On Pattern Recognition of Brain Connectivity
in Resting-State Functional MRI

Ana Catarina Salgueiro Cardoso

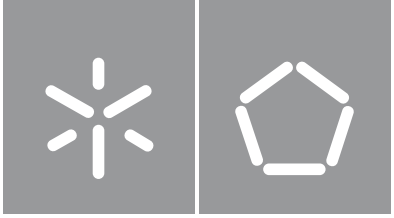
UMinho | 2019

Universidade do Minho
Escola de Engenharia

Ana Catarina Salgueiro Cardoso

On Pattern Recognition of Brain Connectivity in Resting-State Functional MRI

December 2019



Universidade do Minho
Escola de Engenharia

Ana Catarina Salgueiro Cardoso

On Pattern Recognition of Brain Connectivity in Resting-State Functional MRI

Master's Dissertation
Integrated Master's in Biomedical Engineering
Specialization on Medical Informatics

Work performed under supervision of
Victor Manuel Rodrigues Alves
José Miguel Montenegro Soares

December 2019

Declaration

Name: Ana Catarina Salgueiro Cardoso

Dissertation Title: On Pattern Recognition of Brain Connectivity in Resting-State Functional MRI

Advisors: Victor Manuel Rodrigues Alves, José Miguel Montenegro Soares

Year of Conclusion: 2019

Master's Degree Designation: Integrated Master's in Biomedical Engineering

Specialization Area: Medical Informatics

This is an academic work that can be used by third parties as long as internationally accepted rules and good practice regarding copyright and related rights are respected. Thus, the present work can be used under the terms of the license indicated below. If users need permission to make use of the work under conditions not provided for in the indicated licensing, they should contact the author through the RepositóriUM of University of Minho.

Attribution-NonCommercial-NoDerivatives
CC BY-NC-ND
https://creativecommons.org/licenses/by-nc-nd/4.0/

University of Minho, ___ / ___ / ___

Signature: _____

Acknowledgments

I am truly grateful to my supervisor Professor Victor Alves for his continuous help and concern. His dedication and all the suggestions were fundamental for this dissertation becoming real. To José Soares, my co-supervisor, and Ricardo Magalhães from ICVS, thank you for assisting me whenever I needed.

To the ones who have joined me in this academic journey five years ago and kept by my side, thank you for the memories.

To my boyfriend, thank you for patience, and motivating words.

To my family, thank you for supporting me and providing me with the very best.

Statement of Integrity

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, ___ / ___ / ___

Signature: _____

Abstract

The human urge and pursuit for information have led to the development of increasingly complex technologies, and new means to study and understand the most advanced and intricate biological system: the human brain. Large-scale neuronal communication within the brain, and how it relates to human behaviour can be inferred by delving into the brain network, and searching for patterns in connectivity. Functional connectivity is a steady characteristic of the brain, and it has been proved to be very useful for examining how mental disorders affect connections within the brain. The detection of abnormal behaviour in brain networks is performed by experts, such as physicians, who limit the process with human subjectivity, and unwittingly introduce errors in the interpretation. The continuous search for alternatives to obtain faster and robuster results have put Machine Learning and Deep Learning in the leading position of computer vision, as they enable the extraction of meaningful patterns, some beyond human perception.

The aim of this dissertation is to design and develop an experiment setup to analyse functional connectivity at a voxel level, in order to find functional patterns. For the purpose, a pipeline was outlined to include steps from data download to data analysis, resulting in four methods: Data Download, Data Preprocessing, Dimensionality Reduction, and Analysis. The proposed experiment setup was modeled using as materials resting state fMRI data from two sources: Life and Health Sciences Research Institute (Portugal), and Human Connectome Project (USA). To evaluate its performance, a case study was performed using the In-House data for concerning a smaller number of subjects to study. The pipeline was successful at delivering results, although limitations concerning the memory of the machine used restricted some aspects of this experiment setup's testing.

With appropriate resources, this experiment setup may support the process of analysing and extracting patterns from any resting state functional connectivity data, and aid in the detection of mental disorders.

Keyworks: Brain Connectivity, Data Visualization, Machine Learning, Medical Imaging Informatics, Pattern Recognition.

Resumo

O desejo e a busca intensos do ser humano por informação levaram ao desenvolvimento de tecnologias cada vez mais complexas e novos meios para estudar e entender o sistema biológico mais avançado e intrincado: o cérebro humano. A comunicação neuronal em larga escala no cérebro, e como ela se relaciona com o comportamento humano, pode ser inferida investigando a rede neuronal cerebral e procurando por padrões de conectividade. A conectividade funcional é uma característica constante do cérebro e provou ser muito útil para examinar como os distúrbios mentais afetam as conexões cerebrais. A deteção de anormalidades em imagens de ressonância magnética é realizada por especialistas, como médicos, que limitam o processo com a subjetividade humana e, inadvertidamente, introduzem erros na interpretação. A busca contínua de alternativas para obter resultados mais rápidos e robustos colocou as técnicas de *machine learning* e *deep learning* na posição de liderança de visão computacional, pois permitem a extração de padrões significativos e alguns deles para além da percepção humana.

