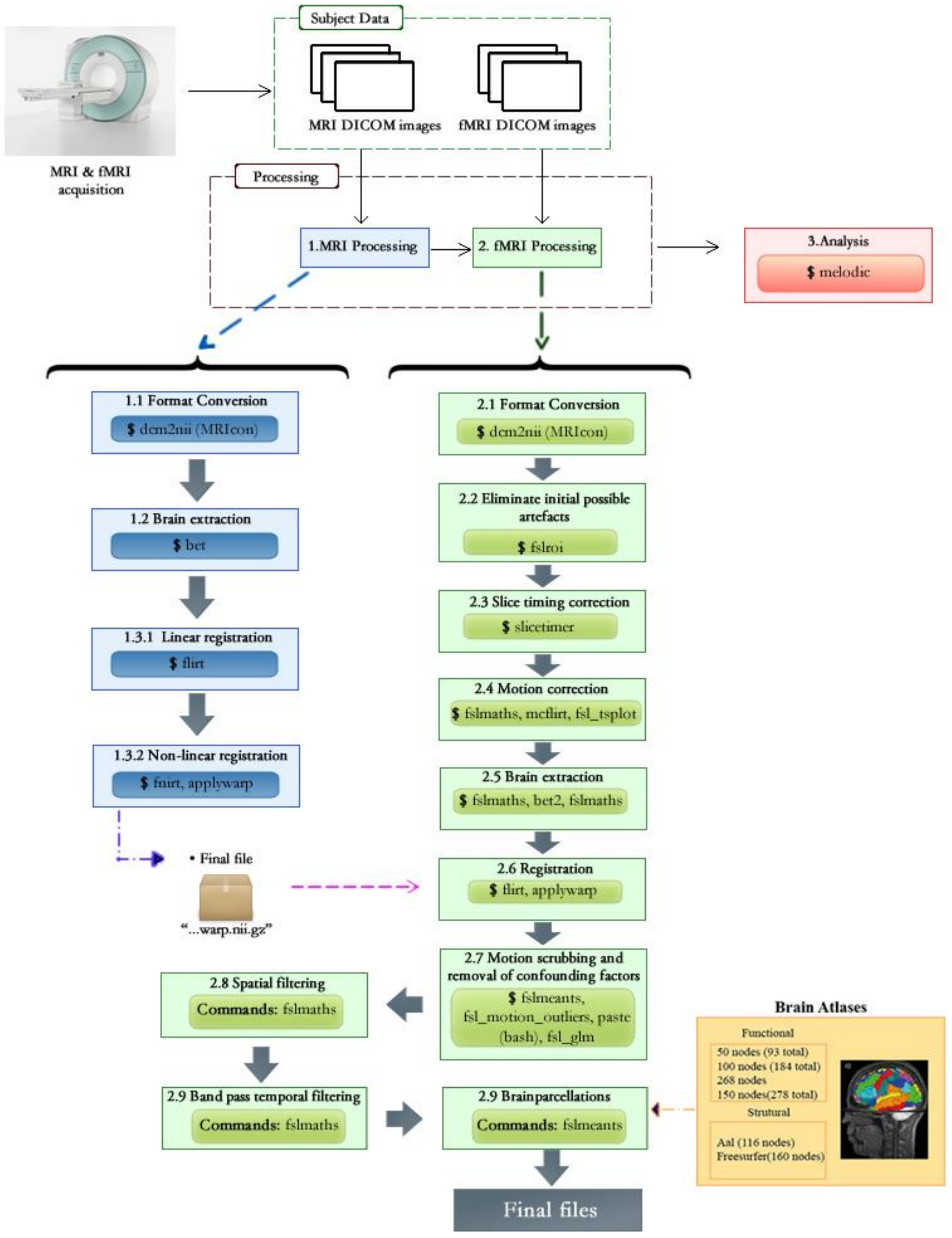


Figure 5.3- Illustration of all steps made in the processing of raw data, MRI and fMRI, until to obtain the final files of fMRI.

Relatively to pre-processing pipeline itself, the full process that englobes fMRI pre-processing and MRI pre-processing was divided in two major parts. The first part includes the format conversion of MRI DICOM images and the brain extraction using the “bet” command. And the second part contains all the other pre-processing steps until the final files ready to create the functional connectivity matrices. The reason why the first part of the extraction brain tool goes until the “bet” step is because it is necessary to control the brain extraction for each subject. Since all methods need a parameter (ratio) in the process to distinguish the brain tissue from the rest part of the brain that differ between subjects. This parameter has to be analyse and adjust in case of being necessary for each acquisition. The default value used in our lab is 0.2 but is often changed for lower values. Therefore, the first part is the conversion followed by the application of the bet command. After the extraction were done, using the default value, the results will be analysed and if necessary the ratio parameter will be adjusted and the extraction will be repeated. This process is repeated until the results are all verified. The adjustments in the ratio parameter are computed in a group not in an individual way, otherwise the multiprocessing advantage will not be taken.

The second part consists on all other steps after brain extraction, from the MRI pre-processing until to get the file with spatial information that will be used in the fMRI spatial filtering pre-processing. Despite of all these pre-processing steps already being detailed described in the materials chapter, it is important refer some resources. One is the mask of white matter and cerebrospinal fluid (CSF) used in the motion scrubbing and removal of the cofounding factors. It is crucial, before the pre-processing, to check if the file is available in the base directory of the running pipeline. In case of not being, the file should be added to the directory. In the future this resource should be online and downloaded automatically whenever necessary. The other resource is a file with the order of the slice acquisition that is used in the slice-timing step. But this file is created automatically by a method that get the information from the NIFTI header file that has the slice acquisition information. This field, has several meta-information, since has the information about the start and the end of volume slice and the timing order. This information is given by a code in the “slice_code” Nifti header since the “slice_dim” be different than zero. The codes variate between 0 and 6, and are interpreted as is below:

- Code 0 - Slice order unknown;
- Code 1 - Sequential, increasing;
- Code 2 - Sequential, decreasing;
- Code 3 - Interleaved, increasing, starting at the 1st MRI slice;
- Code 4 - Interleaved, decreasing, starting at the last MRI slice;
- Code 5 - Interleaved, increasing, starting at the 2nd MRI slice;

- Code 6 - Interleaved, decreasing, starting at one before the last MRI slice.

During the development of the pipeline we verified that the module misc in the algorithms folder of the Nipype package has some troubles in the MergeCSVFiles method, a Nipype method for merge files. The problem occurs in the application of this method to merge three different files (motion parameters, motion outliers and the mean values of GM and CSF regions) in the Motion scrubbing and in the removal of confounding factors step, where in the method could not open the files and concatenate the different columns to create a file with all the columns. This method was modified to solve the problem, as can be seen in the Appendix B – Development Environment in the Processing, misc module. The pipeline constructed was formed by a sequential of pre-processing steps described as nodes in the Nipype pipeline. The different parameters used in each node was similar to example in the

Appendix A – Processing rs-fMRI.

The results were created in a base directory folder that contains the mask of the white and CSF brain tissue and where was created the configuration files, the "config_conversion.txt" and "Processing_configuration.txt". This folder contains the "Data" folder with the DICOM and, after the conversion, the NIFTI files of MRI and fMRI acquisition for each subject.

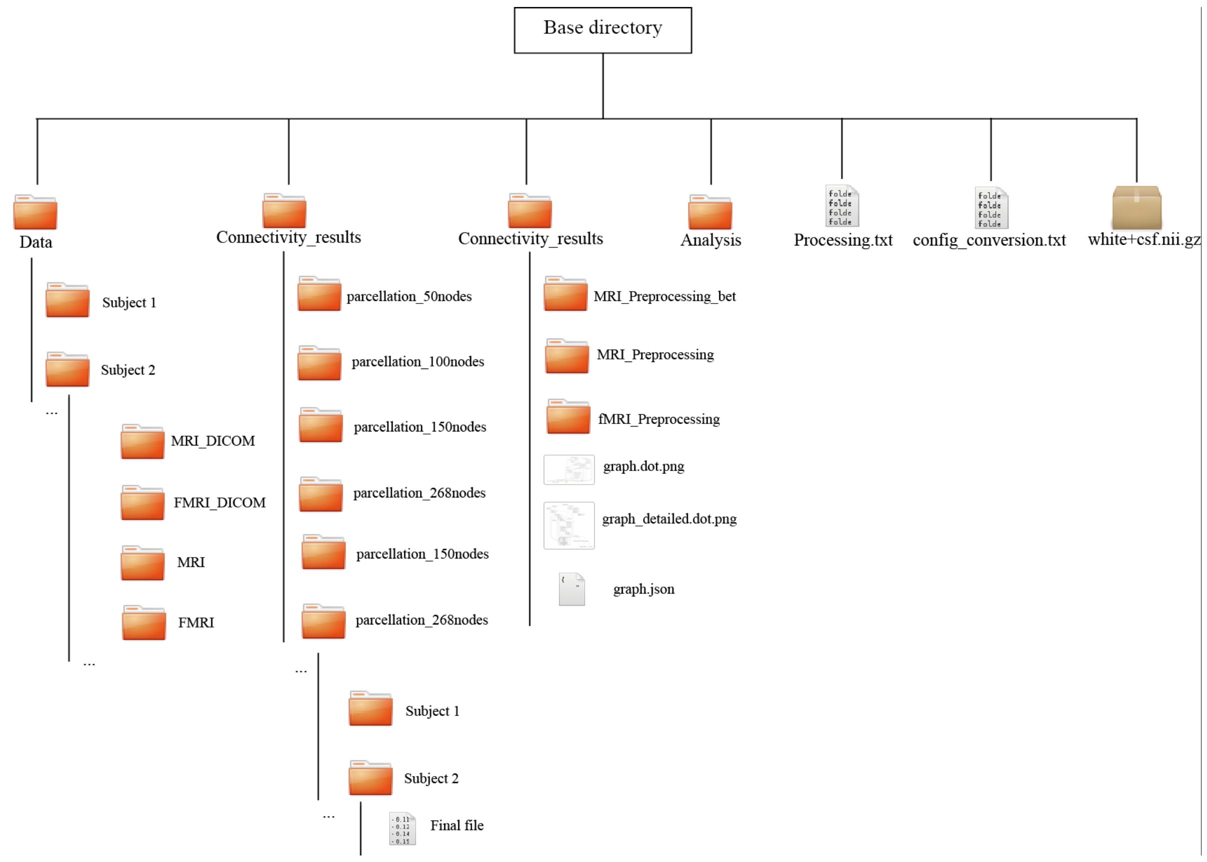


Figure 5.4 – Schema of the base directory after the application of the processing pipeline.

The folder with the processing contains a folder for each part that formed the full pipeline, which includes the steps made until the brain extraction (folder "MRI_Pre-processing_bet"), the pre-processing made after the bet extraction until the final of MRI data (folder "MRI_Pre-processing"), the final pre-processing steps applied to the fMRI data (folder "fMRI_Pre-processing") and then the analysis folder. Each of this folder contains a folder for subject with the steps made in each part. Besides that, the Nipype creates other files, such as the graph of the pipeline in a json file, and two images with the schema of the pipeline wherein one offer more details than the other. The folder "Connectivity_results" contains all the files for create the functional connectivity matrices for each parcellation used then for each subject. Then if the analysis is activated, there is created the folder "Analysis_results" with the respective results for each subject. Thus, it's created a structure as is summarized in *Figure 5.4*.

5.2.3. Pipelines Created

The principal pipeline created was used in the full processing of MRI and fMRI data, however other pipelines were created. These pipelines are similar to the main pipeline, because they share the same order. The main difference between them is in the beginning step, since the data resources are in different pre-processing stages. But it was also created some variations in the pipeline in order to analyse and test using the Deep Learning.

Thus, the different pipelines created in the module Processing was the following:

- The full processing pipeline – ***Processing*** – This method creates a full pipeline as above describes, since the raw data MRI and FMRI DICOM images until get the final objective files. The inputs are the base directory. There is an optional list where can be specified a subject group in case of not being necessary to do the processing to all subjects. Also, an option for spatial filtering step, since some acquisitions don't be needed and an option for the analysis process.
- HCP processing pipeline – ***HCP_Processing*** – This pipeline was created specifically to HCP data, which consist in some specific steps. This pipeline also presents a relatively variation to the main pipeline, since its possible separate the full acquisition for each subject in two parts where the percentage of each part could be chosen. This division is made before the temporal filtering since in this step there is a normalization of the signal in all the acquisition. So, if the division process was after the temporal filtering the objective was lost, once they were related by an average value that appear in the two data parts. This division was applied due the lack of data in the Deep Learning application to correctly validate and test the models. The inputs are the base

directory where is the folder contains the data images, the option of applying or not the spatial filtering, the option of made the melodic analysis, the input to select the number of cores used in the computation and then the two inputs responsible for the division of the acquisition. Also, there are an option to activate the division as well as to choose the percentage division.

- Two temporal filtering pipeline – ***pre_temporal_filtering_pipeline*** – This pipeline does the same steps as the main pipeline but only begins the pre-processing process before the temporal filtering. The pipeline was developed also for the In-House data and as before, it can made the acquisition division in two parts before the temporal filtering. The reasons are the same as the HCP pipeline. The inputs are also similar to the HCP pipeline, but in this there is also possible delimited a subject group that the pipeline will use instead of using all the subjects.

5.3. Datasets Creation

This method is responsible to create all the datasets, which are constituted by two major parts: the data where is the functional connectivity values and the respective labels. The data are the structures that save the functional connectivity value of each subjects. And the labels are a class that save some information about the subject that can be categorical, ordinal, integer-value or real-value. This can be the own identity information of the subject, the gender, the state in a disease, the blood type and others. The label is the information that will be used in the supervised learning for training, validation and testing Deep Learning models.

The Functional Connectivity (FC) was analysed through the correlation matrix, or the ROIs FC analysis method, as scientifically named. The FC features were produced under matrices form from the different files produced after the brain parcellation step. Each file has one line for each volume in the acquisition and one column for each region of the atlas used in the parcellation process (e.g. the average value for each volume and region). Therefore, the correlation matrices size is correlated to the parcellation used, which in turn affects the number of the total values present in the matrix (*Table 5.3*). In the *Table 5.3* are the values of the upper or lower triangular of the matrix, corresponding the values non-redundant and the values not present in the diagonal. Because the matrices are symmetric, and the diagonal values are the correlation values of the regions with themselves, they are not useful. In this work was used two different approaches to extract two functional connectivity types, the static and the dynamic.

Table 5.3. Features of the correlation matrices by each brain parcellation.

Atlas applied	Matrix Size	Number of values	Features
150 nodes	278 x 278	77284	38503
268 nodes	268 x 268	71284	35778
100 nodes	184 x 184	33856	16836
Freesurfer	160 x 160	25600	12720
Aal	116 x 116	14456	6670
50 nodes	93 x 93	8649	4278

5.3.1. Static Functional Connectivity

The static FC was computed through the Pearson's correlation between the full time-series of each pair of brain regions, resulting in the symmetric matrix with the Pearson's coefficients and one adjacency matrix per subject [10];[184]. The Pearson correlation coefficient between two series X and Y of size N is given by the Equation (13):

$$\rho_{X,Y} = \frac{\sum_{n=1}^{N}(X_n - \bar{X})(X_n - \bar{Y})}{\sqrt{\sum_{n=1}^{N}(X_n - \bar{X})^2}\sqrt{\sum_{n=1}^{N}(X_n - \bar{Y})^2}} \quad (13)$$

This method is widely used in this field, and allow to infer the strength of functional connectivity estimating the linear correlation coefficient between two temporal signals. If the regions have the same behaviour, they are activated and deactivated at the same time, the Pearson's value will be high demonstrating that it is probable that the regions have a functional connection.

Then a Fisher's r-to-Z transformation was applied to each correlation matrix to improve the normality of the correlation coefficients. Thus, the result is a correlation matrix by subject.

5.3.2. Dynamic Functional Connectivity

The dynamic Functional Connectivity (dFC) was calculated using the BOLD Phase Coherence Connectivity approach [63];[64]. But before the measurement of the phase coherence was needed estimate the phase of the time-series of each region for each repetition time (TR) corresponding each volume. So, it was applied the Hilbert transform to get the phases of the BOLD signal. Then, it was computed the phase coherence between each pair of the brain areas. Therefore, the $dFC(r1, r2, t)$ for regions $r1$ and $r2$ at time t was obtained applying the Equation (14):

$$dFC(r1, r2, t) = \cos(\theta(r1, t) - \theta(r2, t)) \tag{14}$$

Using this equation, when the two areas $r1$ and $r2$ have temporally similar phase meaning that the bold signals are align, the result is $cos(0) = 1$ as the dynamic Functional Connectivity. In other hand if the signals are orthogonal, this is the difference between phases is approximately 90°, the dFC will be practically null. In this work the temporal analysis is done for each repetition time, so for each volume of the acquisition result matrix with the dynamic FC information.

5.3.3. Python Modules

To support the datasets creation as well as their analysis was created a set of Python modules with different classes and functions. In all, four modules were created: *Load_data, ToolsProcessingFC*, *DataSet_creation* and *Data_analysis*.

The *Load_data* module is a little module with some functions related to get the files that will be used in the dataset creation.

The *ToolsProcessingFC* module has the methods to create the correlation matrices from the resulting files of the pre-processing pipelines and several methods to support the datasets creation. Wherein the static correlation matrices were created by using the Pearson's correlation and Fisher's transformation of r to z, and dynamic by the BOLD Phase Coherence Connectivity approach [63];[64]. The module also has the methods to extract only the information non-redundant from the matrices.

The *DataSet_creation* is the main module once in it are the major methods that create the datasets and analyse them using functions from the other modules. It contains the class "Dataset_FC", class used to manage the diverse Datasets as objects with several attributes with meta-information or data. The attributes used to save some meta-information about the dataset was:

- FC_type – Identify the type of functional connectivity of the dataset, dynamic or static;
- name – The name of the dataset;
- path_location - The directory where was created the dataset;
- size_data – The number of examples present in dataset; each dataset is a correlation matric;
- size_row_data – The number of inputs for the data in array form with the non-redundant values;
- size_row_label – The number of data labels examples existent in the dataset, which should be equal to the size_row_data.
- classtype - The type of the label, if it's a string, an integer or a float;
- number_nodes – The number of nodes of the data used in the creation of the data;

- dataset_label_disc – An optional description about the label used in the dataset;
- info_dic_label – A dictionary that contains as key thee real information extracted from the csv file and the corresponding final label assigned during the dataset creation. For example, in the case of the subject id, the information about the identity extracted from the csv file is transformed in a label depending in the number of subjects that already exist saves. This dictionary besides have information useful to know which label corresponds to the real value, it is useful in the creation of the datasets to know if the new label added to the dataset already exist;
- dic_brain_spatial_information – It's a dictionary used when is introduced information about the nodes wherein the keys are the information, and the values are list of the modules that share that information;
- nodes_description – It's a dictionary that is used when is added information about the nodes of the parcellation that had in the origin of the creation of the correlation matrices. So, the key is the identity of the node, and the value is the information.

The attributes with meta-information are important since some the information saved about the dataset object is relevant to manage the object, such as the name, the base directory or the functional connectivity type used to create the dataset. Besides it contains other information as the spatial information essential to the adjustment of the matrices by nodes regions. Relatively to attributes with data information was:

- joint_data – The array with only non-redundant values is saved with the respective labels wherein the shape is equal to number of examples and each example is a tuple where first part is the data and second the label;
- data – There is saved the data with only non-redundant values for each subject, so is an array of examples;
- full_matrix_data – The data is saved a full correlation matrix for each subject resulting in an array of matrices;
- full_matrix_data_structured – It has the correlation matrix as "full_matrix data" but now the correlation was done respecting a spatial information, the nodes were associated in group with similar function or belonging to a same brain region;
- normal_label – It contains the labels as was extracted from the csv file and it can be a string, an integer or a float;
- bin_label – This is other form to describe the labels very useful for classification where the classes are transform in binary arrays. A binary array with the same size of the different number of

existent classes, where the each of element corresponds a one class. Thus, for each binary array there is an element that is one, corresponding to a class, and all the other elements are zero.

Besides the different types of data saved in the object, the object also provides methods to get the data normalized between 0 and 1, and normalized between -1 and 1, for each data type with the exception of course of data related to labels information. Once this object provides so many data hypothesis it raises the question "Why are the data saved in so many different ways?".

The reason is directly related to the fact of this data will be used in the Deep Learning application to test if different approaches can or not change the model performance. The two library packages used in the development of the Deep Learning models also have a role in the arrangement of data. For example, the data used in the first models developed only with the Theano package was a tuple wherein the first element was the data and the second was the label. So, in case of the data has 200 examples, the shape will be (200,2). Consequently, was created a type of data to this end and was saved in the "joint_data" attribute. This data is also used to test the consistency of the data created and retrieved, e.g. if the labels corresponding correctly to data, since the tuple created never changes.

Relatively to the other attributes data, they were all designed to be applied in models developed with the Keras package. There are two major arrangements of the data: one is the data that will be used as input of non-linearity layers under a form of array, and the other type of data will be used as input for convolutional layers under a form of matrix. Then, for each situation, was done some processing to the FC values, such as the normal normalization and the normalization between -1 and 1. Additionally for the convolutional layer data was also created matrices with spatial information and the respective normalizations. These matrices are a little different from the normal matrices since they have the nodes organized by regions that were put together using functional or structural information. Therefore, the values inside these regions have somehow spatial information that could help the convolutional layers to find some features in the data. Thus, depending in the package and the approached used, the data can be organized as in the *Figure 5.5*.

The different arrangements of the data, with the exception of the normalized, were saved in the dataset object attributes to save time and reduce the computation required, due the data dimensions. For example, the smaller dataset for the In-House data using the static functional connectivity and the 50 nodes parcellation (93 nodes) has a total of 325128 values. And after this only gets bigger, when it's used atlas with more nodes in the parcellation or when is used the HCP data or used the dynamic functional connectivity, resulting in most of the situation billions of values. Every time it is necessary this type of data, it will be required a great computational effort and will take too much time (some minutes, as opposed to the expected seconds). In other hand, the object size saved is very large, and the costs associated to storage of the data are higher. The normalization values were not saved since they will increase several times the final size of the object.

To help the data retrieve from the *DataSet_creation* module has a set of methods where is possible to choose the type of the data and the label (in normal or in binary form). And the given data is ready to feed the Deep Learning models.

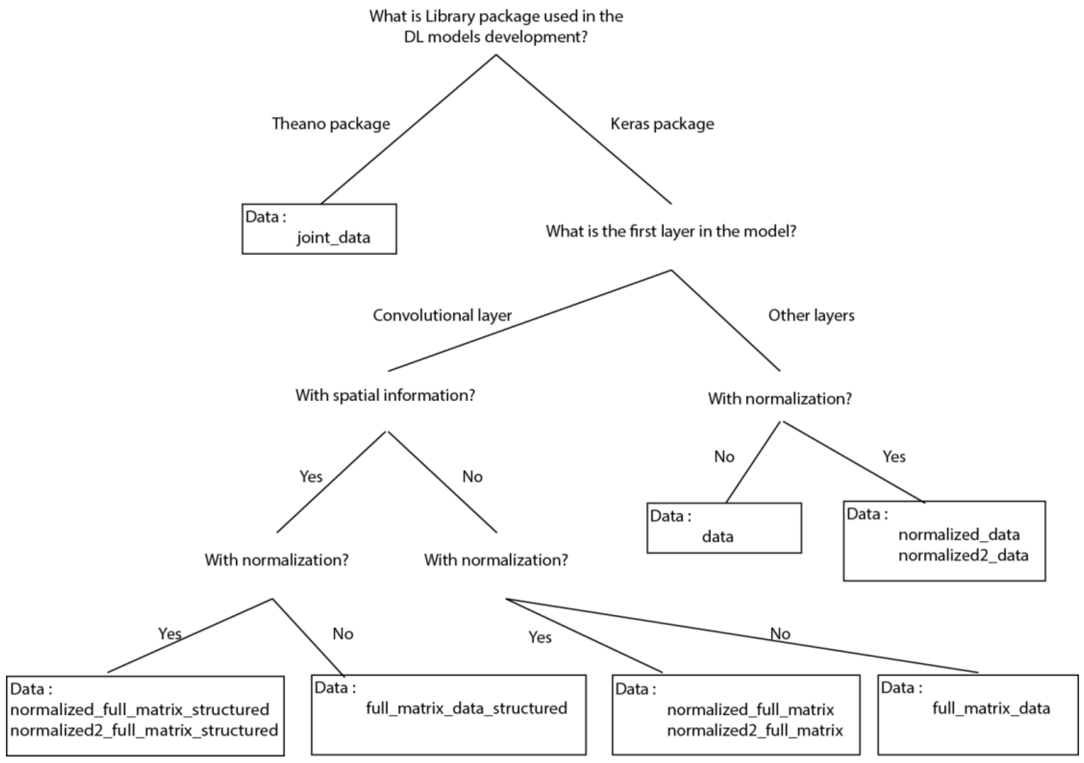

What is Library package used in the DL models development?
Theano package
Keras package
Data : joint_data
What is the first layer in the model?
Convolutional layer
Other layers
With spatial information?
With normalization?
Yes
No
Data : data
Data : normalized_data normalized2_data
Yes
No
With normalization?
With normalization?
Yes
No
Yes
No
Data : normalized_full_matrix_structured normalized2_full_matrix_structured
Data : full_matrix_data_structured
Data : normalized_full_matrix normalized2_full_matrix
Data : full_matrix_data

Figure 5.5 – The different types of data saved in the attributes of the object DataSet_FC, organized by package used in the DL models development, the first layer in the model, spatial information and if have normalization or not.

About "DataSet_FC" class still need to be mentioned how is performed the creation process and the respective analysis procedures of the produced data. To create a dataset object there is only needed mentioned the name of the dataset, the path location where will be created the object and the functional connectivity type that will be used in the dataset (e.g. static or dynamic). An example could be seen in the *Figure 5.6*, where the name is "76sub_static_sub", the location path "HCP/100sub/Rest1/50nodes/static", and the FC type "static".

```
dataset = dataset_FC("76sub_static_sub","HCP/100sub/Rest1/50nodes/static","static")
```

Figure 5.6 – Python example to creation of a dataset object

The created object has no data saved, only has some meta-information so it is needed to add data. To do it the method that adds the data has to identify where are the files with the mean time-series for each acquisition volume and brain region as well as the file with subject's information used to create the labels of dataset. So firstly, it is required identify the base directory where is the pre-processing folder with all the subjects time-series files and with the same Nipype pipelines methods used, in order to be easy, the data exchange between the different methods parts in the work. Then, there is the hypothesis to use the information saved in the configuration files or to use the interactive mode to descend over the different folders until arrive to the parcellations folders with the target files. Next it is chosen the wanted parcellation and the files from all or a limited number of files. After, the labels will be selected, which are in .csv or .xslv files in the base directory, to extract the label information. Continuing the process, it will show a menu giving two hypotheses: one is the extraction of the label, with the conversion for integer or float if possible and, in last case, as a string. Secondly, there is the hypothesis of add some processing to the information in the csv and only before extract the final labels. In this work was created three possible situations, the first is the incremental id, wherein for each subject id is given a number beginning in zero and then incremented by one. The second is for the sex, where the "M" and "F" is replace by "0" and "1", and the last situation is to calculate the age that used the birth date in the file to calculate the age. This method needs be informed of the date format birth. But regardless of each hypothesis choose since the beginning, it is always necessary to identify the subject id to save or to use in the processing process. The header information is important but not vital once if it do not exist the columns index can be chosen manually. As soon as all the necessary information is selected, the correlation matrices are created using the ToolsProcessingFC module. A different type of data is created (normal and binary labels), finishing with the save process of all the information in the object file. As soon as the object file is saved, proceeds to analysis methods. In the different analysis processes is used the Data_analysis module that contains all the methods to do the analysis and visualization of the values. The method that adds data can be used as many times as needed, since the files used to create the correlation matrices were created by the same parcellation atlas.

The analysis process is done in two parts, the first is the computation of the different statistics values measurements and the other is related to the data visualization, (how the values are distributed for label or datatype or both). The method creates a "Stats" folder in the base directory that contains the "Matrices" and the "Values" folders. In the first folder there is one folder for each type of data, normal, normalized, with spatial information and others, wherein are saved all the plots of the matrices examples present in that data. An example of a matrix with static functional connectivity is showed in the *Figure 5.7*.

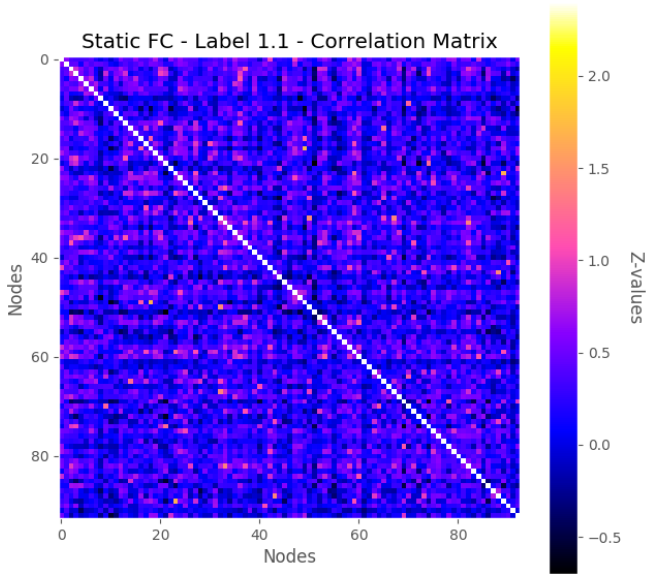


Figure 5.7 – A correlation matrix for 50 nodes parcellation (93 in total) with the values of the static functional connectivity of the label 1.

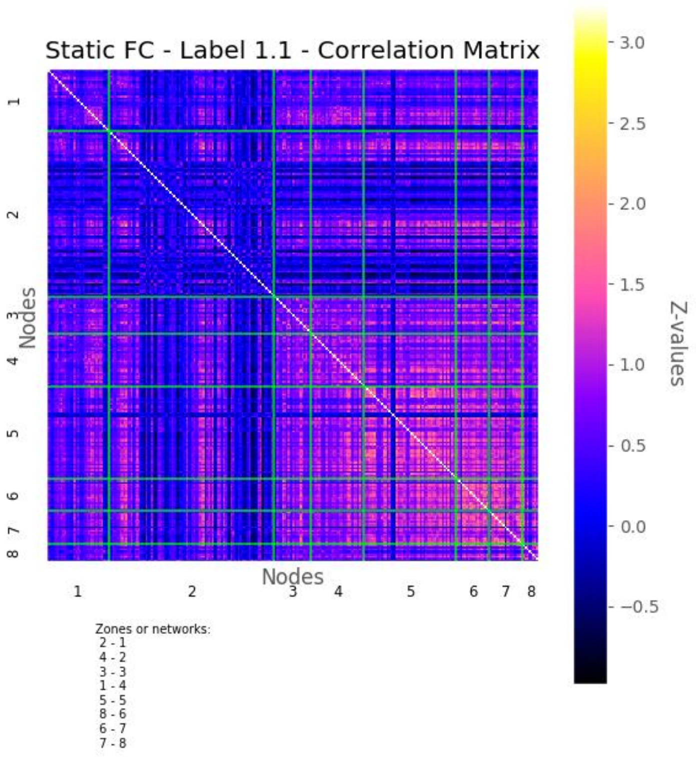


Figure 5.8 -– A correlation matrix for 268 nodes parcellation with the values of the static functional connectivity of the label 1 with spatial information.

To help to understand the different correlation values they are displayed in a colour map. During the matrices creation, the diagonal values in the transformation r to z are converted in an infinite number, in order to be possible plot the matrix. Besides to help the visualization, the plot title and the file name has the information about the type of functional connectivity showed as well as the corresponding label followed by the matrix number for that label. In the case of the *Figure 5.7* is "1.1", so label 1 and the matrix 1 for this label, since can exist more than one matrix for label. The plot of the matrices with spatial information has a little different presentation, *Figure 5.8*. The nodes are grouped and placed in a group, where each group (left side of the subtitle) represents a functional network or a brain zone (right side of the legend). Each brain region was delimited by lines, in order to identify the pattern.

The "Values" folder has the files with statistics measurements, such as mean, standard deviation, minimum and maximum, and the plots with the values distribution. The analysis in this case was only made over the non-redundant data, since there was not necessity of analysis over repeated and matrix diagonal values. The analysis process provides always the file "Values data stats.csv" that contains the statistics measurements of each different data type. Despite of being created a file for each data type, the statistics measurements is not done in a global but at label level. The name of the file is "Stats values by label_" concatenate with data type. Relatively to plots, is created a histogram for each data type where is possible to see the frequency distribution of values for 0.01 intervals (but can be changed). Furthermore, in the plot it is showed the lines of average values as well as the first and second standard deviations intervals, as we can see in the *Figure 5.9*. The same plot is done for each label of each data type, which are disposed in a folder with the name of data type in study inside the "Analyse by label" folder. As happened with the matrices plots, in the name is described the type of functional connectivity of the data with the respective label number.

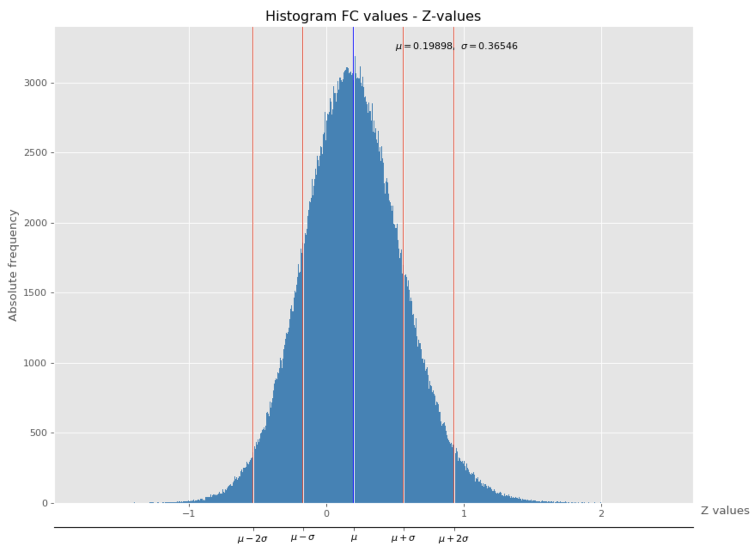


Figure 5.9 –Plot of the histogram of values for the 50 nodes parcellation and normal data.