O objetivo desta dissertação é projetar e desenvolver uma configuração experimental para analisar a conectividade funcional ao nível do voxel, a fim de encontrar padrões funcionais. Nesse sentido, foi delineado um *pipeline* para incluir etapas a começar no download de dados até à análise desses mesmos dados, resultando assim em quatro métodos: Download de Dados, Pré-processamento de Dados, Redução de Dimensionalidade e Análise. A configuração experimental proposta foi modelada usando dados de ressonância magnética funcional de *resting-state* de duas fontes: Instituto de Ciências da Vida e Saúde (Portugal) e *Human Connectome Project* (EUA). Para avaliar o seu desempenho, foi realizado um estudo de caso usando os dados internos por considerar um número menor de participantes a serem estudados. O pipeline foi bem-sucedido em fornecer resultados, embora limitações relacionadas com a memória da máquina usada tenham restringido alguns aspetos do teste desta configuração experimental.

Com recursos apropriados, esta configuração experimental poderá servir de suporte para o processo de análise e extração de padrões de qualquer conjunto de dados de conectividade funcional em *resting-state* e auxiliar na deteção de transtornos mentais.

Palavras-chave: Conectividade Cerebral, Descoberta de Padrões, Informática de Imagem Médica, *Machine Learning*, Visualização de Dados.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

1D One-Dimensional.

2D Two-Dimensional.

3D Three-Dimensional.

ACN Anticorrelated Network.

AI Artificial Intelligence.

ANN Artificial Neural Network.

BOLD Blood Oxygenation Level Dependent.

CBF Cerebral Blood Flow.

CNN Convolutional Neural Network.

CSF Cerebrospinal Fluid.

DL Deep Learning.

DMN Default Mode Network.

EEG Electroencephalography.

EPI Echo-Planar Imaging.

FC Functional Connectivity.

FLAIR Fluid Attenuated Inversion Recovery.

fMRI functional Magnetic Resonance Imaging.

FNN Feedforward Neural Netwoork.

FWHM Full Width at Half Maximum.

HRF Hemodynamic Response Function.

MEG Magnetoencephalogram.

ML Machine Learning.

MLP Multilayer Perceptron.

MRI Magnetic Resonance Imaging.

MSE Mean Squared Error.

PET Positron Emission Tomography.

ReLU Rectified Linear Unit.

RF Radio-Frequency.

RNN Recurrent Neural Netwoork.

rsfMRI Resting State Functional Magnetic Resonance Imaging.

RSN Resting State Network.

SC Structural Connectivity.

TanH Hyperbolic Tangent.

TE Echo Time.

TR Repetition Time.

Glossary

Artificial Intelligence The ability of a computer to do typically human activities like thinking, making decisions or learning.

Artificial Neural Networks are computational models inspired by biological neural networks, and represented as a system of interconnected neurons that compute outputs from inputs, when information is fed through the network.

Autoencoder Unsupervised artificial neural network trained to efficiently encode input data, and reconstruct it from a representation using a decoder.

Blood Oxygenation Level Dependent Technique used to generate images in fMRI that correlates the signals among different brain regions through the time.

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

Brain Parcellation Subdivision of the brain structures into well-defined parcels or regions of meaningful anatomical or functional significance.

Connectome Network representation of whole brain connectivity, comprising grey matter and axonal connections.

Convolutional Neural Networks Specialized kind of neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

CPU Electronic circuitry within a computer that handles all instructions that it receives from hardware and software running on the computer.

Data Visualization Graphical representation of information and data.

Design Science Research Rigorous scientific research methodology that includes the techniques, principles, and procedures that must be followed to design and develop successful solutions to the problems.

Feedforward Neural Networks Artificial neural networks in which information flows in a single direction, from the input, through the net, to the output.

Functional Connectivity Temporal dependency of neuronal activation patterns in the brain regions.

GPU Programmable logic chip specialized to display functions.

Gray Matter Unmyelinated neurons and other cells of the nervous central system.

Machine Learning Area of Artificial Intelligence that enables computers to learn without explicit programming.

Medical Imaging Informatics Discipline that results from the combination of medical imaging and biomedical informatics.

Pattern Recognition Automated recognition of patterns and regularities in data.

Perceptron Building block of an artificial neural network, and inspired by biological neurons.

Recurrent Neural Networks Artificial neural networks with feedback connections in neurons.

Resting State Networks Brain activity observed through BOLD signals when a subject is not performing an explicit task.

Time-series Series of data points listed in time order.

Voxel Volume element representing a value on a regular grid in three-dimensional space.