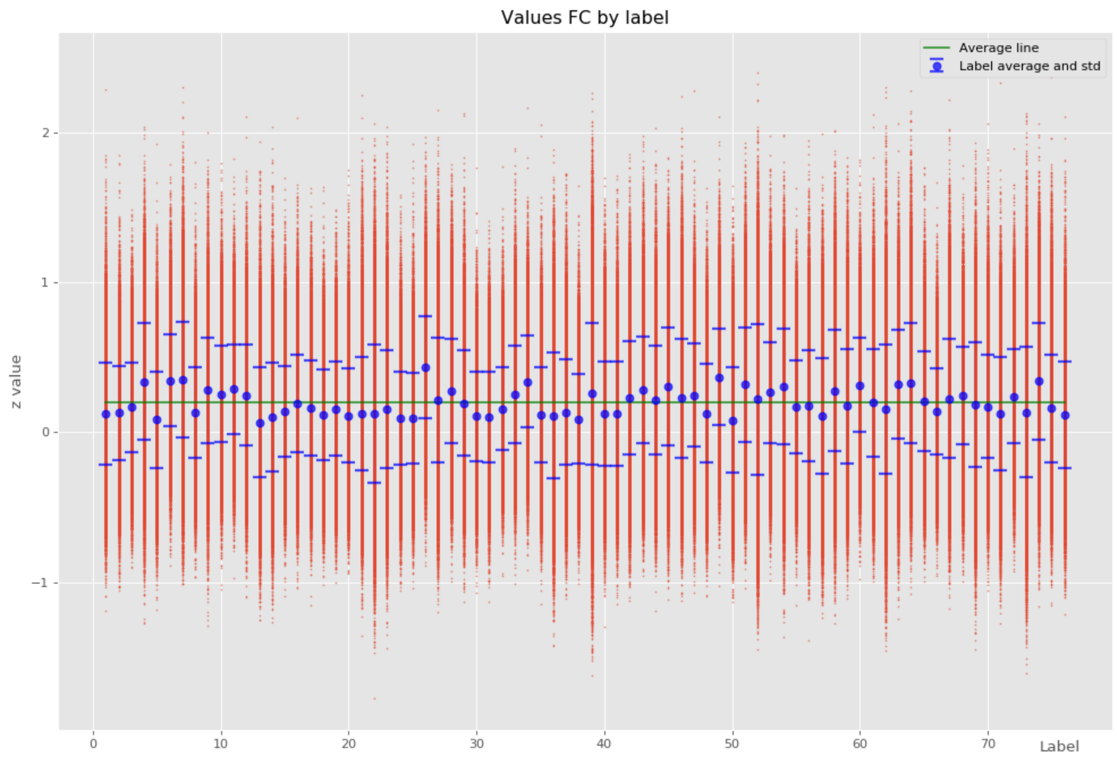


Figure 5.10 – Plot of the all the values by each label for normal data using the 50 nodes parcellation.

In addition, it is created a plot where is showed all the values by label for a specific data type in order to rapidly compare the values and to find the statistical significant differences. Also, it is presented the mean, the standard deviation and the overall mean, as showed in *Figure 5.10*.

Briefly, the datasets are represented by the DataSet_FC object where some attributes are saved as meta-information and some with different types of data. Different DL models' approaches were done, in order to represent the data in a simple way. The analysis part is essential to visualize the values and how they variate, which can be useful to explain the final results obtained by the Deep Learning models tests.

5.4. Created Data - Functional Connectivity

This module part as represented in the *Figure 5.1*, it is not properly a method but a group of materials that will be used to train, validate and test Deep Learning models. It comprehends all the datasets produced using the methods and modules already mentioned in the previous method. It was organized as is represented in the *Figure 5.11*, where the data is divided according to the source (In-House data or HCP data), the acquisition session (session 1 or session 2), the parcellation used in pre-processing and the type of functional connectivity (static or dynamic). Then, for each of these possibilities, the data can be created from the resulting files of the two different approaches used in the Nipype pipelines. As the acquisition is divided in two parts, the first produces final files set and the other produces two final files set. So, for each possibility it is created three different datasets using the same label information: first one has all the information about the acquisition, while the other two have the information part of the global acquisition.

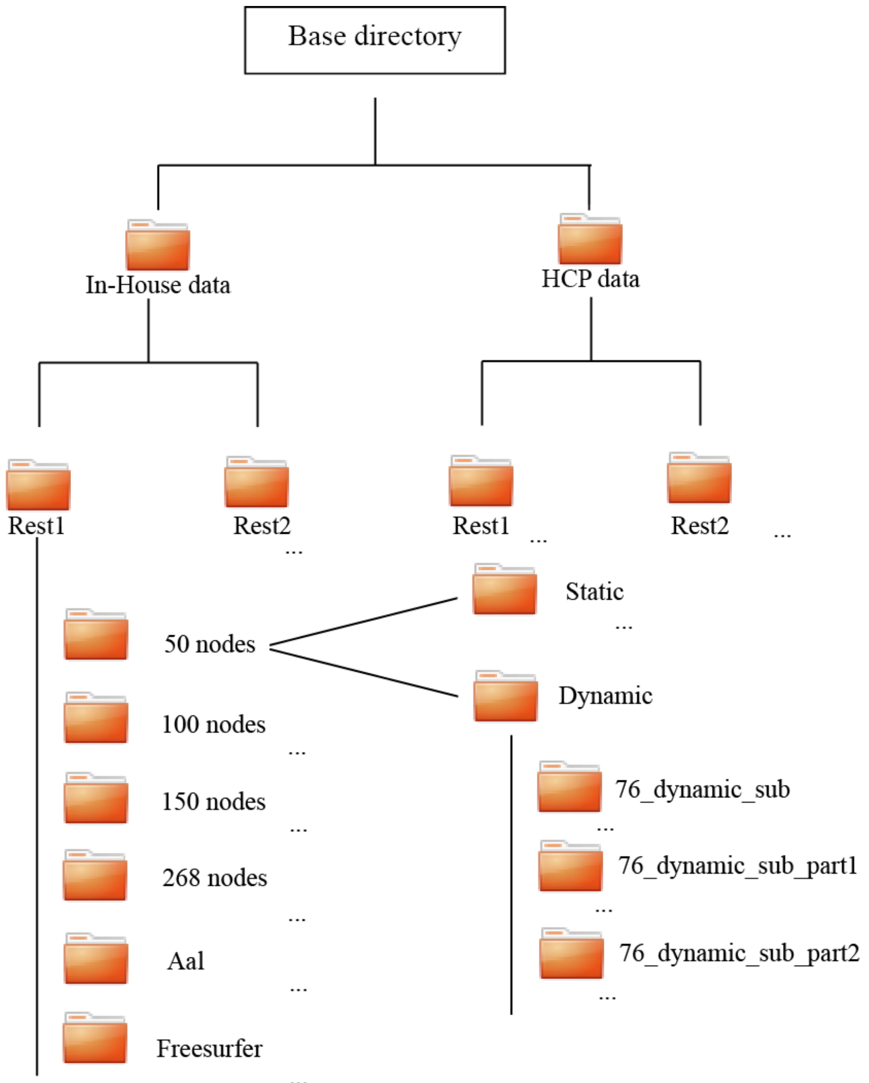


Figure 5.11 – Structure of the created datasets.

The functional connectivity data was deeper analysed for each source, functional connectivity, session and parcellation. It is important to note that statistical metrics only was made over the non-redundant values and without the values in the matrix diagonal.

5.4.1. In-House Data

5.4.1.1. Static Functional Connectivity

Once this data is constituted by 76 subjects and in the static functional connectivity is created a correlation matrix by subject, it has for each atlas parcellation and session 76 matrices. But once was used two different approaches in the Nipype pipelines, the data can be created using the global, the first half and the second half of the acquisition.

Relatively to using all the acquisition, the statistics of the values present in the correlation matrices by atlas are shown in the *Table 5.4*. In these values can be seen that is an inverse relation between the mean z-value and the number of nodes of the atlas, and it is the 268 nodes atlas that achieve the lower mean z-value. Besides the values of the session 2 are in lower comparatively to session 1.

Table 5.4. Mean, standard deviation, maximum and minimum z-value of the static FC matrices using all the acquisition for each atlas and session of the In-House data.

	Session 1			*Session 2*		
Atlas applied	**Mean z-value**	**Maximum z-value**	**Minimum z-value**	**Mean z-value**	**Maximum z-value**	**Minimum z-value**
150 nodes	0.2796±0.4637	2.8899	-2.1073	0.2018±0.4632	2.8755	–2.1305
268 nodes	0.2538±0.46129	2.9204	-2.2490	0.1911±0.4651	3.1195	-2.1083
100 nodes	0.3114±0.4699	2.9463	-2.0314	0.2246±0.4703	2.7680	-2.2300
Freesurfer	0.3439±0.4795	3.4451	-1.8653	0.2650±0.4873	2.8192	-2.0470
Aal	0.3280±0.4886	2.8869	-1.8499	0.2650±0.4873	2.8192	-2.0470
50 nodes	0.3670±0.4810	2.830	-1.765	0.2721±0.4849	2.7332	–2.0472

Using the first half of the acquisition the statistical measures of the created data can be summarized as in *Table 5.5*. In the case of the second half the metrics was as in *Table 5.6*. The standard deviation and the maximum values are higher, while the minimum values are lower in the divided parts acquisitions, when compared with the total acquisition. This was already expected because the time of acquisition is to small, which become the signal oscillations a major influencer in final value. In the other hand, when compared the divided parts in each acquisition method, the second half showed higher average z-values.

Table 5.5. Mean, standard deviation, maximum and minimum z-value of the static FC matrices using the first half of the acquisition for each atlas and session of the In-House data.

	Session 1			*Session 2*		
Atlas applied	**Mean z-value**	**Maximum z-value**	**Minimum z-value**	**Mean z-value**	**Maximum z-value**	**Minimum z-value**
150 nodes	0.2764±0.6138	4.2972	-2.7941	0.2140±0.6330	3.3690	-3.5542
268 nodes	0.2555±0.6132	3.4828	-3.0041	0.2009±0.6352	3.6711	-2.9958
100 nodes	0.3087±0.6193	3.2499	-3.0148	0.2391±0.6398	3.3453	-2.9218
Freesurfer	0.3305±0.6354	4.1142	-2.7181	0.2725±0.6445	3.4051	-2.9049
Aal	0.3305±0.6354	4.1142	-2.7181	0.2793±0.6527	3.3157	-3.0219
50 nodes	0.3700±0.6310	3.6500	-2.3620	0.2901±0.6520	3.1633	-2.5348

Table 5.6. Mean, standard deviation, maximum and minimum z-value of the static FC matrices using the second half of the acquisition for each atlas and session of the In-House data.

	Session 1			Session 2		
Atlas applied	**Mean z-value**	**Maximum z-value**	**Minimum z-value**	**Mean z-value**	**Maximum z-value**	**Minimum z-value**
150 nodes	0.2958±0.6218	3.7783	-2.9929	0.2210±0.6325	3.4543	-3.0774
268 nodes	0.2555±0.6132	3.4828	-3.0041	0.3115±0.3549	3.2442	-2.1195
100 nodes	0.3909±0.6283	3.3063	-2.7760	0.2457±0.6381	3.8660	-3.1082
Freesurfer	0.3669±0.6341	4.0272	-2.8313	0.2861±0.6504	3.7428	-2.9178
Aal	0.3509±0.6382	3.6952	-2.7990	0.2861±0.6504	3.7428	-2.9178
50 nodes	0.3909±0.6283	3.3063	-2.7760	0.2955±0.6544	3.8072	-2.6652

5.4.1.2. Dynamic Functional Connectivity

In this case, once the pre-processed acquisitions are made by 355 volumes, the number of dynamic matrices for each subject and parcellation is 355. So, for each parcellation there is a total number 355 x 76, so 26980 matrices. In the *Table 5.7* are shown some stats about the all matrices produced for each parcellation and session using all the acquisition.

Table 5.7. Mean, standard deviation, maximum and minimum z-value of the dynamic FC matrices for each atlas and session of the In-House data using all the acquisition.

	Session 1			*Session 2*		
Atlas applied	**Mean z-value**	**Maximum z-value**	**Minimum z-value**	**Mean z-value**	**Maximum z-value**	**Minimum z-value**
150 nodes	0.1990±0.7073	1	-1	0.1393±0.7169	1	-1
268 nodes	0.1817±0.7101	1	-1	0.1325±0.7177	1	-1
100 nodes	0.2202±0.7041	1	-1	0.1536±0.7158	1	-1
Freesurfer	0.3439±0.4795	1	-1	0.1774±0.7132	1	-1
Aal	0.2407±0.7010	1	-1	0.1814±0.7129	1	-1
50 nodes	0.2571±0.6973	1	-1	0.1841±0.7136	1	-1

5.4.2. HCP Data

5.4.2.1. Static Functional Connectivity

The HCP data was formed by 100 subjects and two sessions, so for each session there are 200 connectivity matrices. In this case was not created the data with the acquisition divided into two parts, since there was no opportunity to create the data, since the datasets associated to HCP are large datasets. The reason is that they have 1200 volumes by acquisition the triple in relation to In-House data.

Table 5.8. Mean, standard deviation, maximum and minimum z-value of the static FC matrices for each atlas and session for HCP data using all acquisition.

	Session 1			*Session 2*		
Atlas applied	**Mean z-value**	**Maximum z-value**	**Minimum z-value**	**Mean z-value**	**Maximum z-value**	**Minimum z-value**
150 nodes	0.3570±0.3570	3.0790	-2.2650	0.3490±0.3430	2.6890	-1.5630
268 nodes	0.3110±0.3550	3.2440	-2.1200	0.3050±0.3430	2.5840	-1.4960
100 nodes	0.3990±0.3640	2.9540	-2.2650	0.3920±0.3500	2.5120	-1.2470
Freesurfer	0.4200±0.3670	3.0540	-1.8740	0.4120±0.3530	2.6410	-1.3120
Aal	0.4000±0.3810	3.1800	-2.0520	0.4010±0.3790	2.7840	-1.2440
50 nodes	0.4820±0.3760	3.0200	-2.0840	0.4740±0.3610	2.5820	-1.1180

Using all the acquisition approach, the stats values about the correlation matrices created in this part are described in the *Table 5.8*. As happened to the In-House data, the mean z-values demonstrated that exist an inverse relationship between the mean z-value and the number of nodes in the atlas, and once again it's the 268 nodes atlas that has the lower mean z-value. Moreover, the mean z-values are higher in the case of the HCP data in relation to In-House data.

5.4.2.2. Static Functional Connectivity

The rs-fMRI in HCP has 1200 volumes, so the total number of matrices are 1200 for subject. But once there is two phases for each session, to each parcellation there is 240000 connectivity matrices with dFC information (1200 volumes x 100 subjects x 2 phases). One more time is showed in the *Table 5.9* some stats about the data present in the matrices. In this case only was created the dynamic functional connectivity for the Aal parcellation, due the same point mentioned previously, the lack of space. The datasets created with this type of information achieved easily the 30 GB.

Table 5.9. Mean, standard deviation, maximum and minimum z-value of the dynamic FC matrices for each atlas and session using HCP data using all acquisition.

	Session 1			*Session 2*		
Atlas applied	**Mean z-value**	**Maximum z-value**	**Minimum z-value**	**Mean z-value**	**Maximum z-value**	**Minimum z-value**
Aal	0.2747±0.6926	1	-1	0.2691±0.6936	1	-1

5.5. Deep Learning Application

The objective in this method was to find optimal architectures and hyperparameters that, using functional connectivity features, could get good performances in subject's classification tasks. Therefore it was tested two different Deep Learning models based in different types of layers the fully connected layers and convolutional layers, and in several architectures with diverse depths. Besides it was always a continuous search for the hyperparameters that fit better to the models created. The fine-tuning of these parameters is fundamental for the learning process to evolve and achieve the main objective, models with good generalization. So it was essential to create a way to study, compare and get a feedback from the developed models in order to make the right decisions towards the continuous improvement of the best performance models. The plan was to design a small framework working alongside of other methods, as part of a bigger framework that deals with functional connectivity extracted from rs-fMRI data with the final objective of creating "good" DL models with the respective validating measures.

The Deep Learning framework was divided in two major parts, the models' fine-tuning part and the final models, as is shown in the Deep Learning application represented in the *Figure 5.1*. As is evidenced by the figure and names, the fine-tuning part is responsible to adjustment of the model continuously until it is approved to pass to the final phase. In the final phase the deeper tests are done, with increasing number of repetitions and saving the model and the performance values for future uses. The validation of each model is made manually, after analysis of the performance measurements obtained in the results.

This part of the development is mostly based in the library package Keras and it was created four modules, the *DL_keras, Keras_utils*, *Data_analysis*, and *Stats_data_module* to generate, manage and analyse Deep Learning models. The first module comprehends the most important methods used in the final application, while the Keras_utils module has more methods to support the other modules. The *Data_analysis* and *Stats_data_module* are the modules that offer the different analysis functions to study the models performance and the classification results.

5.5.1. Keras Package

The Keras library[177], a powerful and easy-to-use Python package, was the main tool used in this method, once it offers a set of methods to develop and evaluate Deep Learning models. It is a library that was designed to be easy implementation, minimal, modular and easy to extend, features that made this library useful to fast developments and experimentations. This package used other Deep Learning designed libraries, such as the Tensorflow and Theano, and they provide to Keras the great capability to work with a diversified group of DL models. Their possibilities of to use the graphics processing unit (GPU), which have a great computational power, that decreases multiples times the time consumption in relation to the normal computation using central processing unit (CPU).

The development of a model with this library normally follow the next steps:

1 **Definition** – In this first step there is the construction of the model. Once the Keras models are defined as a layer sequences, the model has to be created by a set of layers added in an intended order. The layers used are Keras objects with different parameters that can be changed and be used according to the construction model. In the first layer, we must be careful to define the shape of the inputs, as well as, to define the correct nodes in the last layer, since they must be equal to the number of labels used.

2 **Compilation** – In the compilation process, the model uses an intended backend DL library (e.g. Theano or Tensorflow) to create the representation of the define model, to prepare the network for training and to make predictions using a hardware (e.g. CPU or

GPU). In this step is defined the loss function to evaluate the weights and the optimizer algorithm used in the learning process.

3 **Fit/training** – After the definition and compilation of the model, we are ready to begin the training process. So, in this method the training data is placed into input and if the validation is required a proper data has to be given. . In this step the number of training epochs and the batch size also has to be mentioned.

4 **Evaluation** – In the last step of the process, it is used a dataset to predict and check the differences between the prediction and the true labels. It gives the error and the accuracy associated to this prediction.

In the construction and compilation process the Keras gives different objects, such as different types of layers, loss functions, initializers, activation functions, optimizers, regularizers and other. Furthermore, Keras provides several functions to configure the developed models, such as the weights and the architecture of each model.

The backend library used in the compilation can be adapted to the user preferences. Although, there is some alterations in some methods, the change of backend at any time don't brings any problem in the computation of the process. We especially worked with Theano library, developed to optimize and evaluate mathematical expressions involving multi-dimensional arrays in an efficient way, that make this package a good tool for development of models[184]. We choose the Theano library: first of all, because it was used in the first Deep Learning models; the creator of this package published recently a book about Deep Learning[98] and, finally, was not find any significant differences in training times. Curiously, during the last year, this has been changing, since the TensorFlow had a rapidly growth with important improvements in the optimization processes. In other hand the group responsible for Theano package announced that theano development will stop. Now, it is advised to use the Tensorflow library as backend instead of Theano. The TensorFlow is an open-source library as Theano, developed for Machine Learning that can do numerical computation using data flow graphs[185]. To change the backend it is only needed to change the keras configuration file "keras.json" or define it in the python module where is the script.

5.5.2. Models Evaluation

5.5.2.1. Models Validation and Testing

The model performance can not be only tested and demonstrated using the training data hoping that the model will generalize and perform well on real and unseen data. That is why we validated the data to measure the stability of the machine learning, in order to get some guarantee that the model has learned the principal features from the data without extract too much noise, demonstrated by lower bias and variances. The values that demonstrate if occur or not the underfitting or overfitting problem inhibits the generalization process that emphasize the importance of model validation before the use in the final prediction test. The bias and variance are two sources of error that the machine learning application try to minimize, demonstrating if the supervised learning algorithms will generalize well beyond the training data. The bias are related to the erroneous assumptions in the learning data between the prediction and the correct value. In other words, it represents how much relevant features are, its relation in the data and if the model fits well or not. The high value corresponds to worst performances related to underfitting problem. The variance is the error caused by the normal variability of a model prediction, also used to measure the sensibility of the learning process to fluctuations that occur in the training set. In case of the variance being too high, means that the model learned to much noise from the data, so it is in overfitting.

The most rudimental evaluation in the final of the training is the residual values. It estimates the final error for the model, also known as the training error. Despite of give us a first feedback of model performing on the training process, it does no demonstrate whether this model is in an underfitting or overfitting stage. Other methods are needed to validate and test the model performance. New techniques have been rising in answer to this need, as the hold-out and the cross-validation.

The hold-out method was one of the first validation tools to appear and it is the most simple. The method divide randomly the primary dataset used to training into two datasets, the training data and the testing data, e.g. the validation data. The split is non-overlapping and can variate the percentage of the split, but the most important is the size of the training data that should not be too small in order to not compromise or inability the learning process. It is recommended a data split around 70% to 80 %. Therefore, the classification model uses the training data in the learning process and, only after that, the validation data is used to measure the difference between the prediction and the true values or labels. In the final it is possible to get the error associated to this model as well as the accuracy in the unseen data. So, there is a better approach in relation to the residuals analysis and it does not compromise much more the time and resources consumption. However, this approach continues to have some problems, especially because the model evaluation can presents high variance, since it depends of the training and test data features. Therefore, was created a different approaches, normally known as cross validation techniques.

The cross-validation is based in the same process of the hold-out, through the split of the data set into two datasets. However it can be applied to more subsets allowing the use of all the data in different moments to train the model. Therefore there is no risk to loose important features. Further, the several splits done in the cross-validation process do not compromise the training and validation data, since provides enough data for both. Some examples of cross-validation techniques are the K-folds cross validation or the Leave-p-out or leave-one-one cross validation. In this particular work we used the K-folds.

The k-folds cross-validation begins by dividing non-overlapping and randomly data into k subsets. The hold-out method is repeated k times and, in each time, one of the k subsets is used as validation set to testing the model, and the remaining k-1 are used to training the model. Therefore, in order to get the total effectiveness, the final error estimation is the average of all k test performances results. This method eliminates the problem associated with the division of the data, since all examples in the data set are used at least once to test and to train the model as k-1. The consequences are only positive if the bias and the variance are reduced. The majority of the studies preferred the k = 5 or 10, but it can be any value bigger than 1.It is important that each fold represents the whole data, so the amount of data indirectly affect the K choose to apply in the cross-validation. Unfortunately, the cross-validation has a problem: once the training and test are repeated k times, it needs k-1, which means more computation and time consumption.

The different validation methods are important to evaluate the model in the final of the training, so they are used as a tool to select the best model architecture and hyperparameters, what in machine learning it is known by fine-tunning of the model. After the validation process, is always made the final test using the test set using unseen data. This data should be a good representation of the data used in the training and validation process, to get a real performance as well as, should be original in order to not create a model that the learning process is already adjusted.

In this work, the validation and evaluation had some adaptations due some classification specifications and data size. The principal classification task in this work was the fingerprint, in which the model training and testing data had some rules. The data used in each classification process was divided in two datasets corresponding to the different acquisition sessions acquired in different times. One dataset was used as training data and the other used as data test. The cross-validation was selected to validate the model in the fine-tuning, however this cannot be applied directly for both situation, in the static and dynamic functional connectivity for different reasons.

In the static approach the major limitation resides in the small size of the dataset by label (subject), since that was only one example for each class in the case of the In-House data or two in case of the HCP data (two acquisitions phases), which make this method impracticable. Once for any subset division, there are always labels that are only present in the training data and others that are only in the validation data. Consequently, the labels present in the test data never enter in the training process that made the prediction practically impossible in the testing data. We already tested and the results confirmed our hypothesis. In order to solve this problem, we used different approaches. Firstly, a full dataset was used for training and the other was used for validation and testing. In the validation, data was split from the test dataset using a 25 % of the examples and the whole data was used to the data test that contains the validation cases. This limitation of course bring us some problems particularly with the model created, since it was adjusted to some features of the test data, which falsely improves the performance of the model. In order to improve the evaluation of this approach, the method was repeated 5 or 10 times, and the final performance was measured by the average metrics in training, validation and test process. The corresponding standard deviation for each situation also was computed to analyse the variance inside the results. This first approach was called as **validation approach 1** in future uses, to be easier to identify it. Because our approach fails in data validation, it was created new approach. In this new approach, we maintained the same training data, however we used different datasets for the validation. To do that the time-series acquisition with a number of volumes was divided into two equal acquisitions that, after the preprocessing steps and the extraction process of the functional connectivity values, results in two datasets. One was used as validation data and the other as test data. The process of training, validation and testing also was repeated 5 or 10 times, in order to obtain more reliable results. This approach was called as **validation approach 2.** Relatively to the dynamic functional connectivity data, the labels have an ample number of examples to use in the cross-validation. As mentioned above (in the point 6.3.1), the cross validation for each k validation it was predictable. It is explained by the form that is extracted the dynamic FC. If the value extracted for each volume is similar and related to previous and next volumes, using the Deep Learning model, it will be easy to predict the testing data.

To other classification tasks with small labels, such as the gender classification, the cross-validation is a good approach, where all the data from the two session can be combined and used to train the model.

5.5.2.2. Performance Measures

The performance measures are all the metrics used in the evaluation of the model in the learning process, with a training and validation data, and then analysed the generalization process when it is applied the test set. Once the performances measures are intrinsically related to type learning process and since there are two different learning tasks in Machine Learning, the classification and regression, is required the application of different measures tools adapted to each situation. In the classification problems the objective is to try to predict the discrete number of values, so the labels are normally in categorical form and represents a finite number of classes. The task classification can be one of the two types: binary or multi-class classification. The binary classification is when are only two classes two predict, as for example if a person is sick or not, while in the multi-class classification there is more than two class labels to predict. In our case, we used the multi-class classification in the fingerprint application, where the objective is predict the real identity for a subjects group. Relatively to regression problems, the objective is to try to predict a continuous value in the output, for example use age to predict. .Thus, for each situation there is a set of different measures.

The first metrics analysed in the model are related to training. They are capable to show if the model has signal of learning. The classification can be checked by analysing the loss and accuracy during the training. The learning process is characterized by the decreasing of the training loss, that is associated with the simultaneously increase of accuracy. Normally, when the model has the capacity to learn, there is no difficult in create a model that learns all the features of the training data resulting in a very low loss function and 100 % of the accuracy. This great capacity to learn the training data, often becomes the model more prone to the overfitting problem, which is the opposite of our final goal. In order to classify the models training was created a set of metrics.

The model training accuracy was analysed accordingly with the next values:

- **Final accuracy** – The last training accuracy;
- **Accuracy in less 20% of the total epochs** – The training accuracy for the model in the -20% of the total epochs made during the training process;
- **Accuracy difference (final -20 %)** – The difference between the final accuracy and the accuracy with -20% total training epochs made;
- **Max accuracy** – The maximum training accuracy obtained during the training;
- **Epoch first occurrence max training accuracy** – The epoch corresponding to the first occurrence of the maximum training accuracy.
- **Validation accuracy corresponding to last point** – The validation accuracy when happen the maximum training accuracy;
- **Accuracy difference** – The accuracy difference between the maximum training accuracy and the corresponding validation;
- **Cost in the maximum training accuracy moment** – The training accuracy when the first minimum training occurrence cost happened;
- **Validation cost in the maximum training accuracy moment** – The validation cost when the first minimum training cost occurrence happened;

- **Cost difference –** The difference between the training cost and the validation cost for the epoch of the first maximum training accuracy occurrence.

It was extracted the next information from the cost:

- **Final cost** – The last training cost;
- **Cost in -20 % of the total epochs** – The training cost for the model in the -20% of the total epochs used during the training process;
- **Cost difference (final -20 %) –** The difference between the final cost and the cost in -20% total training epochs made;
- **Min cost** – The minimum training cost obtained during the training;
- **Epoch first occurrence min training cost –** The epoch corresponding to the first occurrence of the minimum training cost;
- **Validation cost corresponding to last point –** The validation cost when happen for the first time the minimum training cost;
- **Cost difference** – The cost difference between the minimum training cost and the corresponding validation cost;
- **Accuracy in the minimum training cost moment** – The training accuracy when the first minimum training occurrence cost happened;
- **Validation accuracy in the minimum training cost moment** – The validation cost when the first minimum training cost occurrence happened;
- **Accuracy difference** – The difference between the training accuracy and the validation accuracy for the epoch of the first minimum training cost occurrence.

The validation, is a tool that measure how the model generalize on new data, not used in the training and used in the fine-tuning, in order to find the best architecture and hyperparameters model. It is important to use a validation data before to extract some information about the results obtained, as it was done in this work. . It was created a set of metrics, based in the accuracy and in the cost validation, to classify the tasks created. The information extracted is similar with what was done before, which instead of to use the training data it was used the data validation.

Finally, it is evaluated the real and final model performance in new and unseen data, known as the test data, in order to conclude whether a model predicts well or not, a particular task. In the classifications tasks were used one more time the accuracy and loss tests. The training process is tested only at the end, which results in less information to manage. The information extracted, based in the test accuracy, is described as:

- **Final test accuracy**
- **Accuracy difference between final training and final test**
- **Accuracy difference between the best training and final test**
- **Accuracy difference between the final validation and test.**
- **Accuracy difference between the best validation and test.**

And in the case of the test cost is described as:

- **Final test cost**
- **Cost difference between final training and final test**
- **Cost difference between the best training and final test**
- **Cost difference between the final validation and test.**
- **Cost difference between the best validation and test.**

The final test has the objective to achieve the minimum loss test cost that it is usually associated with the best model accuracy. However, sometimes when is evaluated several models, the values are not completely correlated (e.g., maximum accuracy and minimum cost), which leads to take a decision under the differences and the importance given to each part. The generalization error, characterized by the difference between the training and test cost, should be as small as possible. That is why the calculus of the differences between the training and the test was done. But, importantly, in most of time the cost do not need the use of this difference, since the final training cost is so small (can be rounding to zero). In addition, if the values are similar between validation and test loss, we can conclude that the validation data is representative of the test data.

In addition to all information extracted before, it was used a set of metrics to evaluate the test results and to get other performance measures. The metrics were described as precision, sensitivity, specificity, false positive rate, F1-score and area under the curve (AUC).

The precision, sensitivity, specificity and the false positive rate are related to the different approaches done in the four parts present in a confusion matrix. In the *Table 5.10* it is organized in two rows and two columns that reports the number of true positives, false positives, false negatives and true negatives,. This table allows a deeper and more detailed analysis when compared with a simple measure of the correct classification (e.g., accuracy). Since the accuracy do not take in account the cases where the data is unbalancing, which can lead to false and misleading results. The true positives and the true negatives are correct predictions, while the false positives and false negatives are wrong predictions. However, accordingly with the classification problem the importance given to the wrong predictions, depends on what is more important to reduce (e.g., false positives or false negatives). The medical field is a real case, which it is more important to reduce the false negatives than the false positives. For example, a prediction of a false diagnosis for a disease when a subject is ill it is worse than a false positive, when a true diagnosis is done in a healthy subject. Besides the direct measures, using the different parts of the confusion matrix, it can be constructed several statistical performance measures. Including the accuracy, the precision, the sensibility/recall, the specificity and the false positive rate.

Table 5.10- Correlation matrix in a table layout.

		Prediction	
		Positive	Negative
True condition	Positive	*True positives (TP)*	*False negatives (FN)*
	Negative	*False positives (FP)*	*True negatives (TN)*

The accuracy a simpler measure already mentioned sometimes in this work and can be described as are in the Equation (15). Relatively to the other measures, there is the precision that quantifies how many of the positively predicted are relevant, and can be obtained as in the Equation (16). The sensitivity measures in how good a test is in detecting the true positive classes, and it's calculated based in the Equation (17). The specificity is the contrary of sensitivity, wherein the objective is measure in how good the test is avoiding the false classes, so the true negative classes, as can be seen in the Equation (18). For last, the false positive rate, gives ratio/probability of a model wrongly classify the negative classes in a test, and is described by the Equation (20). The different statistical measures present here has the problem of lead to deceive the real model performance if they are analysed in separated. Some examples, the case of the precision, if the model in test only a few number of positive prediction to the more confident cases, it will improve the precision rate. Other example, if a model returns always true predictions, the sensibility will be maximum, although the model does not need any learning. Thus, during the analysis and evaluation of the final test classification is important take all into account. Besides, there is other measures that use the average of the other statistical measures. The f1-score is an example, and is a good tool to measure the test accuracy, and it's nothing more that the harmonic average of the precision and recall (Equation (20)). The maximum score that could be achieved is 1, and only happen in the case of a perfect precision and recall while in other hand is 0 for the worst scenario.

$$Accuracy = \frac{\sum TP + \sum TN}{\sum Total} \quad (15)$$

$$Precision = \frac{\sum TP}{\sum TP + \sum FP} \quad (16)$$

$$Sensibility/recall = \frac{\sum TP}{\sum TP + \sum FN} \quad (17)$$

$$Specificity = \frac{\sum TN}{\sum TN + \sum FN} \quad (18)$$

$$False\ Positive\ Rate = \frac{\sum FP}{\sum FP + \sum TN} \quad (19)$$

$$F1\ score = 2 \times \frac{1}{\frac{1}{recall} + \frac{1}{precision}} \quad (20)$$

For last it was used the area under an ROC curve (AUC) for measure the performance of the classification models. The ROC curve exhibited the true positive rate (e.g., sensitivity) against the false positive rate (e.g., 1- specificity) for different cut-off points of a parameter. So each point in the ROC characterizes a sensitivity/specificity pair corresponding to a particular decision threshold. The threshold is applied because is not easy to separate the two different populations: the true and negative classes. This creates a distribution of the test results as represented in *Figure 5.12*, where exist the overlap of negative and positives classes. Therefore it is necessary to select a cut-off to discriminate between the two populations, maximizing the examples: classified correctly as positive (e.g., true positives), classified wrongly as positive (e.g., false negative), classified correctly as negative (e.g., true negatives) and classified mistakenly as positive (e.g., false positives).

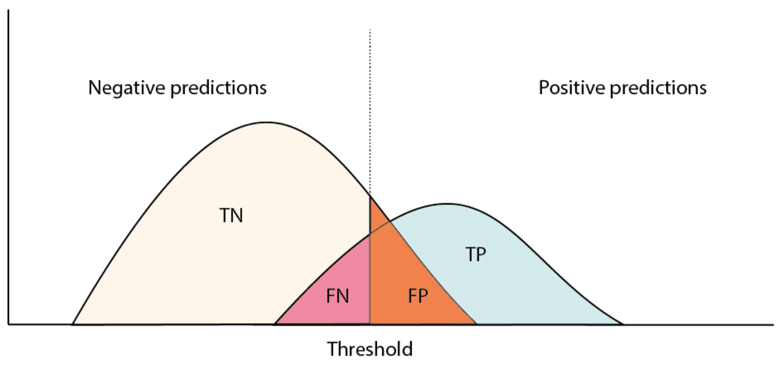


Figure 5.12 –A demonstrative example of use of the threshold to distinguish the distribution of the true positive from the true negative cases.

Besides the ROC curve to be a support tool to select the best cut-off value, also has the property of quantify the performance of the classification, and can also be used to compare two different classification models. The best classification occur when the classification can perfect discriminate the true and negative labels, that corresponds to 100 % sensitivity and 100 % specificity that translates in the plot as a ROC Curve that pass through the upper left corner. In other words, how much closer the ROC curve is to the upper left corner, better it will be the overall model accuracy. The area under the curve (AUC) is a method that measures the area below the ROC curve to get the overall accuracy of the classification model. The AUC value variates between 0 and 1, and the higher values indicate better test performance. The different values obtained demonstrate the performance of the classification model behaviour interpretation of the model as is show in the *Table 5.11*.

Table 5.11- Interpretation of the AUC values relatively to the classification performance[186].

Values	**Model classification performance**
$0,0 \leq AUC < 0,5$	Bad predictor
$AUC = 0,5$	Random predictor
$0,5 < AUC \leq 0,7$	Little predictor
$0,7 < AUC \leq 0.9$	Moderate predictor
$0,9 < AUC < 1,0$	High predictor
$AUC = 1,0$	Perfect predictor

Table 5.12 – An example of the computation of the micro, macro and weighted average precision for a multi-class classification with 2 classes.

Class 1	**Class 2**
TP1	TP2
FP1	FP2
Precision 1 (P1)	Precision 2 (P2)
Number of examples /support (S1)	Number of examples /support (S2)

$$Micro\text{-}average\ precision = \frac{TP1 + TP2}{TP1 + TP2 + FP1 + FP2} \tag{21}$$

$$Macro\text{-}average\ precision = \frac{P1 + P2}{2} \tag{22}$$

$$Weighted\ average\ precision = \frac{S1 * P1 + S2 * P2}{S1 + S2} \tag{23}$$

The different measures mentioned previously are designed and based in the binomial classification wherein the classification task has only two labels to classify. As mentioned before, in this work the principal classification task is multi-class once there is more than one subject to classify. Thus, it was used some approaches to get the global performance metrics, such as the micro, the macro and weighted average. In the micro-average method the values are individually sum up to the true positives, false positives, false negatives and true negative. An example of the micro-average to the precision can be seen in the equation (20) inside of the

Table 5.12. In other hand, the macro-average was only made to the average of the metric in analysis for each class, as can be seen in the example for the precision in the equation (22) in the

Table 5.12. For last, the weighted approach that is an average that take account unbalanced datasets once in this method comparatively to micro and macro average the number of examples for each class is used in the method. As example is the equation (23) in the

Table 5.12.

5.5.3. Models Fine-tuning Part

The models fine-tuning begins by the design of the model architecture and the hyperparameters. Since this part is intended to compare different models topologies, the inputs not only accept singular values but also values in list form. The list inputs will be used to create all the combinations possible, creating multiple models associated a set of train hyperparameters. Therefore, all the inputs inserted under a list form will be, in practical sense, the variable in study in the test.

5.5.3.1. Method Inputs

The method inputs are a set of parameters that can be changed accordingly to the study, which can be adapted to the architecture type of the model or hyperparameters used in the training or even both. In order to better exemplification of which part, they will be explained in three parts, although in the method this division do not happened The first part includes the inputs used to create the convolutional layers that will be add to the model. Once the features used (the correlation matrices) are spatial data, it was implemented the convolutional module with 2D convolutional layers. Thus, it was created a set of inputs to help the development of this type of layers:

- convolutional_list_layers – It is the input that identify the number of layers and the number of features maps applied. The entry format is a list, wherein their elements are the number of features maps used in each layer. Does not have a default value.
- conv_kernel_sizes – It is used to specify the length and the width of the filters used in the construction of the features maps, so the input entry must be tuple shape. Does not have default value.
- conv_activation_function – It is an input that allow to specify the activation functions used in the convolutional layers. An entry value can be a single value wherein all the layers of the model is applied the same activation function, or a list of values where each activation function corresponds to layer of the model. But if the length dimension is smaller than the

number of convolutional layers, the layers where activation function is lacking, it is used the linear activation, the default activation. Has the linear activation as the default value.

- conv_strides – It is an integer or a tuple of two integers used to specify the length and width stride made between two consecutive convolutional processes during the creation of the features maps. If the input is a singular value the stride is made in equal form for the two spatial dimensions. Has the tuple (1,1) as the default value.
- conv_kernel_initializer –Input to specify initializer used to create the kernel weights matrix wherein the entry values are keras initializer objects. Has as default value the glorot uniform initializer.
- conv_bias_initializer –Input to specify the initializer used to create the bias vector wherein the entry values are keras initializer objects. Has as default value the zeros initialization.
- conv_kernel_regularizer – Input to specify the regularizer functions applied to the kernel weights matrix wherein the entry values are keras initializer objects. Has as default value the no utilization of any regularizer.
- conv_bias_regularizer – Input to specify the regularizer functions applied to the bias vector where in the entry values are keras regularizer objects. Has as default value the no utilization of any regularizer.
- conv_activity_regularizer – Input to specify the regularizer functions applied to the convolutional layer output, this is the final activation, wherein the entry values are keras regularizer objects. Has as default value the no utilization of any regularizer.
- conv_dropout_rate – Input that permit to choose the use of dropout layers wherein the value inserted is a rate between 0 and 1 corresponding to rate pretended for the dropout layer. The entry value can be a single value or a list of values, where each element correspond an each layer of the model. But if the length of the list is smaller than the size of the model, the layers where dropout information is missing, it is assumed that will be not used dropout. Has as default value the no application of any dropout.
- conv_batch_normalization – An input that the entry values are booleans values under list form or single values. If they are under list form, each element correspond to a layer in the model beginning by the input layer if not the batch normalization is equal for all layers. The batch normalization is activated of course if the value is true. It does nothave a default value.
- Max_pooling_list - The entry value of this input is a list of boolean values where is possible specify which layers will implement the max pooling process. Once again, each element in

the list correspond to each layer of the model where will be inserted the max pooling layers. And if there is lack of information the default is not use the max pooling. It does not have a default value.

- Pooling_size - Input wherein is entry value is an integer or a tuple of 2 integers representing the factors by which each feature map will be downscale in height and width. If the input is only one integer, the two spatial dimensions are equal downscale by the same factor. Has the tuple (2,2) as default value.
- Pooling_strides - Input to identify which strides values will be used in the max pooling process wherein the entry value is a single integer (the same as tuple of two integers where the value is the same) or a tuple of two integers. Has the stride (1,1) as default value.

The second part are the inputs responsible for the customization of the fully connected layers module. In case of not being used the convolutional layers, the final model created is only constituted by fully connected layers. The layers are implemented using the class Dense from keras module and set of inputs to select the intended parameters for each layer which can be summarized in the following parameters:

- list_layers_nodes – The input that defines the number of layers and the nodes by layer, in each element is a layer with the number of nodes equal to the element. This only select the hidden layers and the input and output layers depend in the input data and labels, respectively. Does not have default value.
- activation_function – The input where is selected the activation function of the layers. The input entry is single or a list of activations functions. In the case of a list, each layer corresponds to an element of the list. The default value is the linear activation function.
- softmax – In this input is possible choose if the output layer is a softmax or not, so the entry value is a Boolean value. The default value is not the use of softmax.
- kernel_initializer – The input where is possible specify the initializer used in the kernel weights matrix and the entry values are keras initializer objects. The default value is the glorot uniform initialization.
- bias_initializer - The input where is possible specify the initializer used in the bias vector and the entry values are keras initializer objects. The default value is the zeros initialization.
- kernel_regularizer – The input that permit apply a regularizer function to kernel weights matrix and the entry values are keras regularizer objects. The default value is the no use of any regularizer.

- activity_regularizer - The input that permit to apply a regularizer function to bias vector and the entry fault values are keras regularizer objects. The default value is the no use of any regularizer.
- batch_normalization – In this input is specify in which layers will add the batch normalization, so the entry format is a list of Boolean values wherein each element correspond to each layer of the model and it is added the normalization case the element is true. In case of the list length to be smaller than the number of layers in the model, the batch normalization will be not applied in the layers without information. It doesn't have default value.
- dropout_rate – This input has an entry a float or a list of floats, values that represent the rate of the dropout to apply, so they can vary between 0 and 1. In the case of the input be a single value, the dropout is made for all layers with the same rate. If it is a list, each element in the list corresponds to each layer model where it is going to be added the dropout layer. If the list and layers numbers does not coincide, the layers which do not have dropout information will not be added the dropout. The default value is no use of dropout in any layer.

Finally there are the inputs related to the different parameters used in model training that are important to model optimization and the learning process, which are the following:

- epochs – Input for the number of epochs used in the training, where the default value is a integer. The default value is equal to 2000 epochs.
- batch_size – Input to specify the batch size used in the training, the default values is also integer. The default value is equal to 1.
- learning_rate – the input to put the pretended learning rate used in the optimization algorithm and the default value is a float. It does not have default value.
- optimizer – the input to specify the algorithm of optimization used in the training process, and the entry values are keras optimizer objects. It has the stochastic gradient descendent (SGD) optimization as the default value.
- loss_function – input to specify the loss function used in the training process and the entry values are loss function keras objects. It doesn't have default value.
- early_stop – This input permit choose the use or not of the early stopping method, so the input values are of boolean type. The default value is no use of early stopping.
- early_patience – In case of the early stopping be activate this input permit choose the number of epochs that the method use to stop if the validation cost not improve in this number of epochs. The default value is 1/5 of the number of epochs.

There is yet three inputs in the method that do not required to be changed, however it is possible if necessary. The three inputs are:

- metrics_list - This variable represents the list of metrics that want to be evaluated by the model during the training and testing. The metric used is the ["accuracy"], the cost is the metric by default, so actually there is always two metrics.
- n_inputs – The number of nodes or features maps in the input layer, this value is obtained directly from the object of the used dataset.
- n_outputs - The number of labels to classify and, the information obtained from the object meta-information of the used dataset.

The default values mentioned all long the inputs description are the values used in the case that the input value is null. The resulting model from their combination is simple model but with some capacity to deal with functional connectivity values under array or matrix for classification tasks. In other hand there is the inputs without default values that have always be mentioned to create a model. The default values can be changed in any time, since the method gives the default values.

5.5.3.2. Models Combinations Test

After the introduction of the different inputs it was used the "*normal_validation_select_hyperparameters_final*" or "*cross_validation_select_hyperparameters_final*" depending in the validation procedure used, traditional validation or cross-validation respectively. All methods use all inputs to create all possible models for them.

The methods begin to check each input that are a singular value or list of values right for the input and transform or make some adjustments if necessary, finishing with each input inside a list. During this process if the input is null and has a default value, it will be replaced. The final result is a dictionary that contains all the inputs, whose will be created all the possible combinations, once the main objective is compare the performance between different models. But this is not mandatory, there is the hypothesis of test only one model each time. The combinatory process is divided into three parts as mentioned in the methods input. This because the combinations of the convolutional layers depends in the number of convolutional layers implemented and the fully connected layers in the number of non-linearity layers used. Each three major inputs parts will firstly combine independently and only after will be combine together. In order to help to inform which combinations will be made, it is presented the information of the combination relative to each part, such as convolutional layers, fully connected layers and train parameters, with the respective total combinations number for each part and total. After this, begins the model creation, training and validation for each combination present in the combinations set. The model constructed is based in the Sequential class from keras, where is added the convolutional layers, fully connected layers and the train parameters according the information provided in each combination. After the model be built and train parameters defined is outputted an information summarized of the model where are described the layers and the parameters by layer, and only then the train begins. During the training wherein there is the optimization of the model in order to verify the evolution of the process are outputted a set of metrics in each training epoch. First, the accuracy and loss for the training data and then to the validation data. As soon as the training is over, consequence of the total number of training epochs be fulfilled or caused by the activation of early stopping process, all the metrics produced are saved in a dictionary for future use. To each model is given a unique id to be easier identify the model created during the analysis of the evaluation metrics and plot results. Moreover, it is a saved an image of the model architecture (e.g., the model layers) with the id of the model in a default temporary folder that in the final of the method will be transferred to the final results folder.

This was the normal process for the traditional validation implementation but there is some modifications relatively to cross-validation. In both traditional and cross validation for each combination create, train and test a model, however the cross-validation depends on the number of k folds used. All the training results are the average of all values obtained in each training. In order to help to find the differences in the results inside the training also is computed the standard deviation. Also, the metrics are used to test the model in the final of each training subset, when the average value of all tests are used as metric to validate the model performance. So this values are computed as well as the respective standard deviation.

After all, the created models are trained and validated, resulting in two lists: one serializable and another no serializable.

The serializable list contains for each combination:

- Id;
- The combination serializable;
- The dictionary with all the metrics results.

The list is in a serializable state to be possible save it, once there is some inputs that are keras objects that cannot be saved in the normal state. So, it was used a method to serialize any input if necessary. To solve the problem with the keras objects, it was was extracted the class and the configurations of the default class of the object and saved in a dictionary.

To the management of this list of results was created the class combinations_results in the Keras_utils module. Therefore, after this method be finished, the first process made is the addition of the new serializable list into a combinations_results object that acts like a repository that will be indispensable for future combinations analysis.

The list not serializable for each combination contains:

- Id;
- The combination not serializable;
- The dictionary with all the metrics results;
- The path for the temp folder with the model architecture plot.

This list of results is created to be used in next analysis methods.

5.5.3.3. Combinations Models' Analysis

This part has as objective to help to evaluate and to analyse which are the best model architectures and hyperparameters, this is names as the fine-tuning process. The method created that implement all the analysis processes was the "final_train_combi" from Keras_utils module, and it was used both lists produce before. The input needs to know which are the base directory where will be saved the different results. Moreover, it is necessary the training, validation and test data in order to proceeds if wanted to the final part where there is the creation, training and test of the final model. All the results produced during this method are placed in a folder named "combinations", since there are results of the combinations part analysis. Then depending of the types of validation used it can be inserted in the "Cross_validation" or "Normal_validation" folder (*Figure 5.15*).

The method use two classes to manage the results from the Keras_utils module, one was already mentioned before the "combinations_results" and the "plots_results". The "combinations_results" class saves the list of results serializable that works like a results repository. Each results list is save individually in a file since can have large dimensions. Also, another file is saved that contains a dictionary with the file names, which contains the results, are the key and the combinations id are the value. This organization becomes easier the management and the retrieving of results. The class has also the method "menu_interactive" that allow to see the existent combinations results in the repository and even analyse any in particular. The data produced in this method is inserted in a folder named "Library_results" which is in the combinations directory (*Figure 5.15*). The other class "plots_results" manages the different results and supports the different analysis methods. This class is important to identify which combinations group is new or has already been tested. If a set of combinations are not saved in this class, it is given to them a new result name that stays associated to the group combinations. Also it is given a name of "test1" to results that comes with combinations, once they are the first results for this combinations set.

In other hand, if the combinations group already exist in the object class, it is verified how many tests were already made and only then created the new name test accordingly with the number of tests that already exist. In order to manage it, the object uses a dictionary as the attribute name of "dic_results" where is saved for each key, the results name and as value a list. Which in the first element contains the description of the combination set and in the second element the list of the names tests that already exist for that combination set.

Moreover, the object uses another dictionary with the name of "dic_id_combi" that has as key the result name and as value a dictionary, which has in the key the tests names, and in the value a list with the ids combinations present in that test.

All object attributes with dictionaries as values have to be updated when is added new results to object, so the method that adds new data is fundamental to the object consistency. Therefore, to prevent this problem is always tested in the object if the combinations group already exist and only then are taken the update actions in the dictionaries. Besides that, the method that adds new data has the function to organize the different results folder and to create a materials set in order to support the analysis process. The method uses a folder in the combinations directory named "Stat_results", where for each results name is associated to a combination set, and it is created a folder with the same name (*Figure 5.15*). Thus, the analysis resulting files and the results information files produced in this method as well as the architectures combinations plots of the test could be put in its own test folder. The method create a file about the combinations in the test:

- **csv_information.csv** – this file has the training and validation results for each combination order by the test accuracy achieved by each one.

In addition, the method also transfer the plot of each combination model architecture saved in a temporary file during the creation and testing of the models to the test directory along with the other analysis files. To finish, the method also provides for each combination two plots saved in the respective belonging test directory. One of the plots has the accuracy and the other the cost training and validation, during the training. The *Figure 5.13* is example for the plot cost and the *Figure 5.14* for the accuracy plot.

After all the different files are saved, the direct purpose of the "plots_results" class is finished. However, the analysis method is not finished yet, there is other method that creates other material to analyse the combinations results set, "create_csv_average_tests".

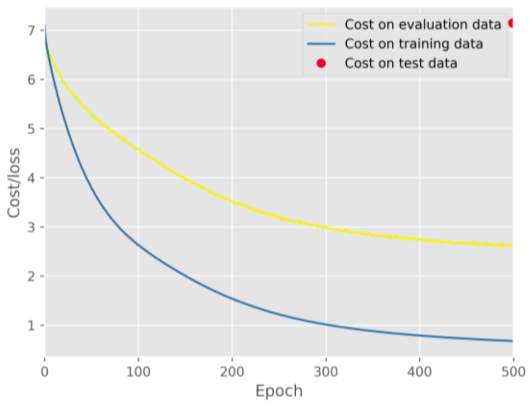


Figure 5.13 – Plot example with the training, validation and test cost during a training process.

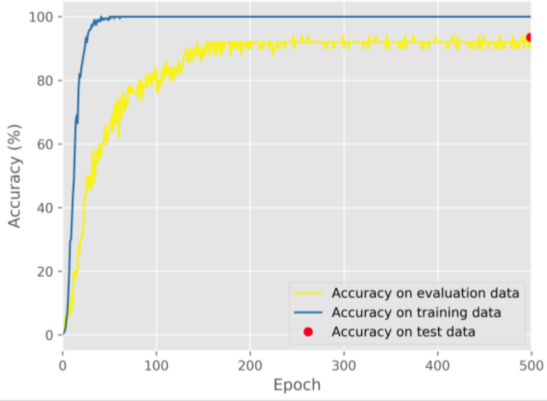


Figure 5.14 - Plot example with the training, validation and test accuracy during a training process.

The "create_csv_average_tests" it is a method that computes the average of the performances measures done over each test in the results set. These are the final performances that will be used to study and to compare the combinations. Besides the average values, also is computed the standard deviation associated to each metric, to more correct analysis and evaluations. Besides the results are demonstrated in descending order by the validation accuracy and there is the description of each combination. Again, it is used the object produced by the "plot_results" class to get the combinations of the tests set that belongs to the result in analysis. And then the training and validation is got for each combination from the object created using the class "combinations_results" that works like a results repository. The file name results of concatenation of "stats_average_" with the result name in analysis and ".csv", in the "Stat_results" directory (*Figure 5.15*).

As an example with two result: results_1 and results_2, where each one has 2 tests, the organization structure is summarised in the *Figure 5.15*.

In the final part there are the option of use an interactive menu that gives some features about the combinations made. The method shows the combinations made and give the option to choose one. After that, there are several options, such as: see the accuracy and cost plots of the training and validation; the model architecture and train parameters; the display of the plots with the accuracy and cost, in training and validation for the combination in analysis and any other combination and pass to the next part in the construction of the final model.

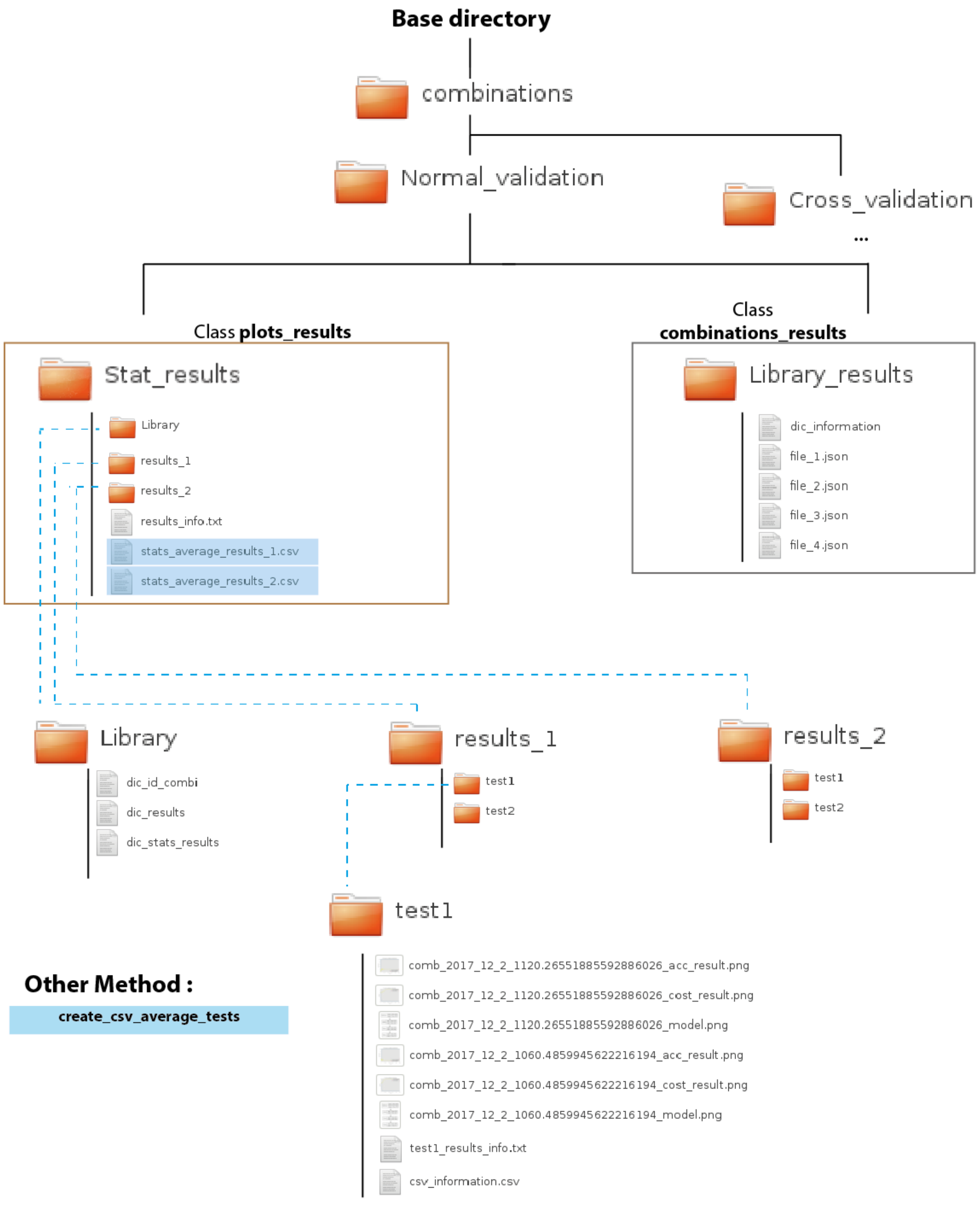

Base directory
combinations
Normal_validation
Cross_validation
...
Class plots_results
Stat_results
Library
results_1
results_2
results_info.txt
stats_average_results_1.csv
stats_average_results_2.csv
Class combinations_results
Library_results
dic_information
file_1.json
file_2.json
file_3.json
file_4.json
Library
dic_id_combi
dic_results
dic_stats_results
results_1
test1
test2
results_2
test1
test2
test1
comb_2017_12_2_1120.26551885592886026_acc_result.png
comb_2017_12_2_1120.26551885592886026_cost_result.png
comb_2017_12_2_1120.26551885592886026_model.png
comb_2017_12_2_1060.4859945622216194_acc_result.png
comb_2017_12_2_1060.4859945622216194_cost_result.png
comb_2017_12_2_1060.4859945622216194_model.png
test1_results_info.txt
csv_information.csv
Other Method :
create_csv_average_tests

Figure 5.15 – A demonstrative example of the data organization, files created and classes as well as other methods involved in the models fine-tuning part.

5.5.4. Final Models Part

This is the final part of global architecture (*Figure 5.1*) designed to create, analyse and save the final models with the best parameters and architectures. The major class in this part is the "sequential_model" present in the DL_keras module. It is responsible for create an object that represents a model. This model has a set of attributes important to the definition of the model. The first is the "id" to be easy to get the model from a repository or identify it. Then has the "version", the version begins in 1 and increases nominally with the number of training processes made over the model. Since a model can be trained more than once, the weights and bias parameters are the resulting values from the last training. Then it has the "model" attribute where is saved the model object from keras that contains all the model: layers, weights and bias parameters. Also has the "best_model_param" attribute that has the best weights parameters for the model, wherein it is saved the best model parameters that occur during the training. The reason why the best model parameters are saved it is because the final model obtained during the training cannot be the best occurred. The initial weights and bias of the model are saved in order to be used in the future. Therefore was used the "initial_weights_bias" attribute to save this parameters. In addition was used the "type" attribute, which describes the type of the model, for example if it has only fully connected layers or if it also has convolutional layers.. The creation process can be made in two different ways: one is adding a Keras object layer in each time, in which is possible to identify the layer index in the model, and method that uses the same inputs combination structure to define the architecture and hyperaparameters.

After the model be defined could proceeds to the model train, validation and test. Processes that can be made in two different ways and so they have different methods to do it. One was already mention in the last paragraph of the combinations section. In final process when is showed an interactive menu can be chosen the option to proceeds to the final test for a specific combination of the combinations set. In this process the last model use the same architecture and all parameters of the combination in analysis. But there is other values that can be added such as the learning rate update factor. Besides it is also possible to select the number of constructed and tested models, important to get the average performance metrics, and a threshold value between 0 and 1, that will be used further in the final analysis. Using this way is not need to mention the type of validation used, normal or cross-validation, since it will be used the same validation approach applied to the combinations part. The other way is little different and it is implemented by the "final_train" method. First of all, it can be done when wanted, not being obligated to follow the process. Then the model architecture, model and training hyperparameters could be selected according to intended. The inputs are similar with the ones described in the combinations inputs section, with exception of the learning rate updating, the analysis threshold (between 0 and 1) used in the final analysis and the number of models created using the defined inputs (repetitions of the tests). Once the method is not capable to know if it is to use the normal or the cross validation, this should be manually selected.

It is important to refer that in the both parts of Deep Learning application, the models are not created with a seed associated to the initializers. In this way it is possible to obtain a large set of results performances, with the variance of the different models with the same architecture and parameters. In other words, this approach get a richer miscellaneous performances about the hyperparameters chosen.

5.5.4.1. Final Results And Analysis

After ending the training, validation and test it is used a set of methods to do three major processes. The first one save the models and its results, the second compute the metrics for the new results and the third make the final analysis to the new classifications. All the results are saved in a folder named “final” that is in the base directory. Then, depending of the validation approach, they can be inserted in the folder “Cross_validation” or “Normal_validation”, which will be mentioned during the manuscript as validation folder.

To save the “sequential_model” object was created other class named “LibraryNN_Keras”. These class is used to manage the objects, including to do the correct storage, the retrieving and provide a different features to select models. The data of creation also is added to the model, in order to be easier to identify the interested model during the selection process. This repository object, creates a folder “NNS” that contains a file for each object model saved as well as a file that ensures the management of the different files.

Relatively to the results they are saved in different ways and classes. A dictionary with all the metrics obtained in the training, validation and test, and meta-information about the “sequential_model” and the data used, such as id, version and size of the training, validation and the test data, is saved in a “results_repositorium” object. Object that is created in the validation folder with the name of “Results”. Using this object a user can rapidly get a result of an intended model by the “sequential_model” id. Besides, the class provide other methods to select a model, where are showed the models order by the test accuracy and data creation. Then, also was used two results lists as was described in the Combinations Models’ Analysis, however the not serializable list adds the output result to test data. One of the list has the results serializable to save, and other no because is used especially to results analysis. In this part, despite of the all analysis process made in combinations part with the combinations_results” and the “plots_results” class as well as the method “create_csv_average_tests”, it was done other analysis processes. .

First for each test made was created the next files, placed in the folder “Stat_results”, using the name “test1” as example:

- **test1_results_info.txt** – it is other file that also has the training, validation and test results, but in this case it was only presented some of the most important metrics in an informative way for a faster reading and analysis.
- **test1_classification_stats_info.txt** – this file is not to help in the analysis but to support other methods, which contains two dictionaries. One contains the value of correct classifications achieved for each label, and the second has all number of possible correct classifications. This information will be essential for future methods analysis in the construction of the classification report.

Secondly other materials are created by the “create_classification_csv_stats_info”, and “create_csv_classification_report_result” methods.

The “create_classification_csv_stats_info” method produces two “.csv” files to help the analysis in how the correct classifications are distributed over the labels. The difference between these two files resides in the analysis approach. One analyses the global values for a result number and each test included in it, named as the concatenation of the result name in analysis more “-classification_stats_info.csv”. The other considers the values obtained for all results, showing also the values for each result, and it’s saved in the “overall_results_classification_stats_info.csv” file. In each file with the approach variants has the absolute frequency, the maximum absolute frequency and the relative frequency for each label. In order to help in the analysis process the results are displayed in a descending order by absolute frequency. The method use the class plot_results to know the tests existent in the results in analysis and the information from the file mentioned previously “_classification_stats_info.txt”.

Finally the method “create_csv_classification_report_result” that creates the classification report for all tests set that share the same combinations set, this is a result that contains a series of tests. What is done is the computation of the average values of precision, sensitivity, specificity, false positive rate, f1-score and AUC for each label and the micro, macro and weighted average of each average metric for all the labels, using the metrics of each test in result in analysis. Also standard deviation was computed for each metric. To support this method is used the class plots_results to get the combinations id of the combinations in the test analysed. Moreover, it is used the dictionary “dic_stats_results” to get the performance metrics of each test in the result folder. The classification report method itself also was provided by module constructed for the purpose of support analysis, called Stats_data_module. The file is created in the “Stats_results” and the name is the concatenation of the results plus the “_classification report.csv”.

Also the meaning of each result as test is different, since in this case each result is associated with a specific model architecture and a set of hyperparameters, and not a combinations set. Per test is created a new result named with a numeric value that corresponds to the number of times of repeated training, validation and tests procedure done.

After the threshold be stablished by the user, the analysis are done in three different levels. The basic procedure begins with getting the correct predictions and the maximum number possible of correct classifications for each label, made by a model. After, the relative frequency is calculated and two sets are done, accordingly with the value: if it is above or below to the threshold. . Then for each group are calculated a metrics set, such as minimum, maximum, the average and standard deviation over all the functional connectivity values of the labels. Besides is saved the plot with both values group in a histogram. To get the necessary information is used a file (test + “_classification_stats_info.txt") created for each test that contains a two dictionaries, one with the correct predictions for each label and the other with the possible maximum number of correct predictions for label. Using this method, the approach can be done in different levels. This method is initially applied to each test and afterwards used to compute each result as well as all the results of the different tests. The different results are placed inside of a “Final” folder that is inserted in the validation folder. All different analysis of tests and results are located in folders accordingly with the variable in analysis, the test and the result. These explanation is described in the *Figure 5.16*.

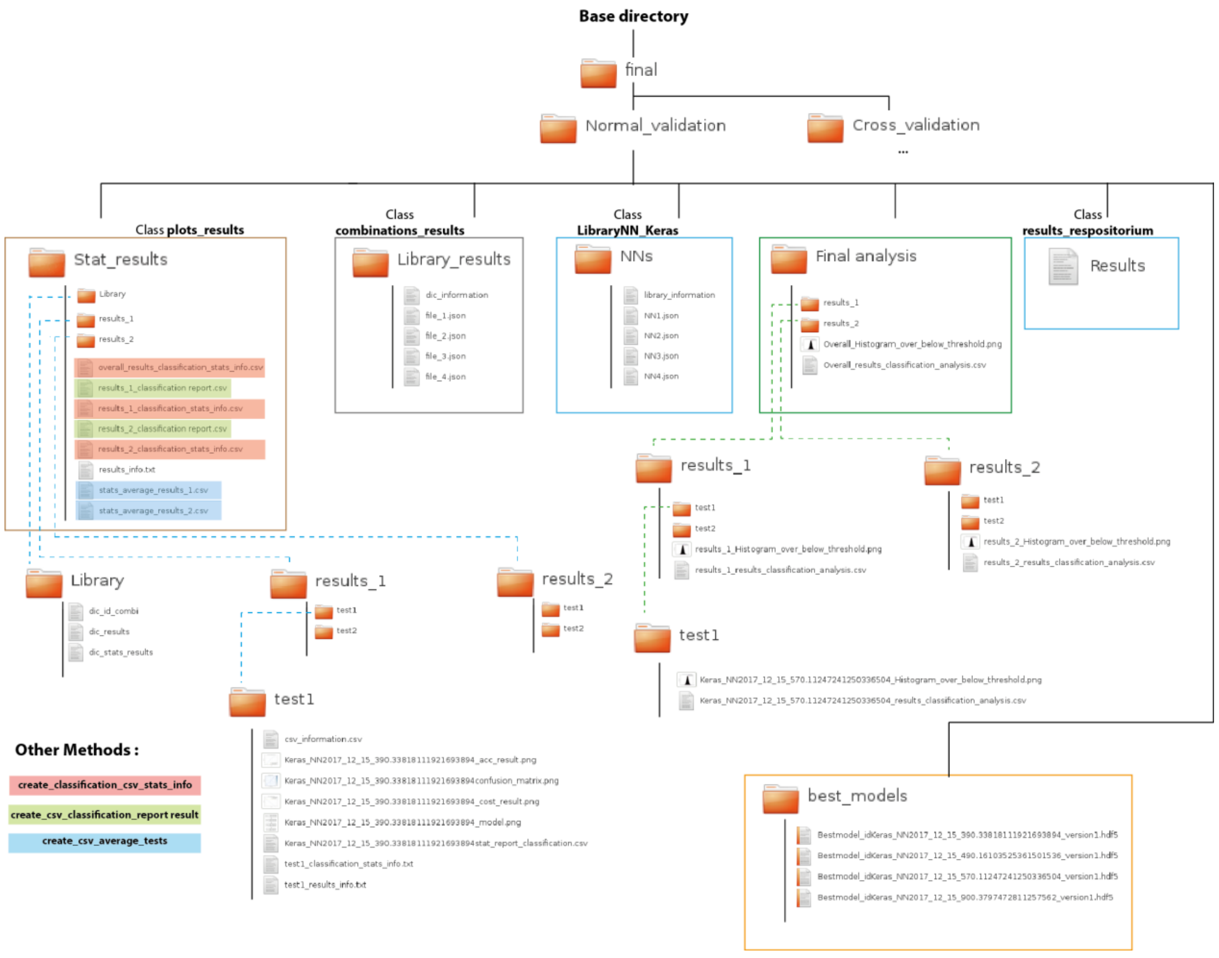

Base directory
final
Normal_validation
Cross_validation
...
Class plots_results
Stat_results
Library
results_1
results_2
overall_results_classification_stats_info.csv
results_1_classification report.csv
results_1_classification_stats_info.csv
results_2_classification report.csv
results_2_classification_stats_info.csv
results_info.txt
stats_average_results_1.csv
stats_average_results_2.csv
Class combinations_results
Library_results
dic_information
file_1.json
file_2.json
file_3.json
file_4.json
Class LibraryNN_Keras
NNs
library_information
NN1.json
NN2.json
NN3.json
NN4.json
Final analysis
results_1
results_2
Overall_Histogram_over_below_threshold.png
Overall_results_classification_analysis.csv
Class results_respositorium
Results
results_1
test1
test2
results_1_Histogram_over_below_threshold.png
results_1_results_classification_analysis.csv
results_2
test1
test2
results_2_Histogram_over_below_threshold.png
results_2_results_classification_analysis.csv
test1
Keras_NN2017_12_15_570.11247241250336504_Histogram_over_below_threshold.png
Keras_NN2017_12_15_570.11247241250336504_results_classification_analysis.csv
Library
dic_id_combi
dic_results
dic_stats_results
results_1
test1
test2
results_2
test1
test2
test1
csv_information.csv
Keras_NN2017_12_15_390.33818111921693894_acc_result.png
Keras_NN2017_12_15_390.33818111921693894confusion_matrix.png
Keras_NN2017_12_15_390.33818111921693894_cost_result.png
Keras_NN2017_12_15_390.33818111921693894_model.png
Keras_NN2017_12_15_390.33818111921693894stat_report_classification.csv
test1_classification_stats_info.txt
test1_results_info.txt
Other Methods :
create_classification_csv_stats_info
create_csv_classification_report result
create_csv_average_tests
best_models
Bestmodel_idKeras_NN2017_12_15_390.33818111921693894_version1.hdf5
Bestmodel_idKeras_NN2017_12_15_490.16103525361501536_version1.hdf5
Bestmodel_idKeras_NN2017_12_15_570.11247241250336504_version1.hdf5
Bestmodel_idKeras_NN2017_12_15_900.3797472811257562_version1.hdf5

Figure 5.16 - A demonstrative example of the data organization, files created and classes as well as other methods involved in the models fine-tuning part.