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

Working Environment Set of software which enables writing programs for a particular language or platform.

1 Introduction

Summary

Medical Imaging is the field concerned with producing images of parts, tissues, or organs of the human body for medical proposes, and it has transformed the field of diagnostics, impacting the public health. As machines are getting biomedical data as faster rates, there has been a growing interest in reducing the human workload to eliminate human bias, and producing faster, reproducible results. Advances in Machine Learning, and more recently Deep Learning, have skyrocketed their popularity for improved accuracy and power to deal with high-dimensional data. Hence, the research community is starting to adopt these techniques as promising alternatives to the traditional statistical methods for analysing the brain's functional connectivity in hopes to find hidden patterns. Motivated by these advances, the main goal of this dissertation is designing and developing an experiment setup to analyse functional connectivity at a voxel level in order to find functional patterns, using resting state fMRI data.

1.1 Context

The human urge and pursuit for information have led to the development of increasingly complex technologies [1]. As Medicine learnt to exploit and incorporate technological knowledge and new techniques to meet its information needs, Informatics started to diffuse in Medicine, to the point where the concept of Medical Informatics emerged. Medical Informatics focuses on the adequate use and efficient management of Information Technology (IT) in health [2]. In other words, through the help of innovative IT solutions, the aim of Medical Informatics is to improve both patient care and opportunities in medical research, enabling the collaborative use of data by the health care system, and research community [3]. The Medical Informatics field is, thus, composed of individuals with diverse backgrounds and levels of training and it has been growing to be today one of the foundations for Medicine and health care [2, 4].

A popular area in the healthcare sector which has been advantaging from the improvements in the

Medical Informatics field is Medical Imaging. It refers to the processes used to produce images of parts, tissues or organs of the human body for medical proposes, to help on decision making and research [5]. By creating faster, less invasive, and more precise imaging, Medical Imaging has transformed the field of diagnostics, and significantly impacted the public health. Over the past decades, it has expanded from the former purpose of diagnosis to the treatment, and monitoring fields, and more recently, the exciting arena of disease prediction [5, 6].

In today's science, understanding the human brain is considered one of the most complex and challenging issues. Neuroimaging is a subarea of Medical Imaging and it addresses techniques designed to study the brain, either in terms of structure or function, although these two are inextricably intertwined [7, 8]. Structural neuroimaging provides static anatomical information of the brain, useful for evaluating injuries and disease diagnosis. On the other hand, functional neuroimaging provides dynamic physiological information, focusing on both cognitive and emotional processes of this organ [8, 9].

As known, the human brain is capable of processing and integrating large amounts of sensory information, and quickly responding to environment challenges [10]. These impressive cognitive competences, along with other mental faculties showing intelligent behaviour, are derived from the human brain's underlying architecture and functional organization [10]. This organ is divided into spatially distributed regions with their own task and function. Despite being anatomically separated, these brain regions are functionally linked and continuously sharing information with each other, forming a very complex network [11]. This temporal dependency of neuronal activation patterns in the brain regions is known as Functional Connectivity (FC). As it seems to be a steady characteristic of the brain, FC has been used as a trait measure in many studies [12]. Considering all the existing connections, and possible patterns yet waiting to be found, the knowledge brain neurons conceal is still overwhelming – which launches FC to an even higher level. Some aspects of functional connectivity are shared across people, as the brain's functional architecture has proven to be very similar across different individuals and different mental states, regardless of the population being studied or method used. In fact, changes in functional connectivity can be used as an early marker of some dementia-related diseases and more [13, 14]. FC is typically measured through functional Magnetic Resonance Imaging (fMRI), a non-invasive functional neuroimaging modality, with whole-brain coverage and high spatial resolution [11, 15]. As soon as data acquisition is complete, an interpretation process follows.

Interpreting medical images is a repetitive and time-consuming task, which involves the detection of abnormalities and quantification of measurements and changes over time. The interpreters of these images are physicians, who unwittingly limit the process with human subjectivity, big variations across

interpreters, and exhaustion [16]. Computers, on the other hand, have demonstrated expertise to master this "job", as they perform tirelessly and consistently a given task [17]. The acknowledgement regarding the potential of these machines has opened a path of new approaches and possibilities to the analysis of medical images.

The continuous search for alternatives reducing the human workload and producing faster, robuster and reproducible results has triggered a fast-growing interest in Machine Learning (ML), and more recently, Deep Learning (DL). Both are subsets of Artificial Intelligence (AI) that give computers the ability to learn from experience, and to deal with complicated concepts by building them out of simpler ones, without the help of a human operator [17, 18]. These traits have put ML – and, in particular, DL – in the leading position as the most exciting method in imaging and computer vision domains, enabling the extraction of meaningful patterns, in which some are even beyond human perception [16, 17].

A useful method for processing and identifying patterns on data is clustering. It involves grouping data points, based on their characteristics, into a number of groups (clusters), using dissimilarity or similarity metrics [19]. For instance, data points with a high level of similarity are assigned into the same cluster, to guarantee that these are more similar among them, than to data points in other clusters [11]. In Medical Imaging, hidden correlations can also be found used for prediction using image clustering. This process is identical to data clustering, except images are used and grouped, instead of data points [20].

After pattern recognition and data clustering, the resulting data must be analysed and interpreted, as it is not yet intuitive. Data Visualisation is the field that helps users understanding abstract data by creating 'images' from it, using algorithms [19]. Differences in colour, size, shape, movement, and proximity appeal the human perception system, and Data Visualization is based on this principle to creatively draw human attention to patterns and outliers. Since data is represented in a graphical or pictorial format, it is quickly processed and easily assimilated by the human brain, opposed to what happens when staring at massive spreadsheets of data [21]. Healthcare has been benefiting from AI's heavy use of Data Visualization, for performing a fast analysis on large amounts of complex data, highlighting the useful one, and thus, enabling to address problems more effectively [21, 22, 23]. Not only is this helpful for AI developers but also to healthcare professionals as both will be able to explain and understand how the system works, manipulates data and responds to it [22].

Combining ML and Data Visualization enhances the quest and odds to discover deeper connections in data, through a better visualization and improved predictive models. ML models are continuously learning from existing data and adapting to it, whereas Data Visualization is giving a better context for the resulting information, and together they engage on achieving levels of knowledge once unknown [24].

1.2 Motivation

Image acquisition and interpretation affect the quality of the information in medical images. Statistical methods are very popular for analysing functional data, and have been successfully applied in multiple studies [19]. Recent advances in biomedical research, specifically regarding imaging technologies, have contributed to the evolution of Medical Imaging, as acquisition devices are obtaining data at faster rates, and with increasing resolution [17]. With the upsurge in the volume of available biomedical data, and growing computational performance, ML and more recently DL have earned an active role in imaging problem solving as an alternative to the traditional statistical methods [18]. ML has demonstrated improved accuracy in multiple studies and proficiency in handling larger amounts of data, with greater dimensionality.

There is still a lot to discover and understand about the human brain. Neuroimaging has been continuously trying to fill the gap on the information regarding the functional brain, as some answers remain unknown due to its complexity. Although some ML techniques have already shortened the path to get explanations, the use of data is far from what can be achieved.

1.3 Objectives

The main purpose of this work is designing and developing an experiment setup for analysing functional connectivity at a voxel level, and find functional patterns, using resting state fMRI data. The tasks to reach this goal are the following:

i. Preprocess rsfMRI data to obtain a time-series per subject.
ii. Design ML models using artificial neural networks to reduce original high-dimensional data to a smaller representation.
iii. Explore methods for pattern recognition of FC data.

1.4 Research Methodology

Before developing any solution, a research strategy must be adopted. The methodology selected to guide this dissertation is Design Search Research (DSR), which brings both practical and theoretical rigor to research, and it is often used in computer science and medical informatics projects. DSR includes the principles, practices, and procedures that must be followed to design and develop a solution to the problem in hands [25]. This process includes six steps:

1. Problem identification and motivation – Specifying the problem, and the importance of a solution.

2. Definition of the objectives for a solution – Deducing the objectives of a solution based on the problem, and on what is feasible.
3. Design and development – Designing the solution (e.g. models or methods), its desired functionality and architecture, and creating the actual solution.
4. Demonstration – Using experiments, simulations or case studies to support the use of the solution to solve one or more instances of the problem.
5. Evaluation – Comparing the objectives of the solution to the results achieved in the demonstration step, to assess whether the proposed solution can solve the problem.
6. Communication – Disclosing the problem and its relevance to an appropriate audience, as well as the solution and its utility and effectiveness.

1.5 Search Strategy

The search strategy was able to explore the relevant material for this dissertation. Literature in the domain of fMRI brain connectivity focusing on pattern recognition including journal articles, books and proceedings, were considered key sources for this dissertation. During the exploration phase, references were identified using a list of academic databases and search engines such as Google Scholar, arXiv, PubMed, IEEE Xplore, SpringerLink, Frontiers, and Wiley Online Library. The following keywords were some of the used: “brain connectivity”, “functional connectivity”, “fMRI”, “pattern recognition”, “deep learning”, “machine learning”.

1.6 Dissertation Outline

Besides this introductory section, this dissertation is structured in four more sections. Section 2 covers the essentials on brain connectivity, the techniques used in Medical Imaging to measure neural activity, and how it provides means to detect mental disorders. Section 3 focuses on a technical background, introducing concepts related to machine learning, its models and their utility, and several tools are explored to set up a proper working environment. Section 4 describes the materials and methods for the proposed experiment setup, from the domain knowledge, to case study results and discussion. Section 5 resumes all final thoughts regarding the developed experiment setup and its results, and presents suggestions that could improve the work in the future.