5.6. Fine-Tuning The DL Models' Parameters

The Deep Learning application begins with the creation of Deep Learning models and with the fingerprint classification task. It was divided in two parts accordingly to the source data: In-House data and Human Connectome Project (HCP).

For both, the first approach was to find the simplest models and verify which hyperparameters are the best to work with. After this point, we goes through more complex models , by increasing the depth model or adding convolutional layers, due the bigger difficulty in the fine-tuning the right parameters. Our first research approach was to create a fully connected layers to add convolutional networks later. The parcellation used to study the different parameters was the 50 nodes (93 nodes), since it achieved the worst performance value in re-test of the fingerprint article (in [19]). In this way there are more space to improve. The parcellation creates smaller correlation matrices with less features to feed the models which make the models lighter in relation to others created with parcellations with more nodes.

The tests were divided in several parts. The first one, and maybe the principal, was the search for the best model resulted from the combination of best activation function, loss function and learning rate. This because there are an important relation between the activation and the loss function in the learning process. Besides that with the addition the learning rate (lr) there is other variable that act upon the process learning in the way that happens. If the lr is too high or too low, the learning process will not occur and it can wrongly suggest that the problem is on other parameters when it is false, leading to the elimination of that possibility. So, it is fundamental to see how the results change with the learning rate. Then it was study the optimizer (learning algorithm), batch size, the initializers of weights and bias. When the process ended, were made new tests in order to increase the depth of the model. In the last step, we used t methods to improve the generalization and reduce the overfitting, where are included the regularization, dropout and batch normalization if is used a batch size greater than 1. It was not possible to study all the hyperparameters, since it is timing immeasurable.

Briefly, the steps followed are described:

A. Joint tests with models' parameters – activation and loss functions, and learning rate.

B. Separated tests for models' parameters – optimizers, batch size and initializers.

C. Change/Increase the models' depth.

D. Use techniques/ methods to reduce overfitting – Regularization, dropout and batch normalization.

6. RESULTS AND DISCUSSION

6.1. Classification Based in Pearson's Correlation Similarity as In the Article "Functional Connectome Fingerprint: Identifying Individuals Using Patterns of Brain Connectivity".

In the article the process applied to compare matrices was the similarity based on Pearson's correlation. During the process is made the Pearson's correlation for each subject using the subject's connectivity matrix of a session - the target matrix, with each existent connectivity matrix of other session - matrices as database. Then it is only chosen the matrix with maximum correlation value. If the subjects of this matrices of this maximum value are the same, the classification is correct, so the classification score obtained is 1. If not, it is 0. In test was used two different datasets mentioned in this work, the HCP and In-House data. The type of FC tested was the static.

6.1.1. HCP Data

The results of accuracy by parcellations are presented in Table 6.1.

Table 6.1. Table of accuracies in descending order for each parcellation and session prediction using the HCP data.

Predict session 1			*Predict session 2*		
Parcellation	**Value**	**Difference**	**Parcellation**	**Value**	**Difference**
150 nodes	0.840	0.000	*150 nodes*	0.835	0.000
100 nodes	0.800	0.040	*268 nodes*	0.770	0.065
268 nodes	0.765	0.075	*Freesurfer*	0.760	0.075
Freesurfer	0.745	0.095	*100 nodes*	0.755	0.080
50 nodes	0.655	0.185	*50 nodes*	0.700	0.135
AAL	0.490	0.350	*AAL*	0.590	0.245

The maximum accuracies obtained were using the parcellation of 150 nodes for both sessions prediction, but was predicting the session one that was obtained the greater value about 0.84 although the accuracy in session two was almost the same once the difference was only one real case (see Table 6.1).

Analysing the results by the number of regions present in the atlas, it can be easy verified that the accuracy order is pretty much related to the number of regions of atlas. Even more when the analysis is one by the atlas type (e.g., structural or functional), wherein the order is completely maintained by descending order of the nodes number. In general, the AAL atlas has the worst performance, even in relation to the 50 nodes atlas (93 total nodes) with fewer nodes. This could lead to the conclusion that, for over the same number of regions, the functional atlas could have better performances. But the Freesurfer, the other anatomical atlas, achieved better results than 100 nodes (184 total nodes) atlas with more number of brain regions (24 more exactly) (in Table 6.1).

For a more accurate analysis, it was also analysed the values of correlation obtained during the prediction processes. And for all the predictions, it was verified that exists a significant difference between the correlations' average values for the correct and incorrect classifications (see Table 6.2).

Table 6.2. Mean, Standard Deviation, Max and Min of the correlations values in the correct and incorrect classifications and respective difference for HCP data.

	Correct classification	**Incorrect classification**	**Difference**
Average value	0.631	0.583	0.048
Standard Deviation	0.095	0.09	0.005
Maximum	0.868	0.765	0.103
Minimum	0.331	0.297	0.034

So, a correct classification is intrinsically dependent of the correlation value obtained during the process, where greater values of correlation have higher probably of be classified correctly.

Now analysing the difference mean correlation values for each atlas, it is possible to conclude that the parcellation has either a relation with the difference correlation value amount. Where atlas with more nodes are associated in general to larger values, which means that exist a linear positive behaviour. So, once the accuracy has the same behaviour, the amount of difference intrinsically correlated with the accuracy (see *Table 6.2*).

Table 6.3. Differences table of correlations mean values between correct and incorrect classification for predict session 1 and 2, in descending order.

Predict session 1			*Predict session 2*		
Parcellation	**Value**	**Difference**	**Parcellation**	**Value**	**Difference**
150 nodes	0.073	0.000	*150 nodes*	0.089	0.000
100 nodes	0.070	0.003	*100 nodes*	0.074	0.015
Aal	0.070	0.003	*268 nodes*	0.059	0.030
268 nodes	0.066	0.007	*Freesurfer*	0.047	0.042
Freesurfer	0.062	0.011	*50 nodes*	0.038	0.051
50 nodes	0.044	0.029	*Aal*	0.033	0.056

6.1.2. In-House Data

Table 6.4. Table of accuracies in descending order for each parcellation for each session prediction using the other session using the In-House data.

Predict session 1			*Predict session 2*		
Parcellation	**Value**	**Difference**	**Parcellation**	**Value**	**Difference**
150 nodes	0.342	0.000	*150 nodes*	0.342	0.0000
100 nodes	0.316	0.026	*268 nodes*	0.303	0.0395
268 nodes	0.316	0.026	*100 nodes*	0.289	0.0526
Freesurfer	0.263	0.079	*Aal*	0.276	0.0658
Aal	0.250	0.092	*Freesurfer*	0.250	0.0921
50 nodes	0.237	0.105	*50 nodes*	0.237	0.1053

Once again, the best results occurred for the 150 nodes atlas, 0.342 for both predictions, and still reasonable to note that the number of nodes continues to have a positive linear relation with accuracy values. Although, the AAL atlas stands out in both cases. In predict subjects from session one it had the same accuracy as the 268 nodes atlas and in the session two it got better more result than the 100 nodes atlas, atlas with much more nodes. This can show for now that the rs-FMRI from In-House data has better results with anatomical atlas than other functional and more specific atlas. Although the results of the Freesurfer, the other anatomical atlas with more number of nodes, didn't have better results than other functional based atlas with a similar number of nodes, and even stayed below AAL performance (see Table 6.4).

Once more, analysing the average correlations values for correct and incorrect classifications, it was verified that the correlation value affects positively the classification result. And again, the difference between the two situations was significant, even more than for HCP data. Moreover, comparing the values obtained by the two data, HCP and In-House data, it's easily discernible that the values in In-House data was very lower comparatively to HCP data. Therefore, the amount of correlation value also is correlated with the average accuracy over the dataset (see Table 6.5).

Table 6.5. Mean, Standard Deviation, Max and Min of the correlations values in the correct and incorrect classifications and respective difference for In-House data.

	Correct classification	**Incorrect classification**	**Difference**
Average value	0.464	0.394	0.070
Standard Deviation	0.082	0.089	-0.006
Maximum	0.650	0.625	0.025
Minimum	0.271	0.190	0.081

An analysis of the correlations values for each atlas separately, it demonstrates not very noticeable that the number of nodes can induce the difference between the correlation values for the two situations. And in turn, it also has relation with the finals accuracy results, wherein the Aal Atlas stands out once again, showing that is a relation between the difference and the correct classification (see Table 6.6).

Table 6.6. Table of differences correlations mean values between correct and incorrect classification for predict session 1 and 2, in descending order, using the In-House data.

Predict session 1			*Predict session 2*		
Parcellation	***Value***	***Difference***	***Parcellation***	***Value***	***Difference***
268 nodes	0.094	0.0000	*268 nodes*	0.092	0.0000
150 nodes	0.089	0.0050	*150 nodes*	0.075	0.0169
Aal	0.088	0.0057	*Aal*	0.073	0.0187
100 nodes	0.086	0.0080	*Freesufer*	0.071	0.0208
50 nodes	0.071	0.0234	*100 nodes*	0.064	0.0272
Freesurfer	0.064	0.0299	*50 nodes*	0.051	0.0407

6.1.3. Compare the Results HCP Data and In-House Data

In overall, the performances results in In-House data were much lower comparatively to the results of HCP data. It is only needed a brief analysis to the interval's values, which to In-House data is [0.237; 0.342] and to HCP data is [0.630; 0.840]. This gap between the dataset can be explained simple by the different quality of the data caused by the different magnetic field of the acquisition machine, as proved in [187]. But, in both, the parcellation exhibits a strong relation with the accuracy results, where the increase of the nodes number the classification performance also increases. Other key point is related to AAL atlas, with the HCP data it had the worst results, but with the In-House data it was the opposite. Their results were above others with parcellations with many more nodes. As referred, this can be explained by taking into account the type of atlas: anatomical or functional. Furthermore, the in In-House data, which had the worst acquisition parameters and machines, such as the magnetic field, 1.5T instead of 3T, and when the brain regions are divided anatomically maybe it could extract better features for subjects' classification. Besides that, the functional atlas was created based in fMRI volumes acquired in 3T, being one more reason for this gap between the two datasets. Although, this can be a solid conclusion, since Freesurfer's results didn't have the same behaviour. But again, the brain regions division could not the more convenient for in In-House data (see Table 6.1 and Table 6.4). The accuracy results for each dataset were a little better for predict subjects of session 2. Moreover, there was no found significant differences between the predict session one or session two for each data case although the interval that divides the sessions' acquisition was different: one year and half instead of one day (Table 6.7).

Table 6.7. Average of accuracy values by session prediction and data used.

	Predict session 1	**Predict session 2**
HCP data	0.740	0.742
In-House data	0.287	0.283

Then, analysing the correlations values, it is possible to verify that the values of HCP data are higher than In-House data's values, with similar standard deviation. The values are even disjointed; the interval of values doesn't superpose (see Table 6.2 and Table 6.5). Besides that, it was possible to observe that smaller difference correlations values between correct and incorrect predictions are associated to best accuracy results when it is only compared between the two datasets. However, in spite of the average of correlation average difference has a negative relation with the accuracy, analysing the value associated to each parcellation and the accuracy results is quickly verified that happens the contrary. So, on the overall it is not possible to take a solid conclusion about how the amount of correlation average difference has influence in the final accuracy (see *Table 6.3*, Table 6.6 and Table 6.8).

Table 6.8. Average of Pearson's correlation average differences between correct and incorrect classifications and the correspondent standard deviation for HCP and In-House data, prediction the session 1 and session 2.

	HCP data	
	Predict session 1	*Predict session 2*
Mean of correlation mean difference	0.064	0.057
Standard deviation	0.0097	0.0199
	In-House data	
	Predict session 1	*Predict session 2*
Mean of correlation mean difference	0.082	0.071
Standard deviation	0.0108	0.0126

Lastly, it was verified how the subject's classification occurs inside the correct situations. For that was studied for each subject how it is the average of correct classifications taking into analysis all predictions or both predictions in each parcellation (predict session 1 and predict session 2). These results show interesting fact, that there is group in both situations that are easily to classify. The results demonstrated that inside the correctly classified group a subject is more 50% of the times correctly classified for both datasets. In the case of HCP, the percentage is 72.71% in all predictions and 69.5% in the full parcellation prediction, larger comparatively to In-House data, 52.8% and 65.3% (see Table 6.9).

Table 6.9. Average values, standard deviation, max and min, of correct predictions for each subject.

	HCP data	
	All predictions	*Predictions by parcellation*
Average value correct classifications	17.45 /24 (72.71%)	8.34 /12 (69.5%)
standard deviation	5.41	2.95
MAX	24 (max)	12 (max)
MIN	2	1 (min)
	In-House data	
	All predictions	*Predictions by parcellation*
Average value correct classifications	6. 34 /12(52.8%)	3. 86 /6(65.33%)
standard deviation	4.00	1.87
MAX	12 (max)	6 (max)
MIN	1 (min)	1 (min)

6.1.4. Comparing with the Results of the Article “Functional Connectome Fingerprinting: Identifying Individuals Using Patterns of Brain Connectivity”

Appling the 268 nodes atlas[50] and using all region timeseries to get the functional connectivity, the accuracy rate was 117/126 (92,9%) and 119/126 (94,4%) for respectively pair target-database session 1-session2 and session2-session1[19]. Our test got worst results: for HCP data was 76,5% and 77% and for in In-House data was 31.6% and 30.3%, corresponding to pairs target-database session1-session2 and session2-session1. Just a brief review to conclude that the values are very distant from the article results. Although a recent article [187] also tried to replicate the method of classification in a own dataset, and the accuracy values range was 42% - 55%. But in other hand they achieved better results when using a larger HCP data with 900 subjects.

In the case of HCP, despite the data belongs to the same online project and to be acquired with the same image and RMI machine parameters, the processing done by the researchers in the article could not the same as ours. Besides that, the set of subjects is not the same, since the release from the article no longer exists. Reasons that are supported with the article [187]. But the more likely reason it’s the pre-processing steps and the parameters used, especially the temporal filtering since some tests demonstrated that it eliminates some important signals used in the classification. However, these reasons cannot be enough to explain the great difference so in the results.

The performance results of In-House data was very low but there was expected, since the data was acquired with RMI machines with smaller magnetic field value, more exactly 1.5T instead of 3T. Thereby the fMRI has no so much quality and there is more noise that can make more difficult identify the real subject signal. Besides the interval of time between sessions is also very different in each case, as before mentioned for the In-House data is one year and half instead of only one day in HCP data. Furthermore, for both cases the rs-fMRI processing was done with an own designed processing pipeline that could be different of the realized in the article.

6.1.5. Use Only the Medial Frontal and Frontoparietal Functional Networks Using the Atlas 268 Nodes

These two networks have effect in the both data classifications, but opposites. For the HCP using only the nodes presented in the network medial frontal and frontoparietal increased the accuracy results. Otherwise, the classification performance decreased for In-House data. This result can prove one of the ideas discussed previously as for the effect that a functional atlas has created for volumes acquired in 3T. Because the acquisition of In-House data was made with 1.5T, so the division cannot be so precise and happened loss of essential information in the classification during the correlation process. These results also support the results presented in the article due the increase of accuracy for only this two networks, but continuing far from the final results: 0.99 and 0.98 in the research work.

Table 6.10. Accuracy and difference between correlations mean values for correct and incorrect classifications, to HCP and In-House data test for the 268 nodes parcellation. Using all nodes or only the nodes from network medial frontal (network 1) and frontoparietal (network 2).

In-House data				
	Nodes in network 1 and 2		All nodes	
	Predict session 1	*Predict session 2*	*Predict session 1*	*Predict session 2*
Accuracy	0.211	0.237	0.316	0.303
Difference between correlation mean values	0.061	0.057	0.094	0.092
HCP data				
	Nodes in network 1 and 2		All nodes	
	Predict session 1	*Predict session 2*	*Predict session 1*	*Predict session 2*
Accuracy	0.845	0.895	0.765	0.770
Difference between correlation average values	0.084	0.088	0.066	0.059

6.2. Fingerprint Subjects' Classification - Deep Learning Approach

6.2.1. Fully Connected Models – In-House Data

In the Table 6.11 are presented the final results obtained with the article approach and they are the objective to be outperformed in this DL approach.

Table 6.11. Resume of the values obtained by the article approach (from [19]) in In-House dataset to predict session 1 and session 2, for the different brain parcellations.

Predict session 1			*Predict session 2*		
Parcellation	**Value**	**Difference**	**Parcellation**	**Value**	**Difference**
150 nodes	0.342	0.000	*150 nodes*	0.342	0.0000
100 nodes	0.316	0.026	*268 nodes*	0.303	0.0395
268 nodes	0.316	0.026	*100 nodes*	0.289	0.0526
Freesurfer	0.263	0.079	*Aal*	0.276	0.0658
Aal	0.250	0.092	*Freesurfer*	0.250	0.0921
50 nodes	0.237	0.105	*50 nodes*	0.237	0.1053

The first approach was study how are the best set of hyperparameters that together with the functional connectivity values, amplify the learning process. Thus, the first objective was to create a simple model and fully connected layer, choose the best hyperparameters and only then increases its complexity. The principal additions towards to more complex systems were the increasing the depth model and adding convolutional networks to the models. With the best set of hyperparameters the models can be extended for other parcellations, where with are made some adjustments to the architecture to adapt to the new number of features and its values. Although the best approach was made a fine-tuning process for each parcellation and session prediction, but the time needed will be tremendous.

Also, it was done some normalizations to data in order to test if it was possible get more interesting features to improve the learning process and so the final performance, especially due to the different behaviours of the non-linearity functions. A vital information before beginning the process is the calculation of a random correct classification in the dataset. As the In-House data has 76 subjects, the probability of correct classify one is 1/76, this is 0.01316.

The parcellation decided to create the best set of hyperparameters was using the 50 nodes atlas, once it have the fewer number of features and it has the worst results, so it have a great interval to improve. Besides was used the session 1 to predict the session 2. The fine-tuning of the hyperparameters follow the procedure defined in 5.6 and was used the Python modules set created to support the creation, testing and validation of the model as well as the analysis of the results. The validation approach use in this part was the 1 where 25% of the test dataset was used as validation, and the other parts was always the second defined in 5.5.2.1.After the final set of hyperparameters be formed, it is passed to prediction of the session 1 using the session 2.

The next parcellations used was the 268 nodes and the Aal atlas, once 268 nodes atlas is the most important atlas in the study of the functional connectivity in the article and the Aal one of the most used in Neuroimaging research as it's the case of Neuroimaging lab in ICVS. In this case was already used the validation approach 2, wherein was the division of the acquisition in two parts, and one was used as test and the other as validation.

6.2.1.1. 50 Nodes Parcellation – Part A

<u>Part A - Fine-tuning of the best combination between the loss function, activation function and learning rate.</u>

Tested model parameters:

i. Learning rate: {0.5; 0.3; 0.1; 0.05; 0.01; 0.005; 0.001; 0.0005; 0.0001}
ii. Loss function set: {Categorical cross-entropy, hinge and squared hinge}
iii. Activation functions set: {Relu, Leaky Relu, Sigmoid, Prelu and tanh function}

Other model parameters:

i. Epochs: 2000
ii. Optimizer: SGD
iii. Bias initializer: Truncated Normal class with standard deviation equal to 1.
iv. Weights initializer: Variance Scalling class, with the average of the inputs and outputs.

Total number of combinations: 9 * 3 * 5 = 135.

Number of tests: 10

The first parameters used had as base the first models developed with Theano module, wherein were realized some tests.

Initiating the hyperparameters fine-tunning process, firstly it was tested a set of alphas for the activation function Leaky relu to discover which has better performance.

The test contained 10 hypotheses for each alpha in the alpha rates set:

{0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9}.

Table 6.12. Average validation accuracy and cost values of 3 tests with the alpha parameter equal to 0,2, 0,3, 0,4, 0,5 and 0,6 in the Leaky Relu activation function for 50 nodes parcellation in the test of the part A.

Alpha	Validation accuracy average	Validation cost mean
0,1	0,08553 ± 0,02648	0,99918 ± 0,00015
0,2	0,08421 ± 0,01785	0,99926 ± 0,00014
0,3	0,08289 ± 0,02568	0,99919 ± 0,00015
0,4	0,08816 ± 0,02825	0,99914 ± 0,00016
0,5	0,08947 ± 0,02186	0,99917 ± 0,00016
0,6	0,08684 ± 0,03540	0,99913 ± 0,00032
0,7	0,06711 ± 0,02528	0,99931 ± 0,00014
0,8	0,09737 ± 0,02510	0,99912 ± 0,00020
0,9	0,07763 ± 0,02387	0,99918 ± 0,00019

The results demonstrated that the best alpha is 0.8 which means that the models gives some importance (near to 1) to the negatives values in the dataset to get better classifications results. The value is 1% then the second best result the alpha 0.4. In the Figure 6.1 it is showed one of the tests with alpha=0.8, where accuracy achieved values above 15%. Also, it is possible to identify that these first models already present learning demonstrated by the final validation accuracy value when compared to the probability of a correct classification by chance, 1.36%. It is important to emphasize that in these tests it was only intended to get a comparison between the results for different tests and not already getting the best model. The probability of the improvement with the increase of epochs it's very likely to happen as the training accuracy continually increases and so the learning process.

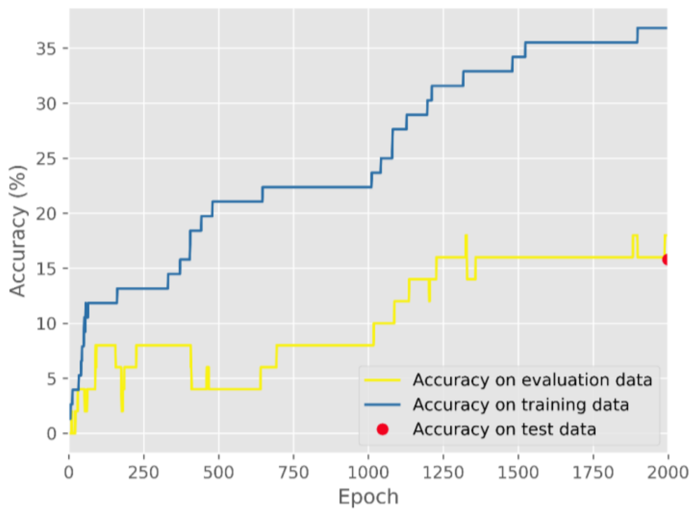


Figure 6.1. The best result of one of 3 tests for Leaky Relu with alpha equal to 0.8.

As mentioned before, the number of combinations in analysis are 135, wherein 15 models with different loss and activation function were tested for 9 learning rates. So, the analysis will be based in these 15 models and in their behaviour over the different learning rate values.

To be easier to manage models' results, it was assigned a letter to each model according to the hyper-parameters activation function and loss function as is in *Table 6.13*.

Table 6.13 – Lettering the combinations of activation and loss functions.

		Activation function				
		Sigmoid	*Tanh*	*Leaky Relu*	*Relu*	*Prelu*
Loss function	*Categorical cross entropy*	A	D	G	J	M
	Hinge	B	E	H	K	N
	Squared Hinged	C	F	I	L	O

Now relatively to results, it was verified by the training accuracy values that exist three different models' behaviours. Firstly, the model where the training accuracy achieved 100% or almost but the learning process continues to happen demonstrated by the continuous cost reducing. And the model is probably entering in the overfitting problem where all data main features were possible already learnt. Secondly, the models that due to slower learning still in the main learning process, wherein the training accuracy in general not is above of 85%. Lastly, the models that don't demonstrate learning or negative learning (last training accuracy is worse than other before), and for that not are important for more analysis. In the Table 6.14 are presented the number of models that are in each category.

Table 6.14. Description of the results regarding to learning behaviour obtained in test part a for 50 nodes parcellation.

	Full learning	**Learning**	**No learning**	**Total**
Number of models	32	39	64	135
Avg. training accuracy	0.9990 ± 0.0043	0.3800 ± 0.2897	0.0204 ± 0.0110	0.4361 ± 0.4031
Avg. training cost	0.1708 ± 0.3810	1.0644 ± 0.5760	7.9883 ± 7.5252	2.5964 ± 4.8948
Avg. validation accuracy	0.2148 ± 0.0440	0.0698 ± 0.0446	0.0143 ± 0.0035	0.0902 ± 0.0825
Avg. validation cost	3.5602 ± 1.0588	1.0900 ± 0.7532	8.0055 ± 7.5426	3.4172 ± 4.7469

With a brief analysis to table is possible to see that the majority of the models don't have interest because their learning curve is frozen. Although some learning had occurred proved by the mean of the values, as they are little larger than the probability of a random choice.

The full learning models group verifies that how should be for all models. The average training cost is lower comparatively to other results and training accuracy near or equal to 100%. And the average validation accuracy is larger than the other groups. Although the validation cost is very superior that the learning group the values can be explained by the majority functions that make up the group. The full learning group it's composed by cross-entropy function associated to higher values due the function nature, while for the learning group it was the Hinge function with less cost values associated. Also, it might have happened the overfitting problem in the full learning group increasing the costs in the validation data.

The models that characterize the first group had mainly in its parameters the categorical cross-entropy as loss function. Only four wasn't the case. So, it's possible to see that loss cross-entropy did a faster and better learning comparatively to others. In the four cases the function involved was Tanh activation function associated to other loss functions - models E and F. So, the Tanh loss function increases the learning velocity but the final results didn´t perform so well. The best result of this cases is a validation mean accuracy of 0.2395, for model E with learning rate equal to 0.3.

The first four best results had the same loss and activation function, the model A of the *Table 6.13*. And the best result, the maximum accuracy value and minimum cost, was for learning rate 0.1.

Then the most interesting models, as previously mentioned, comprehend mostly models with the tanh activation function - models D, E and F. By descending order, the performance was for each first case model, D-0.2763, E-0.2395 and F-0.2158. Additionally, as model A, the model D had the best result for learning rate equal 0.01. This two models, A and D was chosen directly for future use due the previous positive points mentioned before. Nevertheless, models in a learning process group won't cast off and so they will analyse deeply.

During the process it was searched models that had a fair value in the validation accuracy and a low accuracy on training data. Because in this combination could be the best models once they have a lot potential to improve, due to low learning velocity in addition to a good performance already proved. Thus, the models that stands out are the models composed by Hinge loss function - models B, H and K. The training accuracy average values are around 0.5 and validation accuracy average values around 0.1. These models presented a good characteristic, their costs are lower than models A and D, and so the difference between validation and training is small, situation that can be advantage to get models with better generalization. However, this could be related to the nature of the loss function, and the cost value lower not be traduced so linearly. In order to test and compare the models of the two categories it was increased the number of epochs to models training more and to learn more features from training data. Once, the models that can classify correctly all subjects in training data mean that all the main features are learned by the model. So, the training epochs were amplified for 15000. A larger number due the low velocity learning of this models. Also, the learning rates tested were the largest from the first learning rate set, once was those that presented learning. Situation caused by the necessity of this models by large learning rates and the new results proved that. But even with 15000 epochs the training accuracy is in general far from 1 (see Table 6.15). So, the accuracy values could increase once more are used more training epochs. The maximum accuracy happens for 0.5 learning rate for model B, where training accuracy is near to 1. Emphasising once again, these models verified that the costs are much near between training and validation. These tests were repeated three times.

Table 6.15. Average accuracy and cost values for training and validation data in the In-house dataset for 50 nodes parcellation, for model K, H and B in the part A.

Model	Learning rate	Average training accuracy	Average training cost	Average validation accuracy	Average validation cost
K	0.5	0.8158	0.9893	0.2105	0.9981
	0.05	0.6974	0.9908	0.1579	0.9986
	0.1	0.7368	0.9903	0.1316	0.9986
	0.3	0.6579	0.9913	0.1316	0.9987
	0.01	0.3684	0.9952	0.0658	0.9994
	0.1	0.6842	0.6842	0.1579	0.9987

H	0.3	0.7763	0.7763	0.1579	0.9987
	0.05	0.5526	0.5526	0.1316	0.9988
	0.5	0.8158	0.8158	0.1319	0.9984
	0.01	0.3421	0.3421	0.0526	0.9995
B	0.5	0.9956	0.9869	0.2500	0.9981
	0.3	0.9737	0.9872	0.1711	0.9985
	0.1	0.7281	0.9905	0.1404	0.9990
	0.05	0.5044	0.9936	0.1316	0.9993
	0.01	0.1447	0.9984	0.0570	0.9997

Due to the behaviour of models B, H and K it was performed a new set of tests where the learning rate was increased. Thus, the new set of learning rates was {1.2;1;0.9;0.7}. As the learning rate had increased, the number of epochs had diminished to 3000 epochs to fast training processes. Moreover, the model E was also tested, because it was the only activation function missing in the combination with hinge loss function. And the tests were repeated four times. The results obtained demonstrate that was good adding the model E because it was the model that performed better. The training accuracy achieved was 1 for any learning rate, need any less than 1000 epochs and had validation accuracies above 0.23. The final cost on training data are the same for all, but in the validation data, the low cost is for the model with better performance. Finally, the learning rate equal to 0.9 was the one that achieved the best results: 0.2796 in validation accuracy (Table 6.16 and .

Table *6.17*).

Table 6.16. Average accuracy and cost values for training data in the In-house dataset for 50 nodes parcellation and model E.

Learning rate	**Average training accuracy**	**Average epoch occurrence (max value)**	**Average training cost**
0.9	1.0 ± 0.0	926 ± 460,7	0.98684 ± 0.0
1.0	1.0 ± 0.0	883 ± 585,7	0.98684 ± 0.0
1.2	1.0 ± 0.0	751 ± 356,9	0.98684 ± 0.0
1.0	1.0 ± 0.0	904 ± 216,4	0.98685 ± 0.0

Relatively to other models, they were tested again three times but now with more number of epochs, 5000 epochs, to exist more training and so had in the final a more accurate conclusion. To the learning rate set was also added three new learning rates {1.5; 1.7; 2}. So, the set of learning rates tested was {2; 1.7; 1.5; 1.2; 1; 0.9; 0.7}.

Table 6.17. Average accuracy and cost values for validation data in the In-house dataset for 50 nodes parcellation and model E.

Learning rate	Validation average accuracy	Validation average cost
0.9	0.2796 ± 0.0235	0.9983 ± 0.0001
1.0	0.2730 ± 0.0599	0.9984 ± 0.0002
1.2	0.2632 ± .03355	0.9984 ± 0.0001
1.0	0.2467 ± 0.0253	0.9984 ± 0.0001

The results demonstrated that the six best results were achieved by model B (Sigmoid and Hinge activation functions). The other results were set aside for further tests. So, in the six models B the training accuracy was beyond the 0.9 or 90% and these results were achieved after in general 4000 epochs - consequence of the slower learning behaviour of these models (Table 6.18). All the training costs obtained were similar, but the best was for the second case with best validation accuracy. Validation results were quite positive, but once more the model in learning rate 1.5 got the large average accuracy, of course that the standard deviation is higher, 6%, which means that the values are very disparate from the average and so the values can be much lower but can also be much higher than the average. Confirmed by the average max accuracy value of 0.33 (Table 6.20). The costs were similar for all training data results, but was the learning rate 1.5 in the validation accuracy that achieved the minimum cost followed by the best result in average validation accuracy, learning rate equal to 1.7. Once in general the learning rate equal to 1.7 achieved performed better was the hyperparameter chosen for future use (Table 6.21).

Table 6.18. Average accuracy, average max accuracy and average epoch occurrence max values for training data in the In-house dataset for 50 nodes parcellation and model B, part A.

Learning rate	Average accuracy	Average max accuracy	Average epoch occurrence
1.7	0.9781± 0.0124	0.9781 ± 0.0124	3255.3333 ± 124.9462
1.5	1.0000 ± 0.0000	1.0000 ± 0.0000	4044.3333 ± 301.0497
2	0.9868 ± 0.0107	0.9868 ± 0.0107	4224.3333 ± 311.4935
1	0.9825 ± 0.0062	0.9825 ± 0.0062	4295.3333 ± 380.9500
1.2	0.9781 ± 0.0224	0.9781 ± 0.0224	4491.6667 ± 30.0624
0.7	0.9386 ± 0.0124	0.9386 ± 0.0124	264.3836 ± 0.0000

Table 6.19. Average cost, average max min cost and average epoch occurrence of the min values for training data in the In-house dataset for 50 nodes parcellation and model B, part A.

Learning rate	Average cost	Average min cost	Average epoch occurrence
1.7	0.9871 ± 0.0002	0.9871 ± 0.0002	5000.0000 ± 0.0000
1.5	0.9869 ± 0.0000	0.9869 ± 0.0 000	5000.0000 ± 0.0000
2	0.9870 ± 0.0001	0.9870 ± 0.0001	5000.0000 ± 0.0000
1	0.9871 ± 0.0001	0.9871 ± 0.0001	5000.0000 ± 0.0000
1.2	0.9871 ± 0.0003	0.9871 ± 0.0003	5000.0000 ± 0.0000
0.7	0.9877 ± 0.0002	0.9877 ± 0.0002	5000.0000 ± 0.0000

Table 6.20. Average accuracy, average max accuracy and average epoch occurrence max values for validation data in the In-house dataset for 50 nodes parcellation and model B, part A.