2 Medical Background and Concepts

Summary

The human brain is a highly sophisticated organ, whose regions are structurally and functionally connected, forming a very complex network. Functional connectivity describes patterns of statistical dependence among distinct and distant brain regions. It is a steady characteristic of the brain, hence it has been used to understand brain's neural activity, and how it relates to human behaviour. Functional MRI currently leads the run for most precise and widely used imaging technique to study the functional brain. It is based on changes in blood flow as a response to neural activation of brain regions, and uses Blood Oxygenation Level Dependent contrast. During a resting state fMRI acquisition, the subject is awake and not thinking of anything in particular. This type of acquisition has increased the clinical utility of fMRI, as recent investigations have suggested that disturbances in functional connectivity at rest may be the foundation of several mental disorders.

2.1 The Human Brain

The human brain is a highly sophisticated organ forming the center of the nervous system, and it is composed of a mass of nerve tissue which can be divided into white matter (axons), and gray matter (neuronal cell bodies and dendrites) (figure 1a) [26, 27]. It has a left hemisphere and a right hemisphere, with a total of six smaller distinguishable areas, all with different functions (figure 1b).

The hemispheres are connected through the *corpus callosum*, a bulky bundle of nerve fibers allowing the continuous communication between both sides of the brain [28]. Together, they form the brain network, a complex "web" of structurally and functionally linked brain regions, which uninterruptedly process and transport information [11]. Although the arrangement of the anatomical structure conditions the brain network dynamic, it does not determine it, let alone explain it by itself [29]. To fully comprehend the dynamic nature of a complex system, one must not think of the system as composed of independent

elements, but rather as a system of interactions [30]. Large-scale neuronal communication in the human brain, and how it relates to human behaviour can be inferred by delving into the brain network, instead of separately studying brain regions comprising it [11, 30].

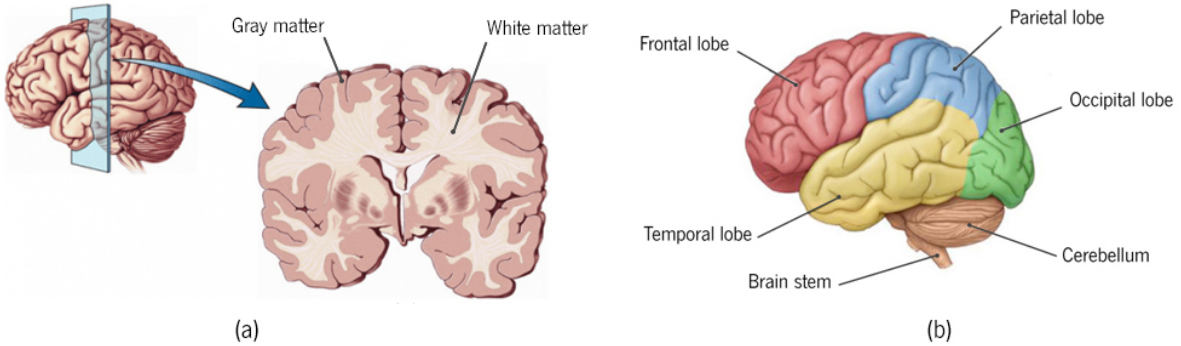


Figure 1: The anatomic human brain: (a) nerve tissue (coronal cut); (b) brain regions (lateral view). Adapted from [31].

Lang et al. [13] describe the concept of connectivity as "the strength of interactions between different brain areas which locally process information". Brain connectivity analysis is essentially carried by two main forms of connectivity: structural connectivity, and functional connectivity.

2.1.1 Structural Connectivity

Structural Connectivity (SC) – or anatomical connectivity – reflects anatomical connections linking the neuronal elements of the human brain, forming the human *connectome* (figure 2) [32]. These anatomical connections are in fact the physical synaptic connections between neighbouring neurons, or white matter fibers connecting neuronal pools[1] in spatially distant brain regions [13, 29]. On shorter time spans (seconds to minutes), SC is quite stable, while for longer time scales (hours to days), a substantial plasticity may be observed [32]. The most used technique to study SC is Diffusion-Weighted Magnetic Resonance Imaging (dMRI), a non-invasive method to infer the spatial orientations and trajectories of fiber tracts, and indirectly study the structure of white matter [29, 32, 34]. It maps the diffusive motion of free water molecules in the brain tissue along the preferred direction of white matter tracts [29]. In order to quantify SC, a set of three indirect measures are used: connection probability, fiber count, and fiber length [29, 32]. SC is, therefore, independent of the subject state [35].

[1] Functional groups of neurons that occur in the grey matter of the brain, and spinal cord [33].