Learning rate	Average accuracy	Average max accuracy	Average epoch occurrence
1.7	0.2533 ± 0.0249	0.2733 ± 0.0000	2661.7000 ± 667.3482
1.5	0.2933 ± 0.0660	0.3333 ± 0.0680	3196.3000 ± 719.0338
2	0.2333 ± 0.0525	0.2667 ± 0.0499	3061.0000 ± 1673.2659
1	0.2667 ± 0.0680	0.3000 ± 0.0432	2928.0000 ± .4725
1.2	0.2667 ± 0.0249	0.2800 ± 0.0283	3935.0000 ± 253.5521
0.7	0.2267 ± 0.0340	0.2733 ± 0.0660	3540.0000 ± 253.5521

Table 6.21. Average cost, average max min cost and average epoch occurrence of the min values for validation data in the In-house dataset for 50 nodes parcellation and model B, part A.

Learning rate	Mean cost	Mean min cost	Mean epoch occurrence
1.7	0.9981 ± 0.0002	0.9981 ± 0.0002	5000 ± 0.0000
1.5	0.9980 ± 0.0003	0.9980 ± 0.0002	4809 ± 270.1000
2	0.9979 ± 0.0003	0.9979 ± 0.0002	4551 ± 635.4533
1	0.9982 ± 0.0003	0.9982 ± 0.0003	4934 ± 93.3381
1.2	0.9981 ± 0.0002	0.9981 ± 0.0002	4877 ± 173.4769
0.7	0.9983 ± 0.0002	0.9983 ± 0.0002	4754 ± 348.3679

Taking again the question of the other results relatively to model H and K, due the bad performances, new tests were done to prove that they are not a good model to use. So, as the models H and K performed better for higher learning rates, new tests were made with a new set of larger learning rates. The new learning rate set was {2; 2.5; 3; 3.5; 4}. But did not happened any improvements that excelled. The validation accuracy remained above 0.23 and it was not possible to identify any result in validation that stood out. Thus, this part was given as finished.

Consequently, the models chosen for future use were:

- Model A

Loss function = Categorical cross-entropy

Activation function = Sigmoid

Learning rate = 0.01

Learning behaviour is quick, so the number of epochs recommended are 1000 or more.

- Model B

Loss function = Hinge

Activation function = Sigmoid

Learning rate = 1.5

Learning behaviour is slow, so the number of epochs recommended are 4000 or more.

- Model D

Loss function = Categorical cross-entropy

Activation function = Tanh

Learning rate = 0.01

Learning behaviour is quick, so the number of epochs recommended are 1000 or more.

6.2.1.2. 50 Nodes Parcellation – Part B

Part B – Fine-tuning of batch size, optimizers and initializers

Part B – Batch size

The process continued the process made in the part 1 but now for the batch size. In this part was used the last three models, A, B and D to make the batch size tests. Thus, all the parameters were maintained for each model with only exception of the batch size.

Tested model parameters:

i. Batch size: {1; 2; 5; 10; 20}

Number of tests: 10

Results:

i. Model A

Table 6.22 - Average accuracy and average cost for training data in the In-house dataset for 50 nodes parcellation and model A, different batch sizes, part B.

Batch size	Average accuracy	Average Cost
1	1.0 ± 0.0	0.0087 ± 0.0001
5	1.0 ± 0.0	0.0626 ± 0.0009
2	1.0 ± 0.0	0.0196 ± 0.0002
10	1.0 ± 0.0	0.2008 ± 0.0067
20	1.0 ± 0.0	0.7812 ± 0.0244

Table 6.23 - Average accuracy, average max accuracy, average accuracy difference (final -20% training) and average epoch occurrence maximum values for validation data in the In-house dataset for 50 nodes parcellation and model A different batch sizes, part B.

Batch size	Average accuracy	Average accuracy diff (final -20%)	Average max accuracy	Average epoch occurrence
1	0.2780 ± 0.0227	0.0020 ± 0.0060	0.2900 ± 0.0241	156.8 ± 165.1
5	0.2740 ± 0.0237	0.0080 ± 0.0098	0.2980 ± 0.0227	166.2 ± 101.2
2	0.2480 ± 0.0370	0.0040 ± 0.0080	0.2700 ± 0.0349	67.5 ± 87.2
10	0.2240 ± 0.0332	0.0040 ± 0.0120	0.2680 ± 0.0402	238.8 ± 97.3
20	0.2120 ± 0.0285	- 0.0120 ± 0.0240	0.2520 ± 0.0392	345.5 ± 57.9

Table 6.24 - Average cost, average minimum cost, average accuracy difference (final –20% training) and average epoch occurrence minimum values for validation data in the In-house dataset for 50 nodes parcellation, different batch sizes, model A, part B.

Batch size	Average cost	Average cost diff (final -20%)	Average min cost	Average epoch occurrence
1	3.3758 ± 0.0477	-0.0119 ± 0.0013	3.3758 ± 0.0477	500.0000 ± 0.0000
5	3.4638 ± 0.0499	-0.0268 ± 0.0035	3.4633 ± 0.0499	498.6000 ± 1.6248
2	3.4299 ± 0.0679	-0.0164 ± 0.0024	3.4299 ± 0.0619	500.0000 ± 0.0000
10	3.5768 ± 0.0170	-0.0409 ± 0.0027	3.5768 ± 0.0170	499.9000 ± 0.3000
20	3.6994 ± 0.0579	-0.0587 ± 0.0047	3.6994 ± 0.0579	500.0000 ± 0.0000

ii. Model B

For model B there was a modification in the number of epochs used in the training for each batch size. Once this model has the characteristic of slow learning, therefore the larger the batch size will amplify that characteristic, but it could not be a disadvantage, since the model can model generalize better and reduce overfitting. So, to get about the number correct of epochs was done a series of attempts for each batch size, always with caution in the number of epochs increased for not make the training process too slow. In the next list are the results obtained:

i. Batch size 1 – 4000
ii. Batch size 2 – 8000
iii. Batch size 5 – 10000
iv. Batch size 10 – 25000
v. Batch size 20 – 40000

Thus, with this change in the training parameters for each batch size there was equality in the performance between models A, B and D due the different behaviours.

Table 6.25 - Average accuracy and average cost for training data in the In-house dataset for 50 nodes parcellation and model B, different batch sizes, part B.

Batch size	Average accuracy	Average Cost
1	0.9790 ± 0.0065	0.9871 ± 0.0001
20	0.9158 ± 0.0244	0.9880 ± 0.0003
10	0.9605 ± 0.0186	0.9874 ± 0.0003
2	0.9790 ± 0.0105	0.9871 ± 0.0001
5	0.9395 ± 0.0271	0.9877 ± 0.0003

Table 6.26 - Average accuracy, average max accuracy, average accuracy difference (final – 20% training) and average epoch occurrence maximum values for validation data in the In-house dataset for 50 nodes parcellation and model B different batch sizes, part B.

Batch size	Average accuracy	Average accuracy diff (final -20%)	Average max accuracy	Average epoch occurrence
1	0.2800 ± 0.0456	0.0320 ± 0.0299	0.3080 ± 0.0371	3093.8 ± 595.1
20	0.2120 ± 0.0204	0.0240 ± 0.0408	0.2400 ± 0.0127	31520.2 ± 5343.4
10	0.2280 ± 0.0588	0.0240 ± 0.0427	0.2920 ± 0.0271	18617.2 ± 4612.8
2	0.1360 ± 0.0150	0.0000 ± 0.0253	0.2280 ± 0.0271	2860.8 ± 1474.3
5	0.1440 ± 0.0265	-0.0320 ± 0.0371	0.2240 ± 0.0150	6327.2 ± 2683.0

Table 6.27 - Average cost, average minimum cost, average accuracy difference (final – 20% training) and average epoch occurrence minimum values for validation data in the In-house dataset for 50 nodes parcellation, different batch sizes, model B, part 2.

Batch size	Average cost	Average cost diff (final -20%)	Average min cost	Average epoch occurrence
1	0.9980 ± 0.0002	-0.0002 ± 0.0001	0.9980 ± 0.0002	3859.4000 ± 281.2
20	0.9986 ± 0.0001	-0.0001 ± 0.0001	0.9986 ± 0.0001	38377.8000 ± 2673.7
10	0.9984 ± 0.0002	-0.0002 ± 0.0001	0.9983 ± 0.0001	22362.4000 ± 3317.4
2	0.9988 ± 0.0001	-0.0001 ± 0.0001	0.9988 ± 0.0001	7217.0000 ± 1109.4
5	0.9988 ± 0.0001	-0.0001 ± 0.0001	0.9987 ± 0.0001	8963.8000 ± 990.90

i. Model D

Table 6.28 - Average accuracy and average cost for training data in the In-house dataset for 50 nodes parcellation and model D, different batch sizes, part B.

Batch size	Average accuracy	Average Cost
1	1.0000 ± 0.0000	0.0017 ± 0.0000
5	1.0000 ± 0.0000	0.0082 ± 0.0002
2	1.0000 ± 0.0000	0.0034 ± 0.0001
20	1.0000 ± 0.0000	0.0395 ± 0.0017
10	1.0000 ± 0.0000	0.0173 ± 0.0005

Table 6.29 - Average accuracy, average max accuracy, average accuracy difference (final – 20% training) and average epoch occurrence maximum values for validation data in the In-House dataset for 50 nodes parcellation and model D, different batch sizes, part B

Batch size	Average accuracy	Average accuracy diff (final -20%)	Average max accuracy	Average epoch occurrence
1	0.1980 ± 0.0260	0.0020 ± 0.0060	0.2020 ± 0.0244	159.4000 ± 131.8834
5	0.1940 ± 0.0457	0.0060 ± 0.0128	0.2020 ± 0.0477	256.7000 ± 143.2802
2	0.1800 ± 0.0200	0.0040 ± 0.0120	0.1800 ± 0.0200	164.5000 ± 117.5272
20	0.1860 ± 0.0380	0.0120 ± 0.0160	0.1900 ± 0.0431	373.0000 ± 88.1034
10	0.1580 ± 0.0384	0.0020 ± 0.0060	0.1620 ± 0.0395	208.2000 ± 153.0561

Table 6.30 - Average cost, average minimum cost, average accuracy difference (final – 20% training) and average epoch occurrence minimum values for validation data in the In-house dataset for 50 nodes parcellation, different batch sizes, model B, part B.

Batch size	**Average cost**	**Average cost diff (final -20%)**	**Average min cost**	**Average epoch occurrence**
1	3.6129 ± 0.1213	-0.0073 ± 0.0023	3.6129 ± 0.1213	500.0000 ± 0.0000
5	3.6937 ± 0.1299	-0.0123 ± 0.0023	3.6936 ± 0.1298	498.2000 ± 3.1875
2	3.6589 ± 0.1139	-0.0084 ± 0.0025	3.6589 ± 0.1139	500.0000 ± 0.0000
20	3.7510 ± 0.1161	-0.0198 ± 0.0029	3.7510 ± 0.1161	500.0000 ± 0.0000
10	3.7412 ± 0.1030	-0.0142 ± 0.0016	3.7412 ± 0.1030	499.8000 ± 0.4000

Results' Discussion

In this test is exhibited besides the validation results the training results. The intention is to demonstrate that all the different models are on equal terms. This is, if the training accuracy is the same for all in order to compare if the models finish in a stage that the learning process is a similar for all models. Thus, due to exist different velocities in learning the train parameters was adapted.

Model A

The training results demonstrated that the accuracy on training data was 100 % for all the cases with different cost, directly proportional to batch size (*Table 6.22*). Also, a first look to validation results it's possible to evidence that the best max accuracy epoch occurs not in the last epoch of training, so there was already happening overtrain causing overfitting (*Table 6.23*).

In the general situation, the batch size 1 and 5 was the performances that more highlighted. But was the batch size 1 that had the best results in all domains, accuracy and cost in the validation data. Batch size 5 followed the best result but has more cost than batch size 2. However, the model got the same final result in the validation accuracy than batch size 1. And even the maximum value achieved was also the highest (*Table 6.23* and *Table 6.24*). So, in the next tests will be used the batch size 1 and 5 for model A. It's not used only the best result because is important to check in future situations if the batch size 5 continues to produce models with less performance.

Model B

In contrast to model B, in any training accuracy result was not accomplished 1, but all values were superior to 0.9 besides the great divergence in the learning rates. So, the adjustments did to epochs

according to batch size was well done. Again, the batch size 1 get the best performance in all domains and the batch size 10 was the second-best result. So, the batch sizes chosen for model B was 1 and 10 (*Table 6.25*,
Table 6.26 and *Table 6.27*).

Model D

As the model A, the training accuracy achieved 1 in all training accuracies and the batch size 1 got once more the best results in all domains. In second place stayed the batch size 5 that also stands out with accuracy values near to obtained by batch size 1. Thus, the final batch sizes for model D chosen was 1 and 5 (*Table 6.28*, *Table 6.29*, *Table 6.30*).

Results summarized:

Model A – batch size 1 and 5

Model B – batch size 1 and 10

Model D – batch size 1 and 5

Part B – Optimizers

The optimizers were tested for the three models (A, B and D) with different batch sizes. The optimizer's tested was: stochastic gradient descendent (SGD), Adadelta, RmsProp, Adam and Nadam.

The tests were made each time for each model. The process beginning testing the SGD optimizer, with momentum and then the best results with the Nesterov accelerated gradient (NAG). Finalized this process, followed a final test with the best test of SGD and the other the other four optimizers. For each case was made 10 tests.

i. Model A

a. SGD optimizer and momentum

Table 6.31 – Average validation accuracy and average cost for the model A with SGD and different momentums, for batch size 1 and 5, part B.

Batch size	Momentum	Average Validation accuracy	Average Validation Cost
1	0.5000	0.2395 $\pm$ 0.0202	3.4220 $\pm$ 0.0716
5	0.5000	0.2342 $\pm$ 0.0184	3.4470 $\pm$ 0.0619
1	0.0000	0.2303 $\pm$ 0.0179	3.4558 $\pm$ 0.0671
1	0.0500	0.2303 $\pm$ 0.0188	3.4383 $\pm$ 0.0398

1	0.1000	0.2263 ± 0.0268	3.4528 ± 0.0429
1	0.0100	0.2237 ± 0.0263	3.4021 ± 0.0341
1	0.0050	0.2224 ± 0.0266	3.4258 ± 0.0581
1	0.0010	0.2224 ± 0.0190	3.4486 ± 0.0457
5	0.3000	0.2197 ± 0.0300	3.4823 ± 0.0425
1	0.3000	0.2184 ± 0.0229	3.4712 ± 0.0417
5	0.0000	0.2171 ± 0.0244	3.5193 ± 0.0698
5	0.0100	0.2118 ± 0.0199	3.4988 ± 0.0440
5	0.1000	0.2092 ± 0.0246	3.5282 ± 0.0455
5	0.0500	0.2079 ± 0.0275	3.5093 ± 0.0413
5	0.0010	0.2026 ± 0.0237	3.4844 ± 0.0433
5	0.0050	0.1987 ± 0.0199	3.5163 ± 0.0429

The results tests demonstrated that batch size 1 is the best option in using the SGD with momentum and no momentum. And in general, that batch size had better performance, since all eight models with batch size 1 are in the 10 first results. Looking for the average validation cost its possible see that the batch size 5 is in general associated to the higher costs. But this can be only a consequence of the model trained with higher batch sizes has a slower learning, causing a higher cost for the same number of training epochs. Although it can result in a better generalization as happened in the second case for batch size 5. The final choice was done between the two first results for different batch sizes but with the same momentum, 0.5.

b. SGD final test adding NAG.

Table 6.32 - Average validation accuracy and coverage cost for the model A with SGD and momentum equal to 0.5 for batch size 1 and 5, part B.

Batch size	**Nesterov**	**Average validation accuracy**	**Average validation cost**
1	False	0.2474 ± 0.0115	3.3954 ± 0.0559
1	True	0.2303 ± 0.0318	3.4466 ± 0.0407
5	False	0.2197 ± 0.0306	3.4992 ± 0.0360
5	True	0.2158 ± 0.0244	3.4817 ± 0.0381

The results demonstrated that in this relation of the two training parameters, the Nesterov didn't affect positively the result.

c. SGD final and the other optimizers

Table 6.33 - Average validation accuracy and cost for the model A with the different optimizers and batch size 1 and 5, part B.

Optimizer	Batch size	Average validation accuracy	Average validation cost
RMSprop	1	0.3632 ± 0.0244	3.9922 ± 0.1658
RMSprop	5	0.3342 ± 0.0244	3.8136 ± 0.0830
Adadelta	1	0.3395 ± 0.0175	3.2641 ± 0.0666
Adadelta	5	0.3395 ± 0.0419	3.1775 ± 0.0478
Adam	5	0.3184 ± 0.0255	3.9299 ± 0.0645
Nadam	5	0.3184 ± 0.0327	4.1089 ± 0.1826
Adam	1	0.3105 ± 0.0105	4.4400 ± 0.1076
Nadam	1	0.2605 ± 0.0305	5.3366 ± 0.3930
SGD*	1	0.2474 ± 0.0115	3.3954 ± 0.0559

*momentum = 0.5

The results showed that the RMSprop and Adadelta are the best optimizers to use once their results was distanced from the others in the validation results for both batch sizes, although the Adam optimizer have better results in the costs in relation to the first result. The models chosen for future tests was RMSprop and Adadelta.

ii. Model B

a. SGD optimizer and momentum

Table 6.34 - Average validation accuracy and average cost for the model B with SGD and different momentums, for batch size 1 and 10, part B.

Batch size	Momentum	Average validation accuracy	Average validation Cost
1	0.3000	0.2421 ± 0.0105	0.9980 ± 0.0001
10	0.5000	0.2368 ± 0.0300	0.9981 ± 0.0001
1	0.1000	0.2368 ± 0.0186	0.9981 ± 0.0001
1	0.0050	0.2316 ± 0.0179	0.9982 ± 0.0001
1	0.5000	0.2237 ± 0.0186	0.9979 ± 0.0002
1	0.0000	0.2211 ± 0.0419	0.9982 ± 0.0001
1	0.0010	0.2184 ± 0.0197	0.9982 ± 0.0001
10	0.1000	0.2158 ± 0.0105	0.9982 ± 0.0001
1	0.0500	0.2105 ± 0.0204	0.9982 ± 0.0000

1	0.1000	0.2026 ± 0.0214	0.9980 ± 0.0001
10	0.0100	0.2000 ± 0.0316	0.9984 ± 0.0001
10	0.0500	0.1947 ± 0.0226	0.9984 ± 0.0001
10	0.0000	0.1895 ± 0.0105	0.9984 ± 0.0001
10	0.0050	0.1842 ± 0.0399	0.9986 ± 0.0001
10	0.3000	0.1842 ± 0.0250	0.9983 ± 0.0001
10	0.0010	0.1842 ± 0.0546	0.9984 ± 0.0002

The results demonstrated once more that momentum increase the performance in comparison a model with no momentum. And in the two best results was present the two batch sizes although the batch size 1 occupied practically all the first places. The final selection comprises the model with batch size 1 and momentum 0.3 and the model with batch size 10 and momentum 0.5.

b. SGD final test adding NAG.

Table 6.35 - Average validation accuracy and cost for the model B with SGD, for batch size 1 and 5, momentum 0.5 and 0.3, with NAG or not, part B.

Batch size	Nesterov	Momentum	Average validation accuracy	Average validation Cost
10	True	0.5000	0.2276 ± 0.0132	0.9981 ± 0.0001
1	True	0.5000	0.2211 ± 0.0234	0.9981 ± 0.0001
1	False	0.3000	0.2197 ± 0.0264	0.9980 ± 0.0001
10	False	0.3000	0.2066 ± 0.0283	0.9982 ± 0.0001

The performance of model B using SGD can be maximized by combination of Nesterov and momentum once they got the best results. Additionally, the batch size 10 was best than the batch size 1, although the accuracy difference was not so great, but both models was selected for future tests. If the analysis is done only based in the cost, it was the batch size 1 and with no use of NAG to be chosen.

c. SGD final and the other optimizers

Table 6.36 - Average validation accuracy and cost for the model B with the different optimizers and batch size 1 and 10, part B.

Optimizer	Batch size	Learning rate	Average validation accuracy	Average validation cost
RMSprop	10	0.0010	0.3500 ± 0.0197	0.9971 ± 0.0002
Adadelta	10	150.0000	0.3237 ± 0.0244	0.9971 ± 0.0001
Adam	10	0.0010	0.3132 ± 0.0357	0.9970 ± 0.0001
Nadam	10	0.0020	0.3026 ± 0.0235	0.9976 ± 0.0001
RMSprop	1	0.0010	0.2842 ± 0.0295	0.9977 ± 0.0002
Adadelta	1	1.0000	0.2790 ± 0.0268	0.9982 ± 0.0001
Adam	1	0.0010	0.2737 ± 0.0367	0.9975 ± 0.0002
SGD*	1	0.0100	0.2276 ± 0.0132	0.9981 ± 0.0001
Adadelta	10	1.0000	0.1947 ± 0.0403	0.9987 ± 0.0002
Adadelta	1	150.0000	0.1947 ± 0.0293	0.9977 ± 0.0002
Nadam	1	0.0020	0.0842 ± 0.0329	0.9989 ± 0.0004
RMSprop	1	0.0150	0.0132 ± 0.0000	0.9998 ± 0.0000
RMSprop	10	0.1500	0.0132 ± 0.0000	0.9998 ± 0.0000
Nadam	10	0.3000	0.0132 ± 0.0000	0.9998 ± 0.0000
Nadam	1	0.3000	0.0132 ± 0.0000	0.9998 ± 0.0000
Adam	1	0.1500	0.0132 ± 0.0000	0.9998 ± 0.0000
Adam	10	0.1500	0.0132 ± 0.0000	0.9998 ± 0.0000

*momentum 0.3 with NAG

The objective value was achieved for model B by three optimizers with batch size 10. But there are significate gaps between the different performances. The first result was achieved once more by RMSprop for the default leaning rate of the optimizer. But on contrary the cost was the large value between the set of three. The reason why this happen could be explicated by the different learning processes in each case. The RMSprop has the faster learning what can lead to more overfitting, although the training costs be the same apparently. The second result was achieved for Adadelta, for a learning rate calculated linearly taking into account the optimal learning rate by the mode in SGD and the default values of Adadelta. For last, it was the Adam optimizer with about less 0.01 validation accuracy than Adadelta. Therefore, this was the three optimizers and learning rates chosen for the next test processes.

iii. Model D

a. SGD optimizer and momentum

Table 6.37 - Average validation accuracy and average validation cost for the model D with SGD and different momentums, for batch size 1 and 5, part B.

Batch size	**Momentum**	**Average validation accuracy**	**Average validation cost**
1	0.0050	0.2184 ± 0.065	3.5353 ± 0.0512
1	0.1000	0.2158 ± 0.0229	3.5135 ± 0.0503
1	0.0000	0.2132 ± 0.0153	3.5644 ± 0.0655
1	0.0500	0.2079 ± 0.0305	3.6680 ± 0.0347
1	0.0100	0.2053 ± 0.0349	3.6319 ± 0.0772
5	0.0000	0.2053 ± 0.0404	3.5692 ± 0.0628
5	0.1000	0.2053 ± 0.0318	3.6361 ± 0.0487
5	0.5000	0.2026 ± 0.0244	3.5896 ± 0.0631
1	0.5000	0.2000 ± 0.0175	3.6860 ± 0.1681
5	0.0050	0.1947 ± 0.0241	3.6734 ± 0.1031
5	0.0010	0.1921 ± 0.0359	3.6249 ± 0.0666
5	0.3000	0.1895 ± 0.0283	3.6800 ± 0.1013
1	0.0010	0.1868 ± 0.0153	3.5578 ± 0.0787
1	0.3000	0.1868 ± 0.0542	3.6071 ± 0.1233
5	0.0500	0.1790 ± 0.0197	3.6642 ± 0.0589
5	0.0100	0.1763 ± 0.0244	3.6541 ± 0.0719

A brief analysis of the results permitted conclude that the best momentum was 0,005 and followed by 0.1 and then 0. But all validation accuracy values are near between them, so they were the selected momentum set. In this set it was momentum 0.1 that achieve minimum costs in training, validation data.

b. SGD final test adding NAG.

An analysis to results verify two points. One was the confirmation of the conclusion for batch 1 in the previous set of tests, wherein the Nesterov accelerated gradient in general has improved the model’s performance. And second it was that momentum 0.005 that got the best results with no use of Nesterov accelerated gradient, which was the hyperparameter used in the final test.

Table 6.38 - Average validation accuracy and cost for the model D with SGD, for batch size 1, momentum 0.05, 0.1 and 0, with NAG or not, part B.

Momentum	Nesterov	Average accuracy	Average Cost
0.005	False	0.2329 ± 0.0295	3.5139 ± 0.0816
0.1	True	0.2237 ± 0.0390	3.5720 ± 0.1434
0.005	True	0.2158 ± 0.0318	3.5551 ± 0.0987
0	True	0.2040 ± 0.0302	3.5987 ± 0.0889
0	False	0.1961 ± 0.0199	3.6280 ± 0.0368
0.1	False	0.1908 ± 0.0251	3.6152 ± 0.0936

c. SGD final and the other optimizers

Table 6.39 - Average validation accuracy and cost for the model D with the different optimizers and batch size 1 and 5, part B.

Optimizer	Batch size	Average validation accuracy	Average validation cost
RMSprop	1	0.3684 ± 0.0166	3.2112 ± 0.0453
Adadelta	1	0.3447 ± 0.0226	3.2908 ± 0.1173
RMSprop	5	0.3421 ± 0.0363	3.6193 ± 0.0592
Nadam	5	0.3263 ± 0.0367	3.9140 ± 0.1536
Adam	5	0.3263 ± 0.0327	3.5427 ± 0.1110
Adadelta	5	0.2974 ± 0.0295	3.3480 ± 0.0679
Adam	1	0.2921 ± 0.0099	4.2471 ± 0.1465
SGD 0.005 momemtum	1	0.2329 ± 0.0295	3.5139 ± 0.0818
Nadam	1	0.0316 ± 0.0179	9.7072 ± 1.3018

From the validation results, show, again that RMSProp is the best optimizer and in this case with batch size 1. In second and third, it was respectively the Adadelta for batch size 1 and once again RMSprop but now for batch size 5, with similar performance accuracies but with different costs. So they was the three results chosen for the future tests.

Part B - Initializers

In this part was tested the different initializers. It was tested a considerable amount of initializers due to its importance in the learning process. Since they have a great effect in how the model converges.

Initializers tested:

i. Zero
ii. Random
iii. Random normal between -1 and 1
iv. Random uniform between -1 and 1
v. Truncated Normal with standard deviation equal to 0.05 and 0.
vi. Variance Scalling using the average number of inputs and output nodes and, the total of inputs and outputs nodes.
vii. Gaussian and standard deviation equal to 0.01
viii. Glorot Normal
ix. Glorot uniform
x. He normal
xi. 11.Orthogonal

Number of tests = 10

In this test was followed a procedure divided in three parts, the first part was test the weights for each initializer and the bias was initialized a zero, and selected the best results, then was made the inverse test for bias initializing the weights as 0 and selected the best initializers. For last was done a final test with the combination of the test results for each model. The epochs also were adjusted to the learning behaviour, wherein slower learnings were trained using more epochs.

In order to understand what models were used from the last results was created a table for each main parameters set model A, B and D, which contains the best other models with more specific parameters, resulting from last tests of part B (*Table 6.40*, *Table 6.41*, *Table 6.42*).

Models summary:

i. Model A

Table 6.40 – Lettering models of the last results for the main model A of the part A.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Model 1A (lr = 0.001)	Model 2A (lr = 0.001)
	Adadelta	Model 3A (lr = 1.0)	Model 4A (lr = 1.0)

ii. Model B

Table 6.41 - Lettering models of the last results for the main model B of the part A.

		Batch size
		10
Optimizer	*RMSprop*	Model 1B (lr = 0.001)
	Adadelta	Model 2B (lr = 150)
	Adam	Model 3B (lr = 0.001)

iii. Model D

Table 6.42 - Lettering models of the last results for the main model D of the part A.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Model 1D (lr = 0.001)	Model 2D (lr = 0.001)
	Adadelta	Model 3D (lr = 1.0)	

In order to reduce the number of final models and get only the best results it was only selected the models that achieved more than 0.34 in the validation accuracy. In addition, it was also selected the first model below the performance line to in the selected set includes all three different models of part A, which means the model A, B and D. Thus, in the selected models was included the models that achieved the best validation accuracy and the second validation cost.

Table 6.43 - Average validation accuracy and cost obtained in the final step for each pair of best initializers for the 50 nodes parcellation.

Case	**Batch size**	**Kernel initializer**	**Bias initializer**	**Average validation accuracy**	**Average validation cost**
1D	1	Glorot uniform	Random uniform	0.3816 ± 0.0263	3.0394 ± 0.0830
2D	5	Random Normal (±0.01)	He normal	0.3513 ± 0.0221	3.4099 ± 0.1010
3D	1	Glorot uniform	VS fan avg	0.3513 ± 0.0289	2.9461 ± 0.0690
1A	5	Glorot uniform	Random Normal	0.3434 ± 0.0266	3.6335 ± 0.0897

3B	10	Variance scalling "fan_avg"	Truncated Normal	0.3382 ± 0.0312	0.9971 ± 0.0002
3A	1	Random Normal ($\pm$0.01)	Random Normal	0.3355 ± 0.0088	3.0725 ± 0.0575
2A	1	Glorot uniform	Random Normal	0.3329 ± 0.0257	3.6289 ± 0.1287
4A	5	Random Normal ($\pm$0.01)	Random uniform	0.3290 ± 0.0166	3.0560 ± 0.0507
1B	10	Glorot uniform	He uniform	0.3276 ± 0.0231	0.9968 ± 0.0001
2B	10	Glorot uniform	Random Normal	0.2816 ± 0.0244	0.9973 ± 0.0001

Part 2 – Initializers additional tests - Categorical hinge and Softsign

Besides the last test was made an additional set of tests in order to verify the effect of categorical Hinge loss function and Softsign activation function. So, to do that, using the principal models (models A and D) and parameters already the fine-tuned parameters for each case was created three more main possibilities. The model B was not used once it's presented a slower learning, needing more time to train. One changing the loss function of the models by categorical Hinge, other replacing the activation function by Softsign and in for last replacing both in the models.

So summarized was created for each situation the models in the (*Table 6.44*,*Table 6.45*,*Table 6.46*,*Table 6.47*,*Table 6.48* and *Table 6.49*).

i. Replacing model A loss function by Categorical Hinge loss

Table 6.44 – Lettering models for the new results using the categorical Hinge loss function with model A parameters.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Test1 (lr = 0.001)	Test2 (lr = 0.001)
	Adadelta	Test3 (lr = 1.0)	Test 4 (lr = 1.0)

ii. Replacing model D loss function by Categorical Hinge loss

Table 6.45 - Lettering models for the new results using the categorical Hinge loss function with model D parameters.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Test5 (lr = 0.001)	Test6 (lr = 0.001)
	Adadelta	Test7 (lr = 1.0)	

iii. Replacing model A activation function by Softsign

Table 6.46 – Lettering models for the new results using the Softsign activation function with model A parameters.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Test8 (lr = 0.001)	Test9 (lr = 0.001)
	Adadelta	Test10 (lr = 1.0)	Test11 (lr = 1.0)

iv. Replacing model D activation functon by Softsign

Table 6.47 - Lettering models for the new results using the Softsign activation function with model D parameters.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Test12 (lr = 0.001)	Test13 (lr = 0.001)
	Adadelta	Test14 (lr = 1.0)	

v. Replacing model A loss function by Categorical Hinge loss and activation function by Softsign

Table 6.48 – Lettering models for the new results using the Softsign activation function and categorical Hinge loss function with model A parameters.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Test15 (lr = 0.001)	Test16 (lr = 0.001)
	Adadelta	Test17 (lr = 1.0)	Test18 (lr = 1.0)

vi. Replacing model D loss function by Categorical Hinge loss and activation function by Softsign

Table 6.49 - Lettering models for the new results using the Softsign activation function and categorical Hinge loss function with model A parameters, part 2.

		Batch size	
		1	*5*
Optimizer	*RMSprop*	Test19 (lr = 0.001)	Test20 (lr = 0.001)
	Adadelta	Test21 (lr = 1.0)	

Table 6.50 - Average validation accuracy and cost obtained for the the new tests with categorical Hinge loss function and Softsign activation function using as base the models A and D, for the 50 nodes parcellation.

Test	**Batch size**	**Kernel initializer**	**Bias initializer**	**Average accuracy**	**Average Cost**
Test15	1	Glorot uniform	Random Normal	0.3829 ± 0.0340	0.9674 ± 0.0241
Test8	1	Glorot uniform	Random Normal	0.3790 ± 0.0262	3.0472 ± 0.0750
Test17	1	Glorot normal	Random uniform	0.3671 ± 0.0208	0.9891 ± 0.0210
Test20	1	Random Normal (±0.01)	He normal	0.3658 ± 0.0281	0.9923 ± 0.0120
Test19	5	Glorot uniform	Random Normal	0.3540 ± 0.0350	1.0336 ± 0.0279
Test17	1	Random Normal (±0.01)	Random normal	0.3540 ± 0.0224	0.9928 ± 0.0255
Test7	1	Zeros	Vs fan avg	0.3527 ± 0.0255	1.0094 ± 0.0187
Test21	5	Random Normal (±0.01)	Random uniform	0.3487 ± 0.0188	0.9963 ± 0.0089
Test10		Random Normal (±0.01)	Random Normal	0.3434 ± 0.0216	2.9157 ± 0.0717
Test9	5	Glorot uniform	Random Normal	0.3382 ± 0.0300	3.4784 ± 0.0999
Test11	5	Random Normal (±0.01)	Random uniform	0.3355 ± 0.0251	2.9551 ± 0.0470
Test18	5	Random Normal (±0.01)	He normal	0.3316 ± 0.0327	1.0500 ± 0.0259
Test12	10	Glorot uniform	He uniform	0.3303 ± 0.0279	0.9967 ± 0.0002
Test14	10	VS fan avg	Truncated Normal	0.3197 ± 0.0250	0.9967 ± 0.0001
Test3	1	Random Normal (±0.01)	Random Normal	0.3132 ± 0.0164	1.0326 ± 0.0146
Test4	5	Random Normal (±0.01)	Random uniform	0.3040 ± 0.0472	1.0303 ± 0.0148
Test2	5	Glorot uniform	Random Normal	0.2987 ± 0.0372	1.0782 ± 0.0555
Test6	5	Random Normal (±0.01)	He normal	0.2882 ± 0.0199	1.1091 ± 0.0294
Test13	10	Glorot uniform	Random Normal	0.2842 ± 0.0301	0.9987 ± 0.0001
Test1	1	Glorot uniform	Random Normal	0.1908 ± 0.0391	1.0668 ± 0.0260
Test5	1	Glorot normal	Random uniform	0.0382 ± 0.0266	1.6096 ± 0.2002

The results were very positively in general, wherein the model achieved better results comparatively to the other initializers' results in *Table 6.50*. As example, the new results have 7 models with accuracy above 0.35 while the other results have only 4 models. Also, the average validation costs are better for the new results. Besides the combination Softsign activation function and the categorical Hinge demonstrated be a good relation, once it achieved the best results over all the results created in the initializers test. The models' selection was done using all the models that accomplished a validation accuracy above 0.35.