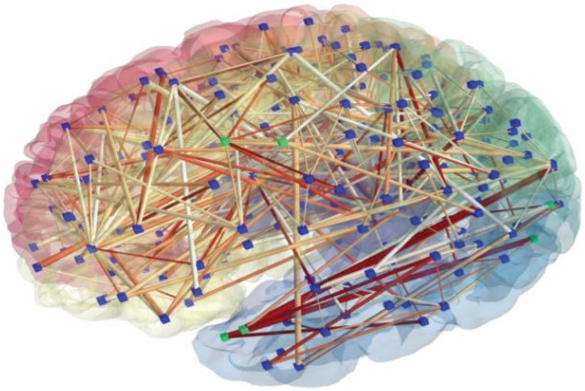


Figure 2: The human connectome. The hubs identify connections of brain regions, and the lines represent the signals among different regions. Adapted from [27].

2.1.2 Functional Connectivity

Functional connectivity (FC) describes patterns of statistical dependence among distinct and distant brain regions. In other words, it is based on the assumption that brain regions that are activated all together are related, even if anatomically separated [35]. FC is highly time-dependent, fluctuating at time spans of milliseconds to seconds, as it is continuously inflected by external events such as sensory stimuli and tasks [13, 32]. Thus, it depends of the subject state. Cross-correlation, co-variance, spectral coherence, and mutual information are measures used to estimate FC [13]. The techniques used to study FC will be mentioned and explained in subsection 2.2.

At low frequencies ($\nu \leq 0.1$ Hz), FC displays fluctuation coherence among distinct brain regions [13]. This phenomenon suggests that there is underlying neuronal activity in the human brain even at rest. Brain regions that are still active when the subject is awake and not focused on the external environment form sub-networks referred as resting state networks (RSNs) [36]. Default Mode Network (DMN) is the standard example of a RSN, as it is active during rest with a high degree of FC between brain areas, and deactivated while the subject is performing cognitive tasks [13, 27, 37]. DMN consists of a group of regions including medial prefrontal cortex (mPFC), precuneus/posterior cingulate cortex (PCC), and medial, lateral, and inferior parietal lobes [13, 34]. Although it does not reflect any cognitive function, DMN is involved in social cognitive processes and it is believed to be related to consciousness [27, 37]. Some RSNs replicate the patterns of activation observed in the brain when the subject is finishing a task, as a way of preparing the brain to external stimuli [37]. The visual and motor networks are two examples of RSNs involved in cognitive tasks. The counterpart to the DMN is known as the anticorrelated network (ACN): while performing a cognitive task, the ACN is activated, reorganizing itself in a task-related or goal-

oriented way. Thus, brain connectivity includes both task-negative and task-positive components in the resting state [13].

2.2 Measuring the Functional Human Brain

Nowadays, there is a diversity of functional neuroimaging techniques to study the human brain. To provide higher comfort to individuals during image acquisition, and to decrease their exposure to risk – e.g. developing injuries or diseases – these techniques were made to be noninvasive, or if not possible, as little invasive as possible. Besides the invasive-noninvasive category, these techniques can be assigned to essentially two types, depending on how the measurement is performed [38]:

- **Direct** – Techniques able to directly measure the neuronal activity in the brain, such as electroencephalogram (EEG), and Magnetoencephalogram (MEG).
- **Indirect** – Techniques that perform an indirect measure of the neuronal activity (based on changes in blood flow and metabolic activity), such as Positron Emission Tomography (PET), and functional Magnetic Resonance Imaging (fMRI).

2.2.1 Electroencephalogram

Electroencephalogram has been in existence since the 1920s, being the oldest functional brain imaging technique to use [39]. EEG records the electrical activity related with neuronal cells depolarization. These voltage fluctuations are oriented perpendicularly to the surface of the brain, and the equipment used to measure them is an array of electrodes placed over the scalp [38]. EEG provides real-time measurements of neuronal activity during different brain states, having a great temporal resolution [38, 39]. On the other hand, this technique has poor spatial resolution, meaning it has trouble identifying where in the brain the underlying signal is being produced.

2.2.2 Magnetoencephalogram

Magnetoencephalogram measures disturbance of the magnetic field produced by electrical activity related to neural depolarization. Opposed to what is observed in EEG, MEG records activity oriented parallel to the surface of the brain. This technique also does great on temporal resolution, and it has limited performance in, again, pinpointing the precise origin of the signal. Nevertheless, MEG has a better

spatial resolution than EEG, as magnetic fields are less distorted by the skull and scalp when compared to electrical fields. Although EEG tends to be more sensitive to activity in more brain areas, MEG allows electric activity to be visible and thus localized with greater accuracy. [38, 39]

Due to the differences in both direct measures of brain activity, researchers often measure EEG and MEG at the same time to take advantage of what each technique has to offer.