Final models - Case1D, Case 2D, Case 3D, Case 1A, Case 3B, Test15, Test8, Test17, Test20, Test16, Test19 and Test7.

Then, before the next process was done additional test to make a learning rate adjustment, wherein the learning rate of the final models was multiplying by the factors {0,01;0,05;0,1;0,5;1}, and made again 10 tests for each case.

Table 6.51 – Average validation accuracy and cost for the learning rate adjustment with the final results of part 2, for 50 nodes parcellation.

Test/case	**Initial Learning rate**	**New best learning rate**	**Average validation accuracy**	**Average validation Cost**
Case 1D	0.0010	0.0001	0.4105 ± 0.0193	2.7445 ± 0.0448
Test8	0.0010	0.0001	0.4013 ± 0.0197	2.8736 ± 0.0577
Case 2D	0.0010	0.0001	0.3947 ± 0.0243	2.9885 ± 0.0798
Test16	0.0001	0.0001	0.3803 ± 0.0308	0.9792 ± 0.0228
Test15	0.0010	0.0001	0.3763 ± 0.0468	0.9850 ± 0.0171
Test16	0.0001	0.0005	0.3693 ± 0.0286	0.9962 ± 0.0199
Case 3D	1.0000	0.5000	0.3645 ± 0.0243	2.9251 ± 0.0647
Test20	1.0000	0.5000	0.3645 ± 0.0289	0.9797 ± 0.0191
Test19	1.0000	0.5000	0.3638 ± 0.0251	0.9836 ± 0.0143
Test7	1.0000	0.1000	0.3618 ± 0.0258	0.9872 ± 0.0133
Case 1A	0.0010	0.0001	0.3500 ± 0.0265	3.3507 ± 0.0762
Case 3B	0.0010	0.0001	0.3263 ± 0.0275	0.9968 ± 0.0001

The new validation results demonstrated that the adjustment of the learning rate was a good approach since the majority of the learning rates has been updated. Besides the performances of the models in the validation data has improved, in the accuracy and in the cost. Before this new test, the model only had two models over 0.38 in the accuracy, and after the number increase to four models. It was selected for the next phase the models with an accuracy over 0.37 in the test accuracy.

The performances metrics from this step to the next decrease in general 0.1. The reason was the use of partial data with no temporal filtering. This problem led us to discover that the temporal filtering is eliminating some important features. After being identified this problem the data was corrected and the values diminish. This created an assumption that the temporal filtering eliminates important features, which will be later explored.

6.2.1.3. 50 Nodes Parcellation – Part C

Part C – Change/Increase the models' depth

Actual model architecture:

3 layers:

i. Inputs 4278 nodes
ii. Hidden layer 200 nodes
iii. Outputs 76 nodes (number of subjects to classify)

Table 6.52 - Average validation accuracy and cost for the learning rate adjustment with the final results of part 2, for 50 nodes parcellation.

Test/case	Batch size	Kernel initializer	Bias initializer	Activation function	Loss Function	Optimizer	Learning rate
Case 1D	1	Glorot uniform	Random uniform	Tanh	Categorical crossentropy	RMSprop	0.0001
Test8	1	Glorot uniform	Random Normal	Softsign	Categorical cross entropy	RMSprop	0.0001
Case 2D	5	Random Normal stddev 0,01	He normal	Tanh	Categorical crossentropy	RMSprop	0.0001
Test17	1	Glorot normal	Random uniform	Softsign	Categorical Hinge	RMSprop	0.0001
Test15	1	Glorot uniform	Random Normal	Softsign	Categorical Hinge	RMSprop	0.0001

i. 3 Layers

The first tests were done for the model with 3 layers, changing the hidden layer nodes with values of the following set:

Nodes set – {50; 100; 200; 500; 1000; 2000; 4000; 8000; 10000}

Table 6.53 – Average validation accuracy and cost for the best models, one hidden layer test, In-House data and parcellation 50 nodes.

Model	Average validation accuracy	Average validation cost
Case 1D	0.3132 ± 0.0129	3.1422 ± 0.0668
Test8	0.3053 ± 0.0255	3.2547 ± 0.0334
Case 2D	0.3000 ± 0.02680	3.1223 ± 0.0389
Test17	0.2895 ± 0.0300	1.0405 ± 0.0234
Test15	0.2921 ± 0.0255	1.0449 ± 0.0251

The best results were obtained for the hidden layer with 200 nodes for all models, and the results are summarized in *Table 6.53*. Then was made a set of tests with dropout, for the next set of rates {0.1; 0.2; 0.5; 0.7} but the results haven't improved.

ii. 4 Layers

a. Maintaining the number of parameters:

The total of parameters for the 200 nodes was 871000. So, with 4 layers the number of the nodes in the middle layers was chosen in order to maintain the total number of parameters. Thus, it were chosen 3 different values for the second layer of the model, and calculated the value for the third layer. The values chosen was 50, 100 and 150 nodes, and to respect the assumption the width in the other hidden-layer was respectively 5000, 2500 and 1000 nodes.

Table 6.54 - Average validation accuracy and cost for the best models, two hidden layer test and maintaining the number of parameters, parcellation 50 nodes.

Model	Average validation accuracy	Average validation cost	Hidden layer 1	Hidden layer 2
Case 1D	0.2658 ± 0.0129	3.8950 ± 0.0806	100	2500
Test8	0.2684 ± 0.0105	3.5108 ± 0.0486	150	1000
Case 2D	0.2132 ± 0.0316	5.2878 ± 0.2114	150	1000
Test17	0.2369 ± 0.0186	1.0856 ± 0.0197	150	1000
Test15	0.2369 ± 0.0322	1.0960 ± 0.0181	150	1000

The results obtained was poorly when compared with the best results. Globally, it was for the combination of 150 nodes in the hidden layer 1 and 1000 in the hidden layer 2. So, there was an indication that are required higher values in the second layer and not so high in the third layer.

b. Variating the number of nodes in the hidden layers model:

Values tested combined:

i. Hidden-layer 1: 100, 500, 1000 and 2000
ii. Hidden-layer 2: 100, 500, 1000 and 2000

The best results are demonstrated in the *Table 6.55*, where the test8, test14 and test15 have improved their performance. All of these models have hyperparameter in common, the Softsign activation function or Categorical Hinge loss function.

Table 6.55 - Average validation accuracy and cost for the best models, two hidden layer test and variating nodes, In-House data and parcellation 50 nodes.

Model	**Average validation accuracy**	**Average validation cost**	**Hidden layer 1**	**Hidden Layer 2**
Case 1D	0.2829 ± 0.0197	3.4141 ± 0.0617	2000	500
Test8	0.3105 ± 0.0271	1.0183 ± 0.0083	2000	1000
Case 2D	0.2893 ± 0.0144	1.0230 ± 0.0128	2000	1000
Test17	0.3079 ± 0.0134	3.1254 ± 0.0613	2000	500
Test15	0.2895 ± 0.0186	3.2189 ± 0.0412	2000	500

5 Layers

a. Add layer to the last better set of hyperparameters:

In this test was added to the last best four layers models a new layer that could have 200, 500 or 1000 nodes.

Table 6.56 - Average validation accuracy and cost for the best models, three-layer test and best parameters, In-House data and parcellation 50 nodes.

Model	**Average validation accuracy**	**Average validation cost**	**Hidden layer 1**	**Hidden Layer 2**	**Hidden layer 3**
Case 1D	0.3026 ± 0.0416	3.2601 ± 0.1062	2000	500	200
Test8	0.2947 ± 0.0214	3.3032 ± 0.0598	2000	1000	500
Case 2D	0.2684 ± 0.0134	3.8723 ± 0.1767	2000	1000	200
Test17	0.3105 ± 0.0283	1.0174 ± 0.0041	2000	500	1000
Test15	0.2921 ± 0.0411	1.0312 ± 0.0118	2000	500	500

Once again, the test17 has improved its results. (*Table 6.56*).

a. Variating the hidden layers nodes:

Table 6.57 - Average validation accuracy and cost for the best models, three-layer test and variating parameters, In-House data and parcellation 50 nodes.

Model	**Average validation accuracy**	**Average validation cost**	**Hidden layer 1**	**Hidden Layer 2**	**Hidden layer 3**
Case 1D	0.3026 ± 0.0144	3.3350 ± 0.6467	2000	2000	200
Test8	0.2816 ± 0.0258	3.2345 ± 0.6360	2000	2000	2000
Case 2D	0.3053 ± 0.0293	3.6108 ± 0.6480	2000	2000	200
Test17	0.2632 ± 0.0186	1.1043 ± 0.6267	2000	200	200
Test15	0.2711 ± 0.0134	1.1239 ± 0.6307	2000	200	200

In this test was tested the combination of the hypothesis 200 and 2000 nodes for each layer. And all the models performed worse than in previous tests (*Table 6.57*). And search for nodes stopped here.

Finals best models:

Table 6.58 – Final best models' architecture for In-House data and 50 nodes parcellation.

Model	**Hidden layer 1**	**Hidden Layer 2**	**Hidden layer 3**	**Average validation accuracy**	**Average validation cost**
Case 1D	200	-	-	0.3132 ± 0.0129	3.1422 ± 0.0668
Test8	2000	1000	-	0.3105 ± 0.0271	1.0183 ± 0.0083
Case 2D	200	-	-	0.3000 ± 0.0268	3.1223 ± 0.0389
Test17	2000	500	1000	0.3105 ± 0.0283	1.0174 ± 0.0041
Test15	2000	500	-	0.2895 ± 0.0186	3.2189 ± 0.0412

From this set was selected the Case 1D and test17 the to the last tests, a simple and a more complex model (e.g. Deep Learning). All the values obtained outperformed the article fingerprint approach since are all above 0.237.

6.3. Other Results

6.3.1. Dynamic Problem - Cross-Validation

In order to demonstrate the problem of using the dynamic was made with functional connectivity dynamic for the In-house data using the 50 nodes parcellation and was used the session 1 to predict the session 2. The model used was a simple one, with 3 fully connected layers:

i. Inputs – 4278
ii. Hidden-layer – 200
iii. Outputs – 76

And the hyperparameters applied are defined in *Table 6.59*. The cross-validation was used with 10 folds and the test was repeated five times.

Table 6.59 – Hyperparameters of one of the models.

Batch size	Kernel initializer	Bias initializer	Activation function	Loss Function	Optimizer	Learning rate	Epochs
20	Glorot normal	Random uniform	Softsign	Categorical Hinge	RMSprop	0.0001	500

Table 6.60 – Results for the dynamic test using the cross-validation

Training average cost	Validation accuracy	Final cost	Test accuracy	Test Cost
$3.30 \times 10^{-08} \pm 0.033$	1.0 ± 0.0	$5.60 \times 10^{-07} \pm 5.49 \times 10^{-07}$	0.0467	1.4575

As can be seen in the *Table 6.60*, the final validation accuracy for each subset was always 1.0 and the final cost is also near to the training cost demonstrating that the validation data values are much related with the training data. In other hand the final test accuracy is very far in the accuracy and cost to the validation values, so it's not possible use the cross-validation with the dynamic functional connectivity.

6.3.2. Static Problem - Cross-Validation

The static data has the problem of have few cases for each label (e.g. subject), for example for In-House data there is one case for each subject and in the HCP are two for each subject. This makes the application of cross-validation inapplicable, since the final value of the cross-validation will be a correct measurement of the performance of the model. Since in the training could not occur any training with the label that appear in the validation. So, the validation values will be always low or even null, no giving the true value of the model performance. So, to prove this problem was made the cross-validation with 5 folds, to In-House data 50 nodes parcellation, predicting the session 2 using the session 1.

The model used was a simple one, with 3 fully connected layers:

i. Inputs – 4278
ii. Hidden-layer – 200
iii. Outputs – 76

And the hyperparameters applied are defined in *Table 6.59*.

As the results show in the *Table 6.61*, the validation accuracy is almost equal the probability of the random probability of a correct classification, and is very far from the test accuracy. So, the value of the validation doesn't characterize the real performance of the model which makes to adjust the hyperparameters analysing these values.

Table 6.61 – Results for the static test using the cross-validation

Training average cost	Validation accuracy	Final cost	Test accuracy	Test Cost
1.23x10^{-07} ± 0.1829	0.0258 ± 0.0317	1.4136 ± 0.0465	0.1447	1.2194

7. CONCLUSION

1. **Can an implementation of a *nipype* pipeline improve several points of a processing workflow?**

1.1 And with differences in the time and data organization?

Regarding the methods applied and obtained results in this work, there are some assumptions and pertinent statements that can be made. The construction of a nipype pipeline brings positive features and several advantages in relation to the normal use, either by commands lines or bash scripts. The fact that the pipeline created, before and during the computation optimized the different preprocessing steps made according to the order that they appear, the type of preprocessing done and the number of acquisitions, decrease the necessary time several times.

1.2 What about the data management?

The different results, produced by preprocessing, were automatically created in a base directory by a structure easily handled. This base directory contains the different folders of the parcellations and the subjects with the respective files. During the definition of the pipeline is not difficult to identify the different parts, since it is only necessary the directory base that is always saved. As happen in the creation of the pipeline, each preprocess step has a name choose by the customizer, and the pipeline will be organized by them.

1.3 And using multi-processing?

Other great feature is the fact that the pipeline compute in parallel and do not need any software. This gives a great advantage to the normal use, when the computation is required to be divided and it has to be done by handmade and there is no processes optimization.

1.4 How to stop failures from affecting the working pipeline?

Also in the nipype pipeline, the pre-processing steps do not have to be made since from the beginning until the end of the row, instead of the normal use. Once the processing is done step by step saving the results during the process, it can run preprocess and then go to another subject. Besides, this gives the possibility of optimizing the process, as already mentioned, making the pipeline fault tolerant. If the preprocessing ends for some reason, when the pipeline rerun, it will be checked the different processes done and begins where the process was stopped. Thus, this is very important, since there is no need of repeated computation and, at the end, it saves a lot of time. Even over in this type of preprocessing wherein are used acquisitions of rs-fMRI with several minutes, making a computation of the pipeline for a single subject last several minutes.

2. **What approach should be used to extract the different types of functional connectivity?**

The functional connectivity analysed and extracted, using the ROI's approach, where the functional connectivity is study between each pair of brain regions defined à priori. To define these regions is used an atlas where define each node by a set of voxels that share a similar characteristic, functional or anatomical. Besides using this approach, this work also extracted the functional connectivity in two different ways originating thus two types of correlation matrices. One was the static where is computed by the Pearson's correlation between the full time-series of each pair of brain regions [10]; [183]. And the other was the dynamic where was computed the phase coherence to estimate the phase of the time-series of each region for each volume in the acquisition (TR) [63]; [64]. To compute the different phases of the BOLD signal was used the Hilbert transform.

3. **Which is the best way and which data to save using a python environment?**

3.1 It is the Python object a good choice?

Since Python is an object-oriented programming language, there is the option to deal with the datasets as objects of a well-defined class that is easily saved and load from a normal file. When using the data as objects there is the opportunity to save different attributes as meta-information attributes as well as different arrangements of data. Attributes that are easy to get from an object. Moreover the fact that it is possible to create a set of methods, such as pre-processing before retrieving a data or as tool for apply in a type of data, they are always available in the object to be used. Thus, to get the data, or any data with some preprocessing or information, is quicker and of easy application; it is only needed two lines of code.

3.2 And which data will be saved?

The data saved in this object datasets were of different types depending majority of the package used in the development of Deep Learning models and the type of models (e.g. fully connected layers and convolutional networks). For fully connected layers it were only save non-repeated values, and for convolutional networks it were saved all the correlation matrices. Besides, once the convolutional networks use the spatial information from images, it was added the option of add structural information about the nodes of a parcellation and create the respective correlation matrices. Further, the object also provides the normalized data. The labels used in the classification supervised learning were also saved in the normal and binary levels. Therefore, the object provides a set of different arrangements and a set of values to test different approaches and help the learning process in the Deep Learning models training. The most used data was saved in the object, in order to be faster the data retrieving and do not be necessary repeated computation, with exception of the normalizations. Once they are not computational heavy, due the NumpY optimizations, they increase to much the dataset size, especially the dynamic functional connectivity with one correlation matrix by volume.

4. **What is the best architecture to implement in order of fine-tunnelling the DL models and its posterior creation?**

   4.1 Which Deep Learning python libraries are more appropriate to use?

In the development of the models, the Python open-source Keras library demonstrated to be the best package to develop the different DL models. It made the development process faster and easier, once the models are easily created and changed, besides providing several methods to manage and to deal with the different models created. They also offer a panoply of different types of layers, and hyperparameters that can be applied to the models. As well it wraps other Deep Learning libraries, such as Theano and Tensorflow, which allows, with the correct CUDA drivers installed, to run the computations associated to the training, validation, and testing of the model, in the GPU instead of using the normal CPU. With this technique, the time-consumption is decreased multiple times, being possible to make more Deep Learning tests in less time and train models with more parameters due to the depth or the number of nodes by layer.

   4.2 Which metrics to use and to compare the models?

In order to compare the different models it was used a set of metrics to assess the performance of the model and to be possible to compare with other models, - to fine-tuning the different hyperparameters of the model. For that, it was used some metrics, such as the accuracy and loss, to assess the behaviour of the model in the training and validation. In addition to the final values, other important values are calculated during the training, such as maximum accuracy in validation with the corresponding epoch occurrence, the differences between the different data (e.g. training and validation), and others. The different metrics used were always saved, in order to be used in the future and the information not to be lost.

4.3 Which metrics are extracted in the final analysis of the models?

In the final analysis of the models, as done before, the loss and accuracy were extracted, and a set of other metrics to compare the differences between the different parts, in training, validation and test data were performed. These metrics are used to compare the different results and to confirm if the model generalizes well. Furthermore, the final classification results were also analysed, which were computed for each label, and globally analysed to different parameters, such as the precision, sensitivity, specificity, false positive rate, f1-score, and Area under the curve AUC. Also all the metrics are saved for future use.

4.4 Was there any limitation in the validation approach used?

The validation approaches used in the models in the case of the fingerprint's classification have some limitations. The cross-validation is the most correct form of validation, however it was not possible to apply, since for static there were few cases for each label. In other hand, in the case of the dynamic, due to the predictable behaviour in successive dynamic functional connectivity matrices. So the best approach was to divide each acquisition session in two equal parts. Then, in the Deep Learning application in the fingerprint's classification, the functional connectivity values of a full acquisition session are used as training data. For other session, FC values are used from two divided parts from full acquisition: one is used to do the validation data and the other is used to do test data.

Thus, joining all the different points, it was possible to create an architecture framework to use rs-fMRI in raw format (e.g., DICOM images) and Deep Learning models that use functional connectivity in classifications tasks. Which comprehends the pre-processing of the raw data, creation of datasets with different functional connectivity information, and a module to help to fine-tuning the different Deep Learning models before getting a "good" model, which is used to the final analysis.

Using Deep Learning application was already done in some fine-tuning processes for fully connected layers, which results in a set of different model architectures and hypermeters. The preliminary analysis reveals some promising results.

In the future, it will be necessary to improve the framework capacity to create models for regression and the set of tools to analyse their results. Due timing limitation it was not possible to explore further the fingerprint's classification. The questions rise in this manuscript should be address in future works.

REFERENCES

[1] Haux, R. (2010a). Medical informatics: Past, present, future. *International Journal of Medical Informatics*, *79*(9), 599–610. https://doi.org/10.1016/j.ijmedinf.2010.06.003

[2] Müller, H., Gao, X., & Luo, S. (2008b). From medical imaging to medical informatics. *Computer Methods and Programs in Biomedicine*, *92*(3), 225–226. https://doi.org/10.1016/j.cmpb.2008.07.004

[3] WHO | Imaging Modalities. (2015c). *WHO*.

[4] Lane, R. D., & Wager, T. D. (2009d). Introduction to a Special Issue of Neuroimage on Brain-Body Medicine. *NeuroImage*, *47*(3), 781–784. https://doi.org/10.1016/j.neuroimage.2009.06.004

[5] Horwitz, B., Smith, J. F., Jacobs, J., Kahana, M. J., Aimone, J. B., Wiles, J., ... Lim, W. a. (2009e). What's new in neuroimaging methods? *Annals of the New York Academy of Sciences*, *14*(4), 260–293. https://doi.org/10.1111/j.1749-6632.2009.04420.x.What

[6] Functional Magnetic Resonance Imaging (fMRI). (2009f). In *Encyclopedia of Neuroscience* (pp. 1652–1652). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-29678-2_1873

[7] Ogawa, S., Lee, T. M., Kay, A. R., & Tank, D. W. (1990g). Brain magnetic resonance imaging with contrast dependent on blood oxygenation. *Proceedings of the National Academy of Sciences of the United States of America*, *87*(24), 9868–72. https://doi.org/10.1073/pnas.87.24.9868

[8] Smith, S. M. (2012h). NeuroImage The future of FMRI connectivity. *NeuroImage*, *62*(2), 1257–1266. https://doi.org/10.1016/j.neuroimage.2012.01.022

[9] Friston, K. J. (1994i). Functional and effective connectivity in neuroimaging: A synthesis. *Human Brain Mapping*, *2*(1–2), 56–78. https://doi.org/10.1002/hbm.460020107

[10] Biswal, B., Yetkin, F. Z., Haughton, V. M., & Hyde, J. S. (1995j). Functional connectivity in the motor cortex of resting human brain using echo-planar MRI. *Magnetic Resonance in Medicine*, *34*(4), 537–41. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/8524021

[11] Gusnard, D. A., Raichle, M. E., & Raichle, M. E. (2001k). Searching for a baseline: functional imaging and the resting human brain. *Nature Reviews. Neuroscience*, *2*(10), 685–94. https://doi.org/10.1038/35094500

[12] Dosenbach, N. U. F., Fair, D. A., Miezin, F. M., Cohen, A. L., Wenger, K. K., Dosenbach, R. A. T., ... Petersen, S. E. (2007l). Distinct brain networks for adaptive and stable task control in humans. *Proceedings of the National Academy of Sciences*, *104*(26), 11073–11078. https://doi.org/10.1073/pnas.0704320104

[13] Beckmann, C. F., DeLuca, M., Devlin, J. T., & Smith, S. M. (2005m). Investigations into resting-state connectivity using independent component analysis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *360*(1457), 1001–1013. https://doi.org/10.1098/rstb.2005.1634

[14] Dennis, E. L., & Thompson, P. M. (2014n). Functional brain connectivity using fMRI in aging and Alzheimer's disease. *Neuropsychology Review*, *24*(1), 49–62. https://doi.org/10.1007/s11065-014-9249-6

[15] Maximo, J. O., Cadena, E. J., & Kana, R. K. (2014o). The implications of brain connectivity in the neuropsychology of autism. *Neuropsychology Review*, *24*(1), 16–31. https://doi.org/10.1007/s11065-014-9250-0

[16] Jung, W. H., Prehn, K., Fang, Z., Korczykowski, M., Kable, J. W., Rao, H., & Robertson, D. C. (2016p). Moral competence

and brain connectivity: A resting-state fMRI study. *NeuroImage*, *141*, 408–415. https://doi.org/10.1016/j.neuroimage.2016.07.045

[17] Ball, G., Aljabar, P., Arichi, T., Tusor, N., Cox, D., Merchant, N., ... Counsell, S. J. (2016q). Machine-learning to characterise neonatal functional connectivity in the preterm brain. *NeuroImage*, *124*, 267–275. https://doi.org/10.1016/j.neuroimage.2015.08.055

[18] Byun, H. Y., Lu, J. J., Mayberg, H. S., & Günay, C. (2014r). Classification of Resting State fMRI Datasets Using Dynamic Network Clusters. *Modern Artificial Intelligence for Health Analytics*, *14*, 2–6.

[19] Finn, E. S., Shen, X., Scheinost, D., Rosenberg, M. D., Huang, J., Chun, M. M., ... Constable, R. T. (2015s). Functional connectome fingerprinting: Identifying individuals using patterns of brain connectivity. *Nature Neuroscience*, *18*(11), 1664–1671. https://doi.org/10.1038/nn.4135

[20] Suzuki, K., Yan, P., Wang, F., & Shen, D. (2012t). Machine learning in medical imaging. *International Journal of Biomedical Imaging*, *2012*, 2012–2014. https://doi.org/10.1155/2012/123727

[21] LeCun, Y., Bengio, Y., & Hinton, G. (2015u). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

[22] Wang, P., Ge, R., Xiao, X., Cai, Y., Wang, G., & Zhou, F. (2016v). Rectified-Linear-Unit-Based Deep Learning for Biomedical Multi-label Data. *Interdisciplinary Sciences: Computational Life Sciences*. https://doi.org/10.1007/s12539-016-0196-1

[23] Cnns, D. (2016w). Guest Editorial Deep Learning in Medical Imaging : Overview and Future Promise of an Exciting New Technique. *IEEE Transactions on Medical Imaging*, *35*(5), 1153–1159. https://doi.org/10.1109/TMI.2016.2553401

[24] Leedy, P. D., & Ormrod, J. E. (2010x). *Practical Research: Planning and Design*. *Practical Research - Planning & Design*. Retrieved from http://www.studentsofferingsupport.ca/portal/OutreachProjects/PreDepReadings/E2_WRITING_Practical Research Planning and Design.pdf

[25] Kothari, C. (2004y). *Research Mathodology : Methods and Techniques*. Retrieved from http://www.mu.edu.et/iphc/images/liblary/Heritage/Heritage_Culture_and_Tourism/Research_Method/Research_Methodology_Kothari.pdf

[26] Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004z). DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH 1. *Design Science in IS Research MIS Quarterly*, *28*(1), 75–105. Retrieved from https://pdfs.semanticscholar.org/fa72/91f2073cb6fdbdd7c2213bf6d776d0ab411c.pdf

[27] Hevner, A., & Chatterjee, S. (2010aa). Design Science Research in Information Systems (pp. 9–22). https://doi.org/10.1007/978-1-4419-5653-8_2

[28] Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006ab). THE DESIGN SCIENCE RESEARCH PROCESS: A MODEL FOR PRODUCING AND PRESENTING INFORMATION SYSTEMS RESEARCH. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.469.2936&rep=rep1&type=pdf

[29] Bunge, S. A., & Kahn, I. (2010ac). Cognition: An Overview of Neuroimaging Techniques. In *Encyclopedia of Neuroscience* (pp. 1063–1067). https://doi.org/10.1016/B978-008045046-9.00298-9

[30] Bassett, D. S., & Sporns, O. (2017ad). Network neuroscience. *Nature Neuroscience*, *20*(3), 353–364. https://doi.org/10.1038/nn.4502

[31] Hodgson, R. J. (2011ae). (v) The basic science of MRI. *Orthopaedics and Trauma*, *25*(2), 119–130.

https://doi.org/10.1016/j.mporth.2010.12.002

[32] Duyn, J. H. (2012af). The future of ultra-high field MRI and fMRI for study of the human brain. *NeuroImage*, *62*(2), 1241–1248. https://doi.org/10.1016/j.neuroimage.2011.10.065

[33] OCDE. (2016ag). *OECD Data-Magnetic resonance imaging (MRI) exams*. OECD Publishing. Retrieved from http://www.oecd-ilibrary.org/social-issues-migration-health/health-at-a-glance-asia-pacific-2016_health_glance_ap-2016-en

[34] Vijayalaxmi, Fatahi, M., & Speck, O. (2015ah). Magnetic resonance imaging (MRI): A review of genetic damage investigations. *Mutation Research/Reviews in Mutation Research*, *764*, 51–63. https://doi.org/10.1016/j.mrrev.2015.02.002

[35] Bandettini, P. A. (2012ai). Functional MRI: A confluence of fortunate circumstances. *NeuroImage*, *61*(2), A3–A11. https://doi.org/10.1016/j.neuroimage.2012.01.130

[36] Buxton, R. B., Uludağ, K., Dubowitz, D. J., & Liu, T. T. (2004aj). Modeling the hemodynamic response to brain activation. *NeuroImage*, *23*, S220–S233. https://doi.org/10.1016/j.neuroimage.2004.07.013

[37] Handwerker, D. A., Gonzalez-Castillo, J., D'Esposito, M., & Bandettini, P. A. (2012ak, August). The continuing challenge of understanding and modeling hemodynamic variation in fMRI. *NeuroImage*. https://doi.org/10.1016/j.neuroimage.2012.02.015

[38] Soares, J., Magalhães, R., Moreira, P., Sousa, A., Ganz, E., Sampaio, A., ... Sousa, N. (2016al). A hitchhiker's guide to functional Magnetic Resonance Imaging. *Frontiers in Neuroscience*, *10*, 515. https://doi.org/10.3389/FNINS.2016.00515

[39] Hemodynamic Response Function | The Clever Machine - fMRI in Neuroscience: Modeling the HRF with FIR Basis Functions. (n.d.-am). Retrieved October 23, 2017, from https://theclevermachine.wordpress.com/tag/hemodynamic-response-function/

[40] Mansfield, P. (1977an). Multi-planar image formation using NMR spin echoes. *Journal of Physics C: Solid State Physics*, *10*(3), L55–L58. https://doi.org/10.1088/0022-3719/10/3/004

[41] Greve, D. N., Brown, G. G., Mueller, B. A., Glover, G., Liu, T. T., & Network, F. B. R. (2013ao). A Survey of the Sources of Noise in fMRI. *Psychometrika*, *78*(3), 396–416. https://doi.org/10.1007/s11336-012-9294-0

[42] Poser, B. A., Norris, D. G., & Hahn, E. L. (2009ap). Investigating the benefits of multi-echo EPI for fMRI at 7 T. *NeuroImage*, *45*, 1162–1172. https://doi.org/10.1016/j.neuroimage.2009.01.007

[43] Gowland, P. A., & Bowtell, R. (2007aq). Theoretical optimization of multi-echo fMRI data acquisition. *Physics in Medicine and Biology*, *52*(7), 1801–1813. https://doi.org/10.1088/0031-9155/52/7/003

[44] Posse, S. (2012ar). Multi-echo acquisition. *NeuroImage*, *62*(2), 665–671. https://doi.org/10.1016/j.neuroimage.2011.10.057

[45] Friston, K. J., Frith, C. D., Liddle, P. F., & Frackowiak, R. S. J. (1993as). Functional Connectivity: The Principal-Component Analysis of Large (PET) Data Sets. *Journal of Cerebral Blood Flow & Metabolism*, *13*(1), 5–14. https://doi.org/10.1038/jcbfm.1993.4

[46] Fox, M. D., & Raichle, M. E. (2007at). Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nature Reviews Neuroscience*, *8*(9), 700–711. https://doi.org/10.1038/nrn2201

[47] Greicius, M. D., Krasnow, B., Reiss, A. L., & Menon, V. (2003au). Functional connectivity in the resting brain: A network

analysis of the default mode hypothesis. *Proceedings of the National Academy of Sciences*, *100*(1), 253–258. https://doi.org/10.1073/pnas.0135058100

[48] Rogers, B. P., Morgan, V. L., Newton, A. T., & Gore, J. C. (2007av). Assessing functional connectivity in the human brain by fMRI. *Magnetic Resonance Imaging*, *25*(10), 1347–1357. https://doi.org/10.1016/j.mri.2007.03.007

[49] Ji, Y., Hervé, P.-Y., Aickelin, U., & Pitiot, A. (2009aw). Parcellation of fMRI datasets with ICA and PLS–a data driven approach. *Medical Image Computing and Computer-Assisted Intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, *12*(Pt 1), 984–91. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/20426084

[50] Shen, X., Tokoglu, F., Papademetris, X., & Constable, R. T. (2013ax). Groupwise whole-brain parcellation from resting-state fMRI data for network node identification. *NeuroImage*, *82*, 403–415. https://doi.org/10.1016/j.neuroimage.2013.05.081

[51] Chen, X., Zhang, H., Gao, Y., Wee, C.-Y., Li, G., Shen, D., & Alzheimer's Disease Neuroimaging Initiative. (2016ay). High-order resting-state functional connectivity network for MCI classification. *Human Brain Mapping*, *37*(9), 3282–3296. https://doi.org/10.1002/hbm.23240

[52] Erhardt, E. B., Rachakonda, S., Bedrick, E. J., Allen, E. A., Adali, T., & Calhoun, V. D. (2011az). Comparison of multi-subject ICA methods for analysis of fMRI data. *Human Brain Mapping*, *32*(12), 2075–2095. https://doi.org/10.1002/hbm.21170

[53] Allen, E. A., Damaraju, E., Plis, S. M., Erhardt, E. B., Eichele, T., & Calhoun, V. D. (2014ba). Tracking Whole-Brain Connectivity Dynamics in the Resting State. *Cerebral Cortex*, *24*(3), 663–676. https://doi.org/10.1093/cercor/bhs352

[54] Leonardi, N., Richiardi, J., Gschwind, M., Simioni, S., Annoni, J.-M., Schluep, M., ... Van De Ville, D. (2013bb). Principal components of functional connectivity: A new approach to study dynamic brain connectivity during rest. *NeuroImage*, *83*, 937–950. https://doi.org/10.1016/j.neuroimage.2013.07.019

[55] Hutchison, R. M., Womelsdorf, T., Allen, E. A., Bandettini, P. A., Calhoun, V. D., Corbetta, M., ... Chang, C. (2013bc). Dynamic functional connectivity: Promise, issues, and interpretations. *NeuroImage*, *80*, 360–378. https://doi.org/10.1016/j.neuroimage.2013.05.079

[56] Wee, C.-Y., Yang, S., Yap, P.-T., Shen, D., & Alzheimer's Disease Neuroimaging Initiative. (2016bd). Sparse temporally dynamic resting-state functional connectivity networks for early MCI identification. *Brain Imaging and Behavior*, *10*(2), 342–356. https://doi.org/10.1007/s11682-015-9408-2

[57] Associations between dynamic functional connectivity and age, metabolic risk, and cognitive performance. (2017be). *Neurobiology of Aging*, *59*, 135–143. https://doi.org/10.1016/J.NEUROBIOLAGING.2017.08.003

[58] Dynamic functional connectivity analysis reveals transient states of dysconnectivity in schizophrenia. (2014bf). *NeuroImage: Clinical*, *5*, 298–308. https://doi.org/10.1016/J.NICL.2014.07.003

[59] Chang, C., & Glover, G. H. (2010bg). Time-frequency dynamics of resting-state brain connectivity measured with fMRI. *NeuroImage*, *50*(1), 81–98. https://doi.org/10.1016/j.neuroimage.2009.12.011

[60] Liu, X., Chang, C., & Duyn, J. H. (2013bh). Decomposition of spontaneous brain activity into distinct fMRI co-activation patterns. *Frontiers in Systems Neuroscience*, *7*, 101. https://doi.org/10.3389/fnsys.2013.00101