2.2.3 Positron Emission Tomography

Positron Emission Tomography uses radioactive isotopes to map cerebral blood flow (CBF) [40]. A small amount of radioactive tracer is injected into a vein, and to revert to a stable configuration, the radioisotope emits positrons. Seconds after the tracer entering the brain, radiation rises to its maximum value, and CBF is saved in a picture. Though $^{15}O_2$ is the most widely used radioisotope, researchers often synthesize in lab radiopharmaceutical compounds that bind to dopamine or serotonin receptors [38]. These receptors are known for operating in the brain to regulate humor, pain tolerance, and satisfaction – a gratification to the patient, and therefore an advantage. There are, however, some downsides to this technique. Performing a PET scan requires a PET camera, the injection, and a cyclotron to produce the radioisotopes. These are specialized and expensive medical devices, which greatly limits the availability of this technology at hospitals or medical centers [38, 39]. Most radiopharmaceuticals have a very short half-life, requiring the production of more compounds between exams, and thus a higher cost [40]. Another disadvantage to PET are its poor spatial and temporal resolutions. Also, PET cannot be proposed for repeated applications for being a small invasive technique, and thus it is not suitable for specific populations, e.g. children [40]. These drawbacks have put PET in the rear end of brain activity techniques.

2.2.4 Functional Magnetic Resonance Imaging

Functional Magnetic Resonance Imaging is a magnetic resonance imaging-based technique introduced in the early 1990s, and its use for studying the functional brain has expanded rapidly since then, currently leading the run for most precise and widely used technique for the matter [41, 42]. Reasons include being non-invasive and not known associated risks for correctly screened subjects, while providing an exquisite temporal resolution, and whole-brain coverage [11, 15, 38]. Besides, this technique can be performed repeatedly on the same individuals, both in health and in disease, at relatively low cost per scan, and it uses equipment widely available [7, 27, 38].

Whenever a brain region is stimulated, changes in neuronal activity of such region occur. These variations are reflected in the magnetic resonance images by a change in the appearance of the tissue of that specific brain region. Nevertheless, these changes are imperceptible to the eye when comparing just two images, and random noise or motion might also be the cause of variation, and not neuronal activity. Thus, images of the brain must be acquired repeatedly, as fast as possible, while interleaving between tasks or conditions, to check for variations related to brain activity. [41]

In order to properly understand fMRI, a few concepts on Magnetic Resonance Imaging (MRI) must be brought forward.

Basics of an MRI System

MRI systems include a superconductive magnet designed to provide a very strong and homogeneous magnetic field, meaning it has exactly the same magnitude and orientation everywhere of the imaging confined space. This uniformity of the magnetic field is obtained by using conductive wrapped wire in repeated loops along a surface of a cylinder (figure 3). The center of these coils of wire is the inner opening – or the *bore* – where the subject to be imaged is positioned. A typical clinical MRI system used in hospitals has a magnetic field from 1.5 up to 3 Tesla (T). Besides the main magnetic field, there are more two forms of magnetic fields needed for MRI: *gradient* fields, and *radio-frequency* (RF) fields. Gradient fields are used to get spatial information from the MR signal, as they change linearly with position, across the region to image. These fields are generated by three gradient coils (one for each of x, y, and z directions), and they are responsible for producing the typical loud sounds heard during MRI scanning. RF fields are magnetic fields that oscillate at the radio-frequency range (approximately 63 MHz to 300 MHz), when swiftly changing the direction of the electrical current passing smaller coils of wire. [41, 43]

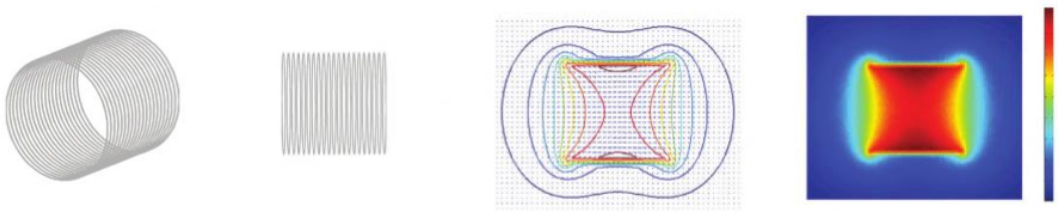


Figure 3: Magnetic field created by passing an electrical current through a continuous conducting wire wrapped on a cylindrical shape. Adapted from [41].

The hydrogen nucleus is composed of a single proton, which has a *magnetic moment*, and due to that property it behaves similarly to a compass needle exposed to the earth's magnetic field [43]. The human body is approximately 70 percent water, and water molecules, along with many biological compounds, have hydrogen atoms in their composition. When a subject is being scanned, a number of protons will align parallel to the main magnetic field, and RF fields emit a RF pulse to disturb the protons' alignment, and making their magnetic moments perpendicular to the magnetic field. The cumulative effect of all magnetic moments of hydrogen nuclei is a net magnetization vector, and when applying a RF pulse, two magnetization vector components are produced: longitudinal magnetization, and transverse magnetization [44]. After the RF pulse stops, the net magnetization vector begins to align back to the direction of the main magnetic field, and protons produce RF energy in the surrounding electrical circuit, which is detected by coils. This regrow rate is known as *relaxation time*, and there are essentially two types: T1, and T2. During T1 relaxation the longitudinal magnetization recovers, as the protons release energy (i.e. time to protons realign with the external magnetic field). During T2 relaxation the transverse magnetization decreases, due to the interaction of the individual magnetic moments of protons (i.e. time to protons lose phase coherence among nuclei spinning perpendicular to the main magnetic field) [44].

Each RF signal is assigned to its location with the help of gradient fields. The strength of this signal depends on the number of protons of the respective tissue, which also gives information on the amount of water that location contains [43, 45].

Image contrast is derived from differences in T1, and T2 relaxation times, and proton density in tissues. Repetition time (TR), and echo time (TE) play a decisive role, as these parameters detect the difference between tissues at separate levels. TR is the time between two RF pulses, and TE is the time between a RF pulse and the peak of its corresponding echo (figure 4). Disparity in T1 recovery between water and fat can be detected with short TRs, but not with long TRs. Variation in signal decay between water and fat can be identified with long TEs, but not with short TEs. For a long TR and a short TE, differences in T1 recovery and in T2 decay between water and fat are not noticeable. Thus, there are four main points to retain: (i) TR relates to T1, and affects contrast on T1-weighted images, (ii) TE relates to T2, and affects contrast on T2-weighted images, and (iii) the contrast in MR images is mostly due to dissimilarity in proton density between two tissue types; (iv) tissues with more protons have higher signal intensity, as opposed to those with fewer protons, which have lower signal intensity. [44]

The most common MRI sequences are T1-weighted, T2-weighted, and Fluid Attenuated Inversion Recovery (FLAIR). A T1-weighted sequence is produced by a short TR, and a short TE, best representing brain anatomy. A T2-weighted sequence is produced by a long TR and a long TE, best depicting brain

diseases. A higher water content in tissues is typical of pathological conditions, and affected areas appear bright on T2-weighted images. FLAIR is identical to a T2-weighted image, but with very long TR, and TE. This sequence is even more sensitive to diseases, and useful to differentiate between the cerebrospinal fluid (CSF), and abnormalities in brain tissues. [44, 46] The comparison between the three sequences is at table 1.

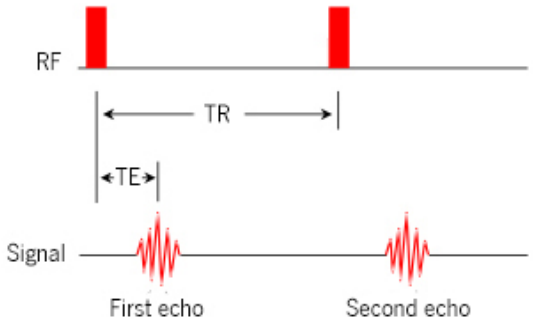


Figure 4: Schematic representation of repetition time (TR), and echo time (TE). Adapted from [47].

Table 1: Differences between T1-weighted, T2-weighted, and FLAIR sequences [46].

Sequence	TR (mse)	TE (mse)	Fat	Cortex	White Matter	CSF	Demyelination
T1	500	14	Bright	Gray	Light	Dark	Dark
T2	4000	90	Light	Light Gray	Dark Gray	Bright	Bright
FLAIR	9000	114	Light	Light Gray	Dark Gray	Dark	Bright

BOLD Contrast

The basic principle of fMRI is that when a region of the brain is activated, it triggers a locally increased energy requirement, and as a response, there is an increase in blood flow, and changes in oxygen concentration in the affected brain region [48]. These two phenomenons are indirect measures of brain activity, as they are based on haemodynamic changes induced by regional brain variations in neuronal activity, and both can be detected using fMRI [49]. In the first mechanism, the change in CBF can be observed invasively by injecting contrast agent, or noninvasively by arterial spin labeling (ASL) [48]. The second mecanism – and the most common one – uses Blood Oxygenation Level Dependent (BOLD) contrast, which detects fluctuations on the blood oxygenation through magnetic properties on haemoglobin [50]. Haemoglobin is the fundamental oxygen-carrying molecule in the blood, and its the

center, there is an iron atom with magnetic properties [39]. When it is fully saturated with oxygen – oxyhaemoglobin (HbO_2) – it is diamagnetic, and when some oxygen atoms are removed – deoxyhaemoglobin (Hb) – it becomes paramagnetic. The presence of deoxyhaemoglobin causes a distortion on the magnetic field surrounding the blood cells, leading