[61] Pan, W.-J., Thompson, G. J., Magnuson, M. E., Jaeger, D., & Keilholz, S. (2013bi). Infraslow LFP correlates to resting-state fMRI BOLD signals. *NeuroImage*, *74*, 288–297. https://doi.org/10.1016/j.neuroimage.2013.02.035

[62] Shine, J. M., Koyejo, O., Bell, P. T., Gorgolewski, K. J., Gilat, M., & Poldrack, R. A. (2015bj). Estimation of dynamic functional connectivity using Multiplication of Temporal Derivatives. *NeuroImage*, *122*, 399–407. https://doi.org/10.1016/j.neuroimage.2015.07.064

[63] Glerean, E., Salmi, J., Lahnakoski, J. M., Jääskeläinen, I. P., & Sams, M. (2012bk). Functional Magnetic Resonance Imaging Phase Synchronization as a Measure of Dynamic Functional Connectivity. *Brain Connectivity*, *2*(2), 91–101. https://doi.org/10.1089/brain.2011.0068

[64] Deco, G., & Kringelbach, M. L. (2016bl). Metastability and Coherence: Extending the Communication through Coherence Hypothesis Using A Whole-Brain Computational Perspective. *Trends in Neurosciences*, *39*(3), 125–135. https://doi.org/10.1016/j.tins.2016.01.001

[65] Ponce-Alvarez, A., Deco, G., Hagmann, P., Romani, G. L., Mantini, D., & Corbetta, M. (2015bm). Resting-State Temporal Synchronization Networks Emerge from Connectivity Topology and Heterogeneity. *PLOS Computational Biology*, *11*(2), e1004100. https://doi.org/10.1371/journal.pcbi.1004100

[66] Vergara, V. M., Miller, R., & Calhoun, V. (2017bn). An information theory framework for dynamic functional domain connectivity. *Journal of Neuroscience Methods*, *284*, 103–111. https://doi.org/10.1016/j.jneumeth.2017.04.009

[67] Petersen, S. E., & Dubis, J. W. (2012bo). The mixed block/event-related design. *NeuroImage*, *62*(2), 1177–84. https://doi.org/10.1016/j.neuroimage.2011.09.084

[68] Power, J. D., Barnes, K. A., Snyder, A. Z., Schlaggar, B. L., & Petersen, S. E. (2012bp). Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion. *NeuroImage*, *59*(3), 2142–2154. https://doi.org/10.1016/j.neuroimage.2011.10.018

[69] Murphy, K., Birn, R. M., & Bandettini, P. A. (2013bq). Resting-state fMRI confounds and cleanup. *NeuroImage*, *80*, 349–359. https://doi.org/10.1016/j.neuroimage.2013.04.001

[70] Bianciardi, M., Fukunaga, M., van Gelderen, P., Horovitz, S. G., de Zwart, J. A., Shmueli, K., & Duyn, J. H. (2009br). Sources of functional magnetic resonance imaging signal fluctuations in the human brain at rest: a 7 T study. *Magnetic Resonance Imaging*, *27*(8), 1019–1029. https://doi.org/10.1016/j.mri.2009.02.004

[71] DICOM introduction and free software. (n.d.-bs). Retrieved from http://people.cas.sc.edu/rorden/dicom/index.html

[72] NIfTI-1 Data Format — Neuroimaging Informatics Technology Initiative. (n.d.-bt). Retrieved from https://nifti.nimh.nih.gov/nifti-1

[73] Nieto-Castanon, A., Ghosh, S. S., Tourville, J. A., & Guenther, F. H. (2003bu). Region of interest based analysis of functional imaging data. *NeuroImage*, *19*(4), 1303–16. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/12948689

[74] Saxe, R., Brett, M., & Kanwisher, N. (2006bv). Divide and conquer: A defense of functional localizers. *NeuroImage*, *30*(4), 1088–1096. https://doi.org/10.1016/j.neuroimage.2005.12.062

[75] Tzourio-Mazoyer, N., Landeau, B., Papathanassiou, D., Crivello, F., Etard, O., Delcroix, N., … Joliot, M. (2002bw). Automated Anatomical Labeling of Activations in SPM Using a Macroscopic Anatomical Parcellation of the MNI MRI Single-Subject Brain. *NeuroImage*, *15*(1), 273–289. https://doi.org/10.1006/nimg.2001.0978

[76] Amunts, K., Kedo, O., Kindler, M., Pieperhoff, P., Mohlberg, H., Shah, N. J., … Zilles, K. (2005bx). Cytoarchitectonic mapping of the human amygdala, hippocampal region and entorhinal cortex: intersubject variability and probability maps. *Anatomy and Embryology*, *210*(5–6), 343–352. https://doi.org/10.1007/s00429-005-0025-5

[77] Thomas Yeo, B. T., Krienen, F. M., Sepulcre, J., Sabuncu, M. R., Lashkari, D., Hollinshead, M., ... Buckner, R. L. (2011by). The organization of the human cerebral cortex estimated by intrinsic functional connectivity. *Journal of Neurophysiology*, *106*(3), 1125–1165. https://doi.org/10.1152/jn.00338.2011

[78] Kahnt, T., Chang, L. J., Park, S. Q., Heinzle, J., & Haynes, J.-D. (2012bz). Connectivity-Based Parcellation of the Human Orbitofrontal Cortex. *Journal of Neuroscience*, *32*(18), 6240–6250. https://doi.org/10.1523/JNEUROSCI.0257-12.2012

[79] Wang, Q., Chen, R., JaJa, J., Jin, Y., Hong, L. E., & Herskovits, E. H. (2016ca). Connectivity-Based Brain Parcellation: A Connectivity-Based Atlas for Schizophrenia Research. *Neuroinformatics*, *14*(1), 83–97. https://doi.org/10.1007/s12021-015-9280-7

[80] Craddock, R. C., James, G. A., Holtzheimer, P. E., Hu, X. P., Mayberg, H. S., & Mayberg, H. S. (2012cb). A whole brain fMRI atlas generated via spatially constrained spectral clustering. *Human Brain Mapping*, *33*(8), 1914–28. https://doi.org/10.1002/hbm.21333

[81] Abraham, A., Dohmatob, E., Thirion, B., Samaras, D., & Varoquaux, G. (2013cc). Extracting brain regions from rest fMRI with Total-Variation constrained dictionary learning. Retrieved from https://hal.inria.fr/hal-00853242

[82] Varoquaux, G., Sadaghiani, S., Pinel, P., Kleinschmidt, A., Poline, J. B., & Thirion, B. (2010cd). A group model for stable multi-subject ICA on fMRI datasets. *NeuroImage*, *51*(1), 288–299. https://doi.org/10.1016/j.neuroimage.2010.02.010

[83] Smith, S. M., Hyvärinen, A., Varoquaux, G., Miller, K. L., & Beckmann, C. F. (2014ce). Group-PCA for very large fMRI datasets. *NeuroImage*, *101*, 738–749. https://doi.org/10.1016/j.neuroimage.2014.07.051

[84] Ashburner, J. (2012cf). SPM: A history. *NeuroImage*, *62*(2), 791–800. https://doi.org/10.1016/j.neuroimage.2011.10.025

[85] Friston, K. J., Holmes, a. P., Worsley, K. J., Poline, J.-P., Frith, C. D., & Frackowiak, R. S. J. (1995cg). Statistical parametric maps in functional imaging: A general linear approach. *Human Brain Mapping*, *2*(4), 189–210. https://doi.org/10.1002/hbm.460020402

[86] SPM - Statistical Parametric Mapping. (n.d.-ch). Retrieved from http://www.fil.ion.ucl.ac.uk/spm/

[87] Dale, A. M. (1994ci). *Source Localization and Spatial Discriminant Analysis of Event-related Potentials: Linear Approaches*. book, University of California, San Diego, Department of Cognitive Science.

[88] Dale, A. M., Fischl, B., & Sereno, M. I. (1999cj). Cortical surface-based analysis. I. Segmentation and surface reconstruction. *NeuroImage*, *9*(2), 179–94. https://doi.org/10.1006/nimg.1998.0395

[89] Fischl, B. (2012ck). FreeSurfer. *NeuroImage*, *62*(2), 774–781. https://doi.org/10.1016/j.neuroimage.2012.01.021

[90] Jenkinson, M., Beckmann, C. F., Behrens, T. E. J., Woolrich, M. W., & Smith, S. M. (2012cl). Fsl. *NeuroImage*, *62*(2), 782–790. https://doi.org/10.1016/j.neuroimage.2011.09.015

[91] Landman, B. A., Huang, A. J., Gifford, A., Vikram, D. S., Lim, I. A. L., Farrell, J. A. D., ... van Zijl, P. C. M. (2011cm). Multi-parametric neuroimaging reproducibility: A 3-T resource study. *NeuroImage*, *54*(4), 2854–2866. https://doi.org/10.1016/j.neuroimage.2010.11.047

[92] Dai, T., & Guo, Y. (2017cn). Predicting individual brain functional connectivity using a Bayesian hierarchical model. *NeuroImage*, *147*, 772–787. https://doi.org/10.1016/j.neuroimage.2016.11.048

[93] Fox, M. D., & Greicius, M. (2010co). Clinical applications of resting state functional connectivity. *Frontiers in Systems Neuroscience*, *4*, 19. https://doi.org/10.3389/fnsys.2010.00019

[94] Sutherland, M. T., McHugh, M. J., Pariyadath, V., & Stein, E. A. (2012cp). Resting state functional connectivity in addiction: Lessons learned and a road ahead. *NeuroImage*, *62*(4), 2281–2295. https://doi.org/10.1016/j.neuroimage.2012.01.117

[95] He, H., Yu, Q., Du, Y., Vergara, V., Victor, T. A., Drevets, W. C., … Calhoun, V. D. (2016cq). Resting-state functional network connectivity in prefrontal regions differs between unmedicated patients with bipolar and major depressive disorders. *Journal of Affective Disorders*, *190*, 483–493. https://doi.org/10.1016/j.jad.2015.10.042

[96] Pearson, K. (1901cr). LIII. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

[97] FISHER, R. A. (1936cs). THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. *Annals of Eugenics*, *7*(2), 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

[98] Goodfellow, I., Bengio, Y., & Courville, A. (2016ct). *Deep Learning*. book, MIT Press.

[99] Zhong, G., Wang, L.-N., Ling, X., & Dong, J. (2016cu). An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, *2*(4), 265–278. https://doi.org/10.1016/j.jfds.2017.05.001

[100] Urban, G., Geras, K. J., Kahou, S. E., Aslan, O., Wang, S., Caruana, R., … Richardson, M. (2016cv). Do Deep Convolutional Nets Really Need to be Deep and Convolutional? *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

[101] Bengio, Y., Courville, A., & Vincent, P. (2013cw). Representation Learning: A Review and New Perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

[102] Sun, S., Chen, W., Wang, L., Liu, X., & Liu, T.-Y. (2015cx). On the Depth of Deep Neural Networks: A Theoretical View. Retrieved from http://arxiv.org/abs/1506.05232

[103] Fenn, J., & Lehong, H. (2011cy). Hype Cycle for Emerging Technologies, 2011. Retrieved from http://www.gartner.com/technology/about/ombudsman/omb_guide2.jsp

[104] Attneave, F., B., M., & Hebb, D. O. (1950cz). The Organization of Behavior; A Neuropsychological Theory. *The American Journal of Psychology*, *63*(4), 633. https://doi.org/10.2307/1418888

[105] Rosenblatt, F. (n.d.-da). THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN. *Psychological Review*, *65*(6), 19–8. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=pdf

[106] Widrow, B., & Hoff, M. (1960db). Adaptive switching circuits. *1960 IRE WESCON Convention Record*. https://doi.org/no DOI, URL correct

[107] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986dc). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

[108] Cireşan, D., Meier, U., & Schmidhuber, J. (2012dd). Multi-column Deep Neural Networks for Image Classification. Retrieved from http://arxiv.org/abs/1202.2745

[109] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012de). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

[110] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., ... Kingsbury, B. (2012df). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. https://doi.org/10.1109/MSP.2012.2205597

[111] Collobert, R., & Weston, J. (2008dg). A unified architecture for natural language processing. In *Proceedings of the 25th international conference on Machine learning - ICML '08* (pp. 160–167). New York, New York, USA: ACM Press. https://doi.org/10.1145/1390156.1390177

[112] Shen, D., Wu, G., & Suk, H.-I. (2017dh). Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering*, *19*, 221–248. https://doi.org/10.1146/annurev-bioeng-071516-044442

[113] Delalleau, O., & Bengio, Y. (n.d.-di). Shallow vs. Deep Sum-Product Networks. Retrieved from https://papers.nips.cc/paper/4350-shallow-vs-deep-sum-product-networks.pdf

[114] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015dj). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, *2*(1), 1. https://doi.org/10.1186/s40537-014-0007-7

[115] Nair, V., & Hinton, G. E. (2010dk). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning*, (3), 807–814. https://doi.org/10.1.1.165.6419

[116] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014dl). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. https://doi.org/10.1214/12-AOS1000

[117] Ioffe, S., & Szegedy, C. (2015dm). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Retrieved from http://arxiv.org/abs/1502.03167

[118] SEER Training:Nerve Tissue. (n.d.-dn). Retrieved November 9, 2017, from https://training.seer.cancer.gov/anatomy/nervous/tissue.html

[119] Stanford University CS231n: Convolutional Neural Networks for Visual Recognition. (n.d.-do). Retrieved November 9, 2017, from http://cs231n.stanford.edu/

[120] Minsky, M., & Papert, S. (1969dp). Perceptrons. article.

[121] Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (n.d.-dq). Greedy Layer-Wise Training of Deep Networks. Retrieved from http://www.iro.umontreal.ca/~lisa/pointeurs/BengioNips2006All.pdf

[122] Bourlard, H., & Kamp, Y. (1988dr). Auto-Association by Multilayer Perceptrons and Singular Value Decomposition. *Biol. Cybern*, *59*, 291–294. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.453.846&rep=rep1&type=pdf

[123] Rumelhart, D. E., McClelland, J. L., & University of California, S. D. P. R. G. (1986ds). *Parallel distributed processing : explorations in the microstructure of cognition*. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*. MIT Press. Retrieved from https://dl.acm.org/citation.cfm?id=104279.104290

[124] Kuremoto, T., Kimura, S., Kobayashi, K., & Obayashi, M. (2014dt). Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing*, *137*, 47–56. https://doi.org/10.1016/j.neucom.2013.03.047

[125] Salakhutdinov, R. (2015du). Learning Deep Generative Models. *Annual Review of Statistics and Its Application*, *2*, 361–385. https://doi.org/10.1146/annurev-statistics-010814-020120

[126] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998dv). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2323. https://doi.org/10.1109/5.726791

[127] Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009dw). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09* (pp. 1–8). New York, New York, USA: ACM Press. https://doi.org/10.1145/1553374.1553453

[128] Glorot, X., & Bengio, Y. (n.d.-dx). Understanding the difficulty of training deep feedforward neural networks. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.207.2059&rep=rep1&type=pdf

[129] Kumar, S. K. (2017dy). On weight initialization in deep neural networks. Retrieved from https://arxiv.org/pdf/1704.08863.pdf

[130] He, K., Zhang, X., Ren, S., & Sun, J. (2015dz). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (Vol. 2015 Inter, pp. 1026–1034). https://doi.org/10.1109/ICCV.2015.123

[131] Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013ea). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. Retrieved from http://arxiv.org/abs/1312.6120

[132] Kline, D. M., & Berardi, V. L. (2005eb). Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing and Applications*, *14*(4), 310–318. https://doi.org/10.1007/s00521-005-0467-y

[133] Mathieu, M., Couprie, C., & LeCun, Y. (2015ec). Deep multi-scale video prediction beyond mean square error. Retrieved from http://arxiv.org/abs/1511.05440

[134] Sutton, R. S. (1986ed). Two Problems with Backpropagation and Other Steepest-Descent Learning Procedures for Networks. In *{P}roceedings of the Eighth Annual Conference of the Cognitive Science Society*. inproceedings, Hillsdale, NJ: Erlbaum.

[135] Qian, N. (1999ee, January 1). On the momentum term in gradient descent learning algorithms. *Neural Networks*. Pergamon. https://doi.org/10.1016/S0893-6080(98)00116-6

[136] Bengio, Y., Boulanger-Lewandowski, N., & Pascanu, R. (2012ef). Advances in Optimizing Recurrent Networks. Retrieved from http://arxiv.org/abs/1212.0901

[137] Johnson, R., & Zhang, T. (2013eg). Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. Retrieved from http://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction

[138] Zinkevich, M., Weimer, M., Li, L., & Smola, A. J. (2010eh). Parallelized Stochastic Gradient Descent. Retrieved from http://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent

[139] Klein, S., Pluim, J. P. W., Staring, M., & Viergever, M. A. (2009ei). Adaptive Stochastic Gradient Descent Optimisation for Image Registration. *International Journal of Computer Vision*, *81*(3), 227–239. https://doi.org/10.1007/s11263-008-0168-y

[140] Duchi, J., Edu, J. B., Hazan, E., & Singer, Y. (2011ej). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization *. *Journal of Machine Learning Research*, *12*, 2121–2159. Retrieved from http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

[141] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., … Ng, A. Y. (2012ek). Large Scale Distributed Deep Networks. Retrieved from https://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks

[142] Zeiler, M. D. (n.d.-el). ADADELTA: AN ADAPTIVE LEARNING RATE METHOD. Retrieved from https://arxiv.org/pdf/1212.5701.pdf

[143] Tieleman, T., & Hinton, G. (2012em). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, *4*(2), 26–31. article.

[144] Kingma, D. P., & Ba, J. L. (n.d.-en). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. Retrieved from https://arxiv.org/pdf/1412.6980.pdf

[145] Dozat, T. (2016eo). Incorporating Nesterov Momentum into Adam. *ICLR Workshop*, (1), 2013–2016. Retrieved from http://cs229.stanford.edu/proj2015/054_report.pdf

[146] Vapnik, V. N. (1998ep). Statistical Learning Theory. *Adaptive and Learning Systems for Signal Processing, Communications and Control*, *2*, 1–740. https://doi.org/10.2307/1271368

[147] Mukherjee, S., Niyogi, P., & Poggio, T. (2002eq). Statistical Learning : Well-Posedness is Necessary and Sufficient for Consistency of Empirical Risk Minimization, (December), 0–24. Retrieved from http://cbcl.mit.edu/publications/ps/mukherjee-AImemoOctNov.pdf

[148] Poggio, T., Rifkin, R., Mukherjee, S., & Niyogi, P. (2004er). General conditions for predictivity in learning theory. *Nature*, *428*(6981), 419–422. https://doi.org/10.1038/nature02341

[149] Bartlett, P. P. L., & Mendelson, S. (2002es). Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, *3*(3), 463–482. https://doi.org/10.1162/153244303321897690

[150] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2016et). Understanding deep learning requires rethinking generalization. Retrieved from http://arxiv.org/abs/1611.03530

[151] CIFAR-10 and CIFAR-100 datasets. (n.d.-eu). Retrieved November 15, 2017, from https://www.cs.toronto.edu/~kriz/cifar.html

[152] Kawaguchi, K., Kaelbling, L. P., & Bengio, Y. (2017ev). Generalization in Deep Learning. Retrieved from http://arxiv.org/abs/1710.05468

[153] Ioffe, S., & Szegedy, C. (2015ew). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. https://doi.org/10.1007/s13398-014-0173-7.2

[154] Wang, J., & Perez, L. (n.d.-ex). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Retrieved from http://cs231n.stanford.edu/reports/2017/pdfs/300.pdf

[155] Wong, S. C., Gatt, A., Stamatescu, V., & Mcdonnell, M. D. (2016ey). Understanding data augmentation for classification: when to warp? Retrieved from https://arxiv.org/pdf/1609.08764.pdf

[156] Desikan, R. S., Ségonne, F., Fischl, B., Quinn, B. T., Dickerson, B. C., Blacker, D., ... Killiany, R. J. (2006ez). An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest. https://doi.org/10.1016/j.neuroimage.2006.01.021

[157] Fischl, B., Van Der Kouwe, A., Destrieux, C., Halgren, E., Ségonne, F., Salat, D. H., ... Dale, A. M. (2004fa). Automatically Parcellating the Human Cerebral Cortex. *Cerebral Cortex*, *14*(1), 11–22. https://doi.org/10.1093/cercor/bhg087

[158] Magalhães, R., Marques, P., Soares, J., Alves, V., & Sousa, N. (2015fb). The Impact of Normalization and Segmentation on Resting-State Brain Networks. *Brain Connectivity*, *5*(3), 166–176. https://doi.org/10.1089/brain.2014.0292

[159] Smith, S. M., Beckmann, C. F., Andersson, J., Auerbach, E. J., Bijsterbosch, J., Douaud, G., ... WU-Minn HCP Consortium. (2013fc). Resting-state fMRI in the Human Connectome Project. *NeuroImage*, *80*, 144–168. https://doi.org/10.1016/j.neuroimage.2013.05.039

[160] Glasser, M. F., Sotiropoulos, S. N., Wilson, J. A., Coalson, T. S., Fischl, B., Andersson, J. L., ... Jenkinson, M. (2013fd). The minimal preprocessing pipelines for the Human Connectome Project. *NeuroImage*, *80*, 105–124.

https://doi.org/10.1016/j.neuroimage.2013.04.127

[161] Allen, N. E., Sudlow, C., Peakman, T., Collins, R., & UK Biobank, on behalf of U. (2014fe). UK biobank data: come and get it. *Science Translational Medicine*, *6*(224), 224ed4. https://doi.org/10.1126/scitranslmed.3008601

[162] Miller, K. L., Alfaro-Almagro, F., Bangerter, N. K., Thomas, D. L., Yacoub, E., Xu, J., ... Smith, S. M. (2016ff). Multimodal population brain imaging in the UK Biobank prospective epidemiological study. *Nature Neuroscience*, *19*(11), 1523–1536. https://doi.org/10.1038/nn.4393

[163] Okano, H., Miyawaki, A., & Kasai, K. (2015fg). Brain/MINDS: brain-mapping project in Japan. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *370*(1668), 20140310. https://doi.org/10.1098/rstb.2014.0310

[164] Poo, M., Du, J., Ip, N. Y., Xiong, Z.-Q., Xu, B., & Tan, T. (2016fh). China Brain Project: Basic Neuroscience, Brain Diseases, and Brain-Inspired Computing. *Neuron*, *92*(3), 591–596. https://doi.org/10.1016/j.neuron.2016.10.050

[165] Amunts, K., Ebell, C., Muller, J., Telefont, M., Knoll, A., & Lippert, T. (2016fi). The Human Brain Project: Creating a European Research Infrastructure to Decode the Human Brain. *Neuron*, *92*(3), 574–581. https://doi.org/10.1016/j.neuron.2016.10.046

[166] Van Essen, D. C., Ugurbil, K., Auerbach, E., Barch, D., Behrens, T. E. J., Bucholz, R., ... Yacoub, E. (2012fj, October 1). The Human Connectome Project: A data acquisition perspective. *NeuroImage*. Academic Press. https://doi.org/10.1016/j.neuroimage.2012.02.018

[167] Van Essen, D. C., Smith, S. M., Barch, D. M., Behrens, T. E. J., Yacoub, E., & Ugurbil, K. (2013fk). The WU-Minn Human Connectome Project: An overview. *NeuroImage*, *80*, 62–79. https://doi.org/10.1016/j.neuroimage.2013.05.041

[168] Human Connectome Project - HCP 3 T Imaging Protocol overview. (n.d.-fl). Retrieved October 2, 2017, from http://protocols.humanconnectome.org/HCP/3T/imaging-protocols.html

[169] Rossum, G. Van, & Drake, F. L. (2006fm). Python Reference Manual. *October*, *22*, 9117–9129. https://doi.org/10.1242/jeb.00343

[170] GitHub Octoverse 2017 | Highlights from the last twelve months. (n.d.-fn). Retrieved December 6, 2017, from https://octoverse.github.com/

[171] Continuum Analytics. (2016fo). Anaconda Software Distribution. Retrieved December 6, 2017, from https://docs.anaconda.com/anaconda/faq#how-do-i-cite-anaconda-in-an-academic-paper

[172] NVIDIA Developer. (n.d.-fp). Retrieved October 12, 2017, from https://developer.nvidia.com/

[173] Matplotlib: Python plotting — Matplotlib 2.1.0 documentation. (n.d.-fq). Retrieved October 8, 2017, from https://matplotlib.org/

[174] scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation. (n.d.-fr). Retrieved October 15, 2017, from http://scikit-learn.org/stable/

[175] NumPy — NumPy. (n.d.-fs). Retrieved October 10, 2017, from http://www.numpy.org/

[176] Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., & Ghosh, S. S. (2011ft). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Frontiers in Neuroinformatics*, *5*, 13. https://doi.org/10.3389/fninf.2011.00013

[177] Chollet, F., & others. (2015fu). Keras. misc, GitHub.

[178] Hanke, M., & Halchenko, Y. O. (2011fv). Neuroscience Runs on GNU/Linux. *Frontiers in Neuroinformatics*, *5*, 8.

https://doi.org/10.3389/fninf.2011.00008

[179] Marques, P., Soares, J. M., Alves, V., & Sousa, N. (2013fw). BrainCAT - a tool for automated and combined functional magnetic resonance imaging and diffusion tensor imaging brain connectivity analysis. *Frontiers in Human Neuroscience*, *7*, 794. https://doi.org/10.3389/fnhum.2013.00794

[180] Churchill, N. W., Spring, R., Afshin-Pour, B., Dong, F., & Strother, S. C. (2015fx). An Automated, Adaptive Framework for Optimizing Preprocessing Pipelines in Task-Based Functional MRI. *PloS One*, *10*(7), e0131520. https://doi.org/10.1371/journal.pone.0131520

[181] Millman, K. J., & Brett, M. (2007fy). Analysis of functional magnetic resonance imaging in python. *Computing in Science and Engineering*, *9*(3), 52–55. https://doi.org/10.1109/MCSE.2007.46

[182] Li, X., Morgan, P. S., Ashburner, J., Smith, J., & Rorden, C. (2016fz). The first step for neuroimaging data analysis: DICOM to NIfTI conversion. *Journal of Neuroscience Methods*, *264*, 47–56. https://doi.org/10.1016/j.jneumeth.2016.03.001

[183] Bandettini, P. A., Jesmanowicz, A., Wong, E. C., & Hyde, J. S. (1993ga). Processing strategies for time-course data sets in functional MRI of the human brain. *Magnetic Resonance in Medicine*, *30*(2), 161–73. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/8366797

[184] Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., ... Zhang, Y. (2016gb). Theano: A Python framework for fast computation of mathematical expressions. Retrieved from https://arxiv.org/pdf/1605.02688.pdf

[185] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Research, G. (n.d.-gc). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Retrieved from http://download.tensorflow.org/paper/whitepaper2015.pdf

[186] Greiner, M., Pfeiffer, D., & Smith, R. D. (2000gd). Principles and practical application of the receiver-operating characteristic analysis for diagnostic tests. *Preventive Veterinary Medicine*, *45*, 23–41. Retrieved from https://pdfs.semanticscholar.org/03eb/3828e995faedce166374fe2aabd18474202b.pdf

[187] Waller, L., Walter, H., Kruschwitz, J. D., Reuter, L., Müller, S., Erk, S., & Veer, I. M. (2017ge). Evaluating the replicability, specificity, and generalizability of connectome fingerprints. *NeuroImage*, *158*, 371–377. https://doi.org/10.1016/j.neuroimage.2017.07.016

[188] BET - FslWiki. (n.d.-gf). Retrieved from http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/BET

[189] Andersson, J. (n.d.-gg). lecture fsl 2 Brain Extraction, Registration & EPI Distortion Correction.

[190] Ashburner, J., & Friston, K. (2003gh). Rigid Body Registration and Interpolation. *Human Brain Function: Second Edition*, 635–653. https://doi.org/10.1016/B978-012264841-0/50034-2

[191] Jenkinson, M. (2009gi). Image Registration and Motion Correction, *28*(3), 2009.

[192] Greve, D. N., & Fischl, B. (2009gj). Accurate and robust brain image alignment using boundary-based registration. *NeuroImage*, *48*(1), 63–72. https://doi.org/10.1016/j.neuroimage.2009.06.060

[193] Jenkinson, M., & Smith, S. (2001gk). A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, *5*(2), 143–56. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/11516708

[194] Jenkinson, M., Bannister, P., Brady, M., & Smith, S. (2002gl). Improved optimization for the robust and accurate linear registration and motion correction of brain images. *NeuroImage*, *17*(2), 825–841. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/12377157

[195] Fslutils - FslWiki. (n.d.-gm). Retrieved from http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Fslutils

[196] FSL Slice Timing Correction - User Guide. (n.d.-gn). Retrieved from http://poc.vl-e.nl/distribution/manual/fsl-3.2/slicetimer/index.html

[197] MCFLIRT - FslWiki. (n.d.-go). Retrieved from http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/MCFLIRT

[198] BET2 - Brain Extraction Tool- User Guide. (n.d.-gp). Retrieved from http://poc.vl-e.nl/distribution/manual/fsl-3.2/bet2/

[199] FSLMotionOutliers - FslWiki. (n.d.-gq). Retrieved from http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FSLMotionOutliers

[200] Spatial Smoothing. (n.d.-gr). Retrieved from http://support.brainvoyager.com/functional-analysis-preparation/27-pre-processing/279-spatial-smoothing-in-preparation.html

[201] Exploratory, M., & Optimised, L. (n.d.-gs). lecture fsl 7 Model-free Functional Data Analysis.

[202] Weber, M. (2012gt). Group analysis of resting-state functional connectivity with MELODIC.

[203] MELODIC - FslWiki. (n.d.-gu). Retrieved from http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/MELODIC

[204] ICA Practical. (n.d.-gv). Retrieved from http://fsl.fmrib.ox.ac.uk/fslcourse/lectures/practicals/melodic/

APPENDIX A – PROCESSING RS-FMRI

The implementation of the processing workflow was constructed based in the FSL software due to advantages comparatively to others. So, each pre-processing step was exemplified with respective FSL commands-line with a short explanation of the process and the reasons of its application.

To achieve a better example of the procedure, it was chosen an example, a subject with the name "Caso_teste". So, the main folder with all images and processing had the same subject name, and its directory is represented by "path". Initially, it only had two folders with the images obtained from the acquisition. One contained the images from MRI, the folder "MPRAGE_SAG_2", and the other contained the images from fMRI, the folder "ep2d_bold...". After, it was added three more folders: the folder "mri", the folder "fmri" and the folder "ica", for the respectively processing steps. The final result of all folders can be visualized in Figure A.1.

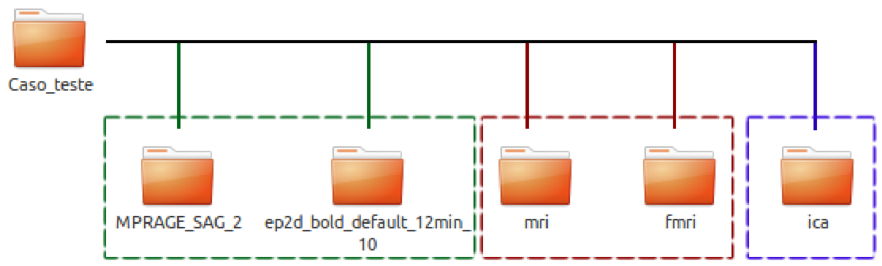


Figure A.1 - Scheme of the folders used in the storage fMRI data and processing for the subject example "Caso_teste".

For a better exemplification, it was created an illustration of all the steps applied with the corresponding commands used, and that included the processing steps, since raw data, MRI and fMRI, until getting the final files as pretended, whose were divided in three main parts: the MRI pre-processing, the fMRI pre-processing and analysis (ICA).

1 Processing image of magnetic resonance imaging (MRI).

The process begins with the MRI, not because the objective is the analysis or study of the last resulting MRI, but because some files there are produced during the process and that are applied in the fMRI pre-processing registration.

In order to a better organization and explanation, all the files resulting from the various processes were placed in the folder "mri" (Figure A.1), thus the path of input and output files are always this folder with only one exception for the input file in the first of process.

1.1 Change of the format of the initial Images of MRI

First it was started with the format transformation of the images .dcm to a single image .nii; in other words, from format DICOM to NifTI-1.

A **DCM** is an image file saved in the Digital Imaging and Communications in Medicine format, created by the National Electrical Manufacturers Association as a standard for distributing and viewing medical images, such as MRI, CT scans, and ultrasound images.[71]

NIfTI-1 is a format adapted from another file format, the ANALYZE^{tm} 7.5, where the NIfTI-1 uses the "empty space" in the ANALYZE 7.5 header to add several new features. Features such as:

- Affine coordinate definitions relating voxel index (*i,j,k*) to spatial location (*x,y,z*);
- Codes to indicate spatio-temporal slice ordering for FMRI;
- "Complete" set of 8-128 bit data types;
- Standardized way to store vector-valued datasets over 1-4 dimensional domains;
- Codes to indicate data "meaning";
- A standardized way to add "extension" data to the header;
- Dual file (.hdr & .img) or single file (.nii) storage;

And many other features. The goal of this format is a best interoperability in the file-exchange between analysis software packages, as FSL, SPM, AFNI and others [72].

To change the image format was used one of the programs available from MRIcron, the dcm2nii. That converts the images DICOM in NIfTI format. Posteriorly allows that these images could be visualized by analysis software as already referenced.

Command: dcm2nii -x Y -o 'path/mri' 'path/MPRAGE_SAG_2/IM-0002-0001.dcm'

Input: in use of this command it is enough give the path of the first image of MRI.

Output: in the folder “mri” was created the NIfTI file “Caso_teste_str.nii.gz”.

1.2 Brain extraction

This pre-processing step is real important, because in the fMRI analysis the only region of interest to study it’s the brain. Only the voxels of the brain will change their intensity values over time in each volume. Thereby it is needed to select only the volume of the brain, which is the region of interest (ROI). For that it is used the BET (Brain Extraction Tool) available in FSL, and what it does is to delete the non-brain tissue from an acquisition of the whole brain. Furthermore, it has another features, like to be able to estimate inner and outer skull surfaces, and outer scalp surface, provided that the quality of images T1 and T2 are good [188].

Command: bet 'path/mri/Caso_teste_str.nii.gz' 'path/mri/Caso_teste_str_bet.nii.gz' -m -f 0.2 -B

Options:

- **-m** generate binary brain mask
- **-f** fractional intensity threshold; from its default value of 0.5 will cause the overall segmented brain to become larger (<0.5) or smaller (>0.5). In this case was applied 0.2.
- **-B** bias field and neck clean-up

Input: The NIfTI file “Caso_teste_str.nii.gz”, created previously.

Output: In the folder “mri” the file “Caso_teste_str_bet.nii.gz”, further the file with the brain mask for extraction only of the brain, which file name is “Caso_teste_str_bet_mask.nii.gz”.

1.3 Registration

This method is indispensable in fMRI processing and will be again approached in fMRI registration. However, this is also performed in MRI processing whose resulting files meant to be applied in the final fMRI registration in the point 2. Here, in this step it’s only used the image from MRI that is transformed in a standard model of respective subject. This allows that this can be used in a set of images or volumes from the same subject, thus the same space characteristics are preserved in all being accordingly to the universal standardization of brain medical images. It is important to refer one more time that this standardization is only relative to brain position and its occupation in the space. For that, it’s required to traduce this model in a values’ matrix that correspond to space vectors, in order to be able to make this model from any other volume of the same subject and acquisition [189].

All of this process it’s intended to compare and have conclusions of several fMRI analysis of acquisitions made in different times and from different subjects. So, to be possible the combination across individuals it is needed that the image data had suffered a “standardization”. Moreover, during the fMRI scanning occurs a lot of motions that can negatively affect the final results. Then, the different voxels that represent the brain don’t retain the same position of volume to volume. Not being in that way possible to have continuity about the voxels localization, it can lead to errors in theirs values conducing to appearance of artefacts and false results. And as this type of medical imaging are associated to studies of brain connectivity, it is fundamental that doesn't happen, or else it will not be possible to reach truly and conclusive results [190].

In the universe of medical imaging, the process and method explained are known and named as Registration. And basically, the registration process is to take in two images and align them, or in other words, one of the image is reshaped to match the other. This is made by finding a relationship between the voxels locations in the two images and, once this is found, the voxel information can be exchanged and combined, being possible the fusion and comparison between them. In addition, the registration has two main categories depending on the registration method, being these feature-based and intensity-based. The difference between them is if it's aligned by manually or automated extracted features or if it's aligned by the intensity voxels values. The intensity-based methods are more common and it was what was used in the registration steps by the FSL software [191].

As said before, during the process of registration occur spatial transformations to image to align, to change shape, orientation, or position of brain structures in the image. So, during a process is applied a transformation model canning this to be of two different types, linear and non-linear. Also, in order to a better description of the model, it's a lot of times described by its Degrees Of Freedom (DOF), which is the number of transformations by independently ways that can be realized [189];[191].

Finally, in this processing example was used both registration type models, first here for pre-processing MRI image and later for fMRI acquisition.

1.3.1 Linear Registration

The application of linear registration was made by the use of the program available by FSL, more properly, the flirt command-line program [181];[182];[183]. The flirt is a fully automated and accurate program to deal and to perform the linear intra-modal and inter-modal brain image registration. As linear transformations can be two types, also can be the registration flirt, being characterized by different DOF [191]:

- Rigid-Body Transformations (6 DOF) – the only transformations permitted are rotations and translations, so each one has 3 DOF, one for each axis. And as the name says, this describes the type of movements of a rigid-body.
- Affine Transformations (12 DOF) – allows tall linear coordinate transformations that can-do translations, rotations, scaling and skew parameters with 3 DOF each of them. Thus, this type of transformation allows both size and shape change for structures in the image.

In this pre-processing was used the affine transformations due to the more functionality, which has larger number of associated transformations.

```
Command: flirt -searchrx -180 180 -searchry -180 180 -searchrz 180 -180 -dof 12 -cost normcorr -in 'path/mri/Caso_teste_str_bet.nii.gz' –ref /usr/share/FSL/5.0/data/standard/MNI152_T1_1mm_brain.nii.gz -omat 'path/mri/Caso_teste_str_2mni.mat' -out 'path/mri/Caso_teste_str_bet_affine_mni.nii.gz'
```

Options:

- **-searchrx** <min_angle> <max_angle>
- **-searchry** <min_angle> <max_angle>
- **-searchrz** <min_angle> <max_angle>
- **-dof** degrees of freedom
- **-cost** cost function; in this case *normcorr*
- **-in** input volume
- **-ref** reference volume
- **-omat** output matrix
- **-out** output volume

Input: file "Caso_teste_str_bet.nii.gz" with only the brain after its extraction.

Output: the file "Caso_teste_str_2mni.mat" with transformation matrix and the brain now with transformations processed, which file name is "Caso_teste_str_bet_affine_mni.nii.gz".

1.3.2 Non-linear registration

For this pre-processing was applied other registration program by FSL, the fnirt. The fnirt is characterized by non-linear transformations, where are included all transformations that aren't in the affine registration. And it is associated a countless quantity of DOF. So, for this registration comparatively to non-linear, instead of being represented by a matrix of values, is represented by a deformation field.

```
Command: fnirt –in='path/mri/Caso_teste_str.nii.gz' –aff='path/mri/Caso_teste_str_2mni.mat' –cout='path/mri/Caso_teste_str_warp.nii.gz' –config=T1_2_MNI152_2mm
```

Options:

- **–in** name of input image

- **–aff** name of file containing affine transform
- **–cout** name of output file with field coefficients
- **–config n**ame of *config* file specifying command line arguments (standard image to compare)

Input: file "Caso_teste_str.nii.gz" with the initial volume MRI, and the file "Caso_teste_str_2mni.mat'" with transformations realized by flirt command.

Output: in the current folder was created the file "Caso_teste_str_warp.nii.gz" with the new resulting volume from the fnirt application.

Then it was used the *applywarp* command for to apply the warps estimated by fnirt to some image (volume).

Command: applywarp –ref=$FSLDIR/data/standard/MNI152_T1_2mm_brain –in=$'path/mri/Caso_teste_str_bet.nii.gz' –warp='path/mri/Caso_teste_str_warp.nii.gz' –out='path/mri/Caso_teste_str_bet_mni.nii.gz'

Options:

- **–in** name of input image
- **–ref** filename for reference image
- **–warp** filename for warp/coefficient (volume)
- **–out** name of *config* file specifying command line arguments (standard image to compare)

Input: It has as input the file "Caso_teste_str_bet.nii.gz" only with the brain's volume. The volume had as reference named "MNI152_T1_2mm_brain"and the resulting volume from fnirt the file "fnirt Caso_teste_str_warp.nii.gz".

Output: the file "Caso_teste_str_bet_mni.nii.gz" that is obtained at the end the brain completely normalized and ready to be used in the next steps of image functional processing.

2 Processing of functional magnetic resonance images

As was made for processing MRI images, also for this second part of pre-processing steps, it was created a specific folder to save all the outputs resultants, creating the folder named "fmri" - Figure A.1.

2.1 Alteration of the format of the initial Images of fMRI

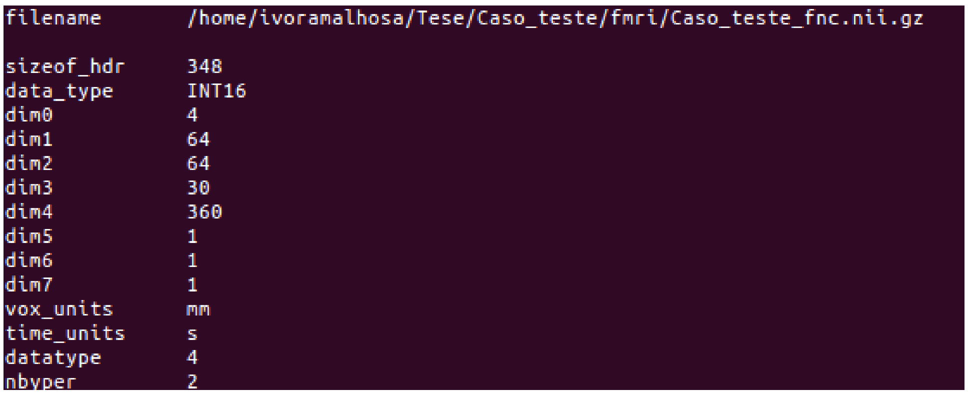
Like happens for the MRI images, it was needed to convert the format of the fMRI images from DICOM to NIfTI. And as before, the program dcm2nii from MRIcon was used to the conversion, such as it is indicated in the command below:

Command: dcm2nii 'path/ep2d_bold_default_12min_10'

2.2 Eliminate possible artefacts of the initial signal

In an effort to minimize artefacts due to not stabilization of the signal in the first extracted volumes, it is removed the first five volumes. So, for this case, where TR is 2 seconds, it was removed the first 10 seconds from the acquisition.

For support as help tool to see the metadata from a specific fMRI acquisition, it can be used the *fslhd* command that retrieves the information in the nifty header of the file. In this step, this command was used to verify the number of volumes of all acquisition, like is demonstrated in the Figure A.2.



```
filename       /home/ivoramalhosa/Tese/Caso_teste/fmri/Caso_teste_fnc.nii.gz

sizeof_hdr     348
data_type      INT16
dim0           4
dim1           64
dim2           64
dim3           30
dim4           360
dim5           1
dim6           1
dim7           1
vox_units      mm
time_units     s
datatype       4
nbyper         2
```

Figure A.2. Header NIfti from the file "Caso_teste_fnc.nii.gz" retrieved by fslhd command.

Looking to the results retrieved, it's possible to find that in this acquisition there was an extraction of 360 volumes, wherefore it is intended to have 355 volumes in the final. For that, it was used another FSL command, the *fslroi*.

The command *fslroi* extracts the region of interest (ROI) from an image, and can be used for three different means [195]:

1. Take a 3D ROI from a 3D dataset (or, if it is 4D, the same ROI is taken from each time point and a new 4D dataset is created);
2. Extract just some time points from a 4D dataset;
3. Control time and space limits of the ROI.

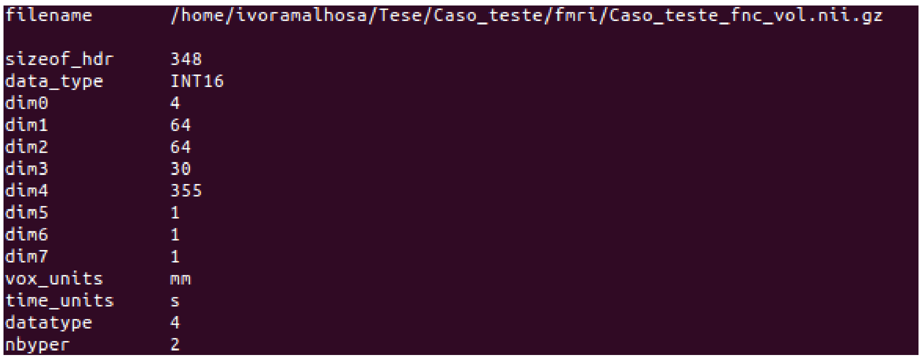
Taking into account the characteristics of the mentioned case, it was used the following command in *fslroi* application.

Command: fslroi 'path/fmri/Caso_teste_fnc.nii.gz' 'path/fmri/Caso_teste_fnc_vol.nii.gz' 5 360

Input: file" Caso_teste_fnc.nii.gz" resulting of the conversion process previously, where will occur the process of selection's volumes pretended.

Output: file "Caso_teste_fnc_vol.nii.gz" with only the waned volumes (in this case the 355 volumes instead of the 360 initial volumes).

Again, it can be used the command *fslhd* to verify if the process before ran like it was intended to. So, only it is needed to use this command for the last file created, the "Caso_teste_fnc_vol.nii.gz", and the results now are like it's in image beneath (Figure A.3).



```
filename       /home/ivoramalhosa/Tese/Caso_teste/fmri/Caso_teste_fnc_vol.nii.gz

sizeof_hdr     348
data_type      INT16
dim0           4
dim1           64
dim2           64
dim3           30
dim4           355
dim5           1
dim6           1
dim7           1
vox_units      mm
time_units     s
datatype       4
nbyper         2
```

Figure A.3. Header NIfti from the file "Caso_teste_fnc_vol.nii.gz" retrieved by fslhd command.

2.3 Slice Timing Correction

For this process, it was used the command *slicetimer* from FSL software. The **slicetimer** is a pre-processing tool designed to correct sampling offsets inherent in slice-wise EPI acquisition sequences. In the process each voxel time-series is analysed and processed independently, and intensities are shifted in time, so that they reflect the interpolated value of the signal at a common reference time point for all the voxels. So, it is applied an interpolation synchronized by Hanning windowing kernel to each time course to calculate the interpolated values resulting. Additionally, is also needed to know in what order the slices were acquired, for known if this was acquires from the bottom or the top of the brain or if this acquire in the slice 1 or 2, for example [196].

Command: slicetimer -i 'path/fmri/Caso_teste_fnc_vol.nii.gz' -o 'path/fmri/Caso_teste_fnc_vol_stime.nii.gz' –odd -r 2 –ocustom='path/fmri/slicetimmings' -v

Options:

- **-i , –in** filename of input *timeseries*

- **-o, –out** filename of output *timeseries*
- **–odd** use interleaved acquisition
- **-r, –repeat** specify TR of data - default is 3s
- **-ocustom** filename of single-column custom interleave order file
- **-v, –verbose** switch on diagnostic messages

Input: The input files are two. One of them is the file with fMRI acquisition where happens the process of time correction and the other it's a file with the order of cuts realized for each repetition time (TR). This to be made appropriately the correction over time taking into account the how the various slices were extracted. To that it was created one new file, the "slicetimings", wherein it was placed the correct order of the slices through application of the slice number in each line of the document. And

Output: At the end it's obtained the corrected acquisition on what the file name is "Caso_teste_fnc_vol_stime".

2.4 Motion correction

In this step it was applied the *fslmaths* command, which it is a simple but powerful program that allows mathematical manipulation of images. In the present, the application includes spatial and temporal filtering, statistic conversion, diffusion tensor decomposition, and TFCE calculation [195].

Command: fslmaths 'path/fmri/Caso_teste_fnc_vol_stime.nii.gz' -Tmean 'path/fmri/Caso_teste_fnc_vol_stime_mean.nii.gz'

Options:

- **-Tmean** mean across time

Input: It was the more recent file resulting from the last processing step, so it's the file "Caso_teste_fnc_vol_stime.nii.gz".

Output: It has only one volume with average values of each voxel of the entire fMRI acquisition.

After the previous command, it was used the file created as reference, that is the average of intensity of each voxel over time, and it was used in the command *MCFLIRT*.

MCFLIRT is an intra-modal motion correction tool designed to use on fMRI time series and it is based on optimization and registration techniques used in FLIRT, a fully automated robust and accurate tool for linear (affine) and inter-modal brain image registration.[197][194]

Command: mcflirt -in 'path/fmri/Caso_teste_fnc_vol_stime.nii.gz' -r 'path/fmri/Caso_teste_fnc_vol_stime_mean.nii.gz' -stages 4 -plots

Options:

- **-in** filename of input
- **–r** use a separate 3d image file as the target for registration
- **–stages** default is 3. 4 specifies final (internal) sinc interpolation.
- **–plots** save transformation parameters in file *outputfilename.par*.

Input: The same file used in the input before that is the file "Caso_teste_fnc_vol_stime.nii.gz", and, as already mentioned, the file "Caso_teste_fnc_vol_stime_mean.nii.gz" was used as reference in the correction of all existing volumes in the acquisition.

Output: In the final of the process, it resulted in two files: one with the total fMRI acquisition corrected over time having as name "Caso_teste_fnc_vol_stime_mcf.nii.gz", and the other with the corrections matrix done for each one of the 355 volumes over of the 6 degrees of freedom possible and which name was "Caso_teste_fnc_vol_stime_mcf.par".

After, in order to analyse and visualise the results obtained relatively to translation according the three dimensions all over the volumes in the motion correction, it was used the *fsl_tsplot* command.

Command: fsl_tsplot -i 'path/fmri/Caso_teste_fnc_vol_stime_mcf.par' -t 'Headmovement - translation' –start=4 –finish=6 -a x,y,z -o 'path/fmri/Caso_teste_fnc_vol_stime_trans.nii.gz'

Options:

- **–I or –in** comma-separated list of input file names
- **–o or –out** output filename for the PNG file
- **–t or –title** plot title
- **–start** position of first column to plot
- **–finish** position of final column to plot
- **–a or –labels** comma-separated list of labels

Input: As this command it's only for the constructions demonstrative graphics, merely has to process one matrix of values already made. This matrix was obtained previously in the application of the *mcflirt* command that has resulted in the file's matrix "Caso_teste_fnc_vol_stime_mcf.par". But, in this command, the unique interest is relatively to translate and not to rotate, so it was needed to select the pretended columns, and hence it had selected the columns between 4 and 6 inclusive.

Output: In the final it resulted in the graphic as is represented in Figure A.*4*, where it has all the translations for all the volumes for the space of three variables from the acquisition "Caso_teste_fnc_vol_stime_trans.nii.gz".

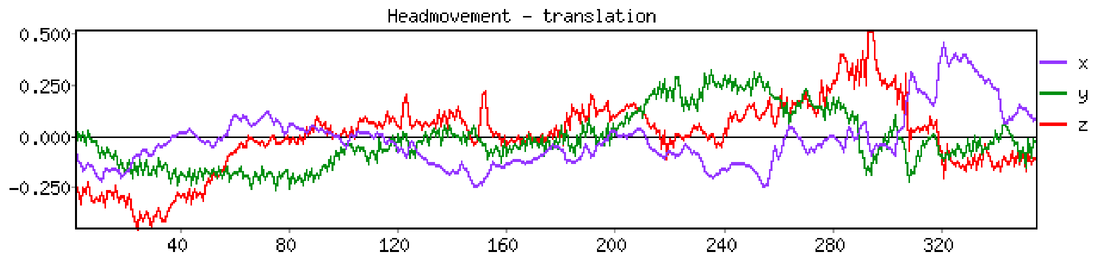


Figure A.4. Representation translation head movement over all volumes using the fsl_tsplot command.

Command: fsl_tsplot -i 'path/fmri/Caso_teste_fnc_vol_stime_mcf.par' -t 'Headmovement - rotation(rad)' –start=1 –finish=3 -a x,y,z -o 'path/fmri/Caso_teste_fnc_vol_stime_rot.nii.gz'

Input: This command is equal to before, so it has the same input matrix file, but in this case, it was intended only the values related to rotation according to three spaces variables and, for that reason,

Output: A graphic with rotation in radians of the head movement over all volumes as it's showed on below in the Figure A.*5*.

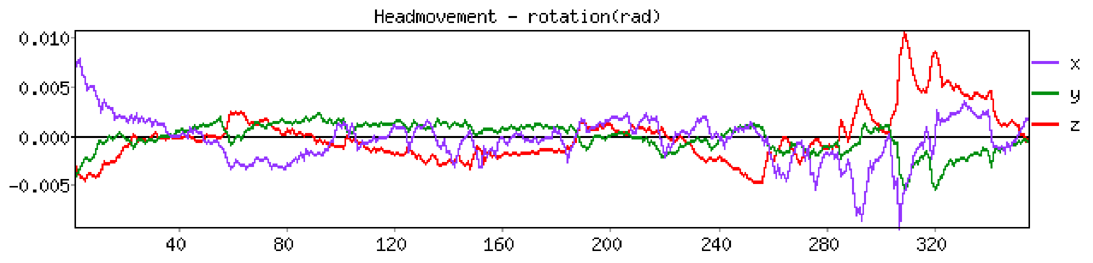


Figure A.5. Representation rotation head movement over all volumes using the fsl_tsplot command.

2.5 Brain extraction

Before to execute the principal extraction process, it is used again the command *fslmaths* to obtain a volume that for each voxel this has the average intensity's voxel of all volumes existing in the acquisition. Then, this is used as reference in the process for a better brain extraction had in account all acquisition.

Command: fslmaths 'path/fmri/Caso_teste_fnc_vol_stime_mcf.nii.gz' -Tmean 'path/fmri/Caso_teste_fnc_vol_stime_mcf_mean.nii.gz'

Input: File "Caso_teste_fnc_vol_stime_mcf.nii.gz" with fMRI acquisition after the movement correction performed in the preceding step.

Output: A file "Caso_teste_fnc_vol_stime_mcf_mean.nii.gz" that it's an average volume from all volumes.

Then, for brain extraction it was applied the *bet2* command, contrary to the processing made in MRI with *bet* command. Although *bet2* has practically the same functionality comparatively to a *bet* command, therefore can be saw like an upgraded version. Even the same command-line syntax can be used in the two commands. The unique main addition in *bet2* is the *–e* option to outputting a "mesh" version of the estimated brain mask [198].

Command: bet2 'path/fmri/Caso_teste_fnc_vol_stime_mcf_mean.nii.gz' 'path/fmri/Caso_teste_fnc_vol_stime_mcf_mean_bet.nii.gz' -m -f

Options:

- **-m,–mask** generate binary brain mask;
- **-f** fractional intensity threshold (0->1); default=0.5;

Input: the file "Caso_teste_fnc_vol_stime_mcf_mean.nii.gz" obtained in the previous command-line program.

Output: As result it had two files: one with only volume of the brain - the file "Caso_teste_fnc_vol_stime_mcf_mean_bet.nii.gz", and the other is a file with a mask. In this mask the voxel intensity only has two values, so it works like delimiter of Region Of Interest (ROI). In this case the ROI was the brain and it was created from the average volume of the acquisition in order that this mask be the best in defining the brain in all volumes of the acquisition.

In the last step was fulfilled the brain extraction and, for that, it was used the command *fslmaths* to extract the brain from each volume in the fMRI after motion correction. For that it was used a mask obtained previously.

Command: fslmaths 'path/fmri/Caso_teste_fnc_vol_stime_mcf.nii.gz' -mas 'path/fmri/Caso_teste_fnc_vol_stime_mcf_mean_bet.nii.gz' 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet.nii.gz'

Options:

- **-m** use (following image>0) to mask current image

Input: the most recent file with fMRI acquisition after motion correction, file "Caso_teste_fnc_vol_stime_mcf.nii.gz".

Output: A file with the fMRI acquisition but just with the brain for each volume, because merely this region matters, since only the intensity of this voxels will variate over time.

2.6 Registration

For this registration, like it was made before for the MRI acquisition, it was made the registration by all axes, with 6 degrees of freedom though it was used another cost function, the *corratio* function. In this process to a better registration process is used the volume with the brain already spatial normalized from the final pre-processing step of the MRI data.

Command: flirt -searchrx -180 180 -searchry -180 180 -searchrz -180 180 -dof 6 -cost corratio -in 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet.nii.gz' -ref 'path/mri/Caso_teste_str_bet.nii.gz' -omat 'path/fmri/fnc_2str.mat'

Options:

- **-searchrx** <min_angle> <max_angle>

- **-searchry** <min_angle> <max_angle>
- **-searchrz** <min_angle> <max_angle>
- **-dof** degrees of freedom
- **-cost** cost function, in this case *normcorr*
- **-in** input volume
- **-ref** reference volume
- **-omat** output matrix

Input: the most recent fMRI acquisition's file with only the brain and, time and motion corrections were made, so it was the output file of the previous command-line program, the file "Caso_teste_fnc_vol_stime_mcf_bet.nii.gz". Besides is use a reference brain obtained in the final of structural MRI data, the "Caso_teste_str_bet.nii.gz".

Output: the resulting file was a matrix with result values from flirt, that after should be applied to correct the acquisition in some voxels problems.

Then it is only necessary to use the command *applywarp* in order to apply to fMRI acquisition the registration flirt with the resulting matrix.

Command: applywarp –ref=$FSLDIR/data/standard/MNI152_T1_2mm_brain –in=$'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet.nii.gz' –warp='path/mri/Caso_teste_str_warp.nii.gz' –premat='path/fmri/fnc_2str.mat' –out='path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni.nii.gz'

Options:

- **–in** name of input image
- **–ref** filename for reference image
- **–warp** filename for warp/coefficient (volume)
- **–premat** filename for pre-transform (affine matrix)
- **–out** filename for output (warped) image

Input: the main file of this command/process its fMRI acquisition most recent and that was used in the flirt command, the file “Caso_teste_fnc_vol_stime_mcf_bet.nii.gz”. Then it’s necessary the utilization of a standard reference, so it was used the “MNI152_T1_2mm_brain” from the standards images given by FSL packages. Moreover, there was another input file, a file with matrix’s affine of the last flirt command, the file “fnc_2str.mat” and, lastly, the file “Caso_teste_str_warp.nii.gz” with the mitigation needed for each volume, and made in the MRI processing by application of fnirt command-line program.

Output: resulted in a file fMRI acquisition’s brain already with the registration process done.

2.7 Motion scrubbing and removal of confounding factors

To know the average values in the different volumes of the CSF and white matter, it was applied the command *fslmeants*.

Command: fslmeants -i 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni.nii.gz' -o 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_csf+white.txt' –label='/home/ivoramalhosa/Tese/white+csf.nii.gz'

Options:

- **–i** input 4D image
- **–label** label input 3D label image
- **–o** output text matrix

Input: It’s the most recent file from the previous process, so the file “Caso_teste_fnc_vol_stime_mcf_bet_mni.nii.gz” and a volume “white+csf.nii.gz” that works like a mask where were defined the regions of CSF and white matter as regions of interest.

Output: It resulted in the file that has the intensity average of the voxels of white matter and CSF for each volume that existed in the input fMRI acquisition. As the acquisition was constituted by 355 volumes, this file had 355 lines with two columns for the average of each region.

After this, it was used the command *fsl_motion_outliers*. This tool is designed to detect time points in an fMRI dataset that has been corrupted by large motion. It creates a confound matrix that can be used in the GLM to completely remove the effects of these time points on the analysis, without any adverse effects in the statistics. This is intended to deal with the effects of intermediate to large motions, which corrupt images beyond anything that the linear motion parameter regression methods can fix [199].

```
Command: fsl_motion_outliers -i 'path/fmri/Caso_teste_fnc_vol_stime_mcf.nii.gz' -o
'path/fmri/Caso_teste_fnc_vol_stime_mcf_scrubbed' -s
'path/fmri/Caso_teste_fnc_vol_stime_mcf_DVARS.txt' –nomoco
```

Options:

- **-i** input 4D image
- **-s** save metric values (e.g. DVARS) as text into specified file
- **-o** output confound file
- **–nomoco** do not run motion correction

Input: The fMRI acquisition before brain extraction, file "Caso_teste_fnc_vol_stime_mcf.nii.gz".

Output: Result in two files .txt as requested, wherein in one had confound matrix with each column indicated which volume had outliers, the file "Caso_teste_fnc_vol_stime_mcf_scrubbe". And in the other, it was found DVARS value for each volume, the file "Caso_teste_fnc_vol_stime_mcf_DVARS.txt". On what DVARS is root mean square intensity

At the end of this second command-line program it was needed to merge the three documents .txt created before. So, for that, it was applied the command *paste* available from command-line bash of OS. The file merged was "Caso_teste_fnc_vol_stime_mcf.par", with information for motion correction in each volume of the fMRI scan in the 6 DOF. Moreover, the file Caso_teste_fnc_vol_stime_mcf_bet_mni_csf+white.txt with values' mean intensity of the voxels in the regions CSF and white matter. And finally, the last file obtained, Caso_teste_fnc_vol_stime_mcf_scrubbed. After, it resulted in the file "Caso_teste_fnc_nuisance_reg_wScrubbing.txt" with all columns of each one of the input files. Thus, the complete command applied was the following:

```
Command: paste -d \ 'path/fmri/Caso_teste_fnc_vol_stime_mcf.par'
'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_csf+white.txt'
'path/fmri/Caso_teste_fnc_vol_stime_mcf_scrubbed' >
'path/fmri/Caso_teste_fnc_nuisance_reg_wScrubbing.txt'
```

Finally, it reaches at last and main command of this step, the *fsl_glm* command. This command is the implementation of a statistic modelling named general linear model (GLM). So, without delving too much, this method wants to predict response Y for each voxel through linear combination's modelling of one or more predictors that are stored in the columns of a determinate "design matrix". Additionally, it is also possible to get the important residuals to analyse and get the values that can't be explained by the GLM design matrix - the errors of GLM.

This command was not applied with the direct intention of its main application. Because of what was wanted, it was the file's residuals output corresponding to acquisition without noise and artefacts, since these were predicted by the glm method from the design matrix given and, therefore, explained in this statistical form. So, it was used as a design matrix of the file resulting on the merge of 3 files relative problems' fMRI volumes of motion or other problems.

Command: fsl_glm -i 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni.nii.gz' -d 'path/fmri/Caso_teste_fnc_nuisance_reg_wScrubbing.txt' –out_res='path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing.nii.gz' –demean -m $FSLDIR/data/standard/MNI152_T1_2mm_brain_mask.nii.gz

Options:

- **-i,–in** input file name
- **-d,–design** file name of the GLM design matrix
- **–out_res** output file name for residuals
- **–demean** switch on de-meaning of design and data
- **-m,–mask** mask image file name if input is image

Input: It's the newest fMRI acquisition with more correction processes that resulted of the registration process accordingly to the file "Caso_teste_fnc_vol_stime_mcf_bet_mni.nii.gz**".**

Output: The output file it's an acquisition as mentioned without several problems related to motion artefacts.

2.8 Smoothing or spatial filtering

Again, it's used the command-line program *fslmaths*, and now it was to do the spatial filtering. Spatial filtering or spatial smoothing means that in the process in itself voxels are averaged with their neighbours. Thus, it has an effect like a low pass filter wherein the low frequencies are amplified, while the high frequencies are removed. So, the result are images blurred and the spatial correlation between them is stronger. During the process, the fMRI signal is convolved with a Gaussian function with a width pretended - width that has to be chosen carefully not to create problems associated to reducing's spatial resolution [200].

Command: fslmaths 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing.nii.gz' -kernel gauss 3.397 -fmean -mas /usr/share/FSL/5.0/data/standard/MNI152_T1_2mm_brain_mask 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing_smooth.nii.gz'

Options:

- **-kernel gauss** gaussian kernel (sigma in mm, not voxels)
- **-fmean** mean filtering, kernel weighted (conventionally used with gauss kernel)
- **-mas** use (following image>0) to mask current image

Input: It's the newest fMRI acquisition with more correction processes that resulted of the registration process accordingly the file "Caso_teste_fnc_vol_stime_mcf_bet_mni.nii.gz".

Output: The output file it's an acquisition as mentioned without several problems related to motion artefacts.

2.9 Band pass temporal filtering

This was the last process in the pre-processing steps made, because time series of each voxel contains scanner-related and physiological signals, and high frequency noise. Hence, it was used a band pass to remove signals of high and low frequency which, without, of course, there would be a loss of important signals especially low frequency signals.

Command: fslmaths 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing.nii.gz' -bptf 25.000000 3.125000 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing_filter.nii.gz'

Options:

- **-bptf** <hp_sigma> <lp_sigma> Band pass temporal filtering; nonlinear high pass and Gaussian linear low pass (with sigmas in volumes, not seconds); set either sigma<0 to skip that fil.

Input: the file "Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing.nii.gz" resulting from the last pre-processing step. The interval of values chosen for the band pass temporal filtering was between 0,01 Hz and 0,08 Hz, corresponding respectively to periods (T) of 100 seconds and 12,5 seconds. As each TR corresponding to one volume has the duration of 2 seconds, and the command works with volumes, the time is traduced in a number of volumes - this is 50 volumes and 6,25 volumes, respectively. Then, in order to not have signal loss, the Nyquist sampling theorem has to be respected, so the number of each volumes is reduced to half, resulting in the final values of 25 volumes and 3,125 volumes.

Output: acquisition corrected in order to high and low frequency noises, such as is in the file "Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing_filter.nii.gz".

3. Data analysis example

The output results of this process were saved in the folder "ICA", which means *Independent Component Analyses*. This was the process implemented by the program Melodic, which is a FSL fully automated tool that decomposes a multiple or, as in this case, a single 4D data set into the maximum time-courses and spatial maps using ICA. For that, the ICA separate statistically and independently the patterns of variation of each voxel time series in order to identify possible brain networks relating to these variations [190];[191];[192];[193].

Command:

```
melodic -i 'path/fmri/Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing_smooth_filter.nii.gz' -o 'path/ica' –report –nobet –tr=2
```

Options:

- **-i** input file names
- **-o** output directory name
- **–report** generate Melodic web report
- **–tr** TR in seconds
- **–nobet** switch off BET

Input: The file resulting from the total fMRI processing: "Caso_teste_fnc_vol_stime_mcf_bet_mni_denoised_wScrubbing_smooth_filter.nii.gz". Also, the repetition time is mentioned in order to a better analysis process.

Output: This command involves analysis application, therefore it results are a large number of statistics files, log files and others. But, besides that resulted in a file with the various components found in the acquisition, that explicit more or less significantly brains connectivity. This file had as name "melodic_IC.nii.gz", and for this test case were found 24 different components. Additionally, as requested by option report, the process has created a folder, also named report, wherein it has files .txt and .html about statistical values for each one of the components found.

APPENDIX B – DEVELOPMENT ENVIRONMENT

1. Theano GPU/CPU Test

```
from Theano import function, config, shared, tensor
import numpy
import time

vlen = 10 * 30 * 768# 10 x# cores x# threads per core
iters = 1000

rng = numpy.random.RandomState(22)
x = shared(numpy.asarray(rng.rand(vlen), config.floatX))
f = function([], tensor.exp(x))
print(f.maker.fgraph.toposort())
t0 = time.time()
for i in range(iters):
     r = f() t1 = time.time()

print("Looping %d times took %f seconds" % (iters, t1 t0))
print("Result is %s" % (r, ))
if numpy.any([isinstance(x.op, tensor.Elemwise) and('Gpu'not in
type(x.op).__name__) for x in f.maker.fgraph.toposort()]):
     print('Used the cpu')
else :
     print('Used the gpu')
```

2. Processing

- **Configuration Files**

```
def create_file_configurationOfProcessing(path):
f = open(path,"w")
    f.write('Folder with the MRI and FMRI Data in the base
directory  = "Data"\n')
    f.write('Folder with FMRI Data = "FMRI"\n')
    f.write('Folder with MRI Data = "MRI"\n')
    f.write('Sufix name of MRI image after subject id =
"_str_crop.nii.gz"\n')
    f.write('Sufix name of fRMI image after subject id =
"_fnc.nii.gz"\n')
    f.write('Processing folder name = "Processing"\n')
    f.write('Folder name of the output processing results  =
"Processing_Results"\n')
    f.write('Folder name of the output first part (bet) MRI
processing results= "MRI_bet_Results"\n')
    f.write('Folder name of the output second part MRI
processing results= "MRI_rest_Of_results"\n')
    f.write('Folder name of the output with the functional
connectivity = "Connectivity_data_final_results"')
    f.close

def create_file_configurationOfProcessing_HCP(path):
    f = open(path,"w")
    f.write('Folder with preprocessed data (subjects folder) in
the base directory  = "Data"\n')
    f.write('Path to folders with results for LR and RL phase  =
"/MNINonLinear/Results"\n')
    f.write('Folder name with LR phase data =
"rfMRI_REST1_LR"\n')
    f.write('Folder name with RL phase data =
"rfMRI_REST1_RL"\n')
    f.write('Name of fRMI volume as input for LR phase =
"rfMRI_REST1_LR.nii.gz"\n')
    f.write('Name of fRMI volume as input for RL phase =
"rfMRI_REST1_RL.nii.gz"\n')
    f.write('File name with regressors of movement =
"Movement_Regressors.txt"\n')
    f.write('Processing folder name = "Processing"\n')
    f.write('Folder name of the output processing results  =
"Processing_Results"\n')
    f.write('Folder name of the output with the functional
connectivity = "Connectivity_data_final_results"')
    f.close
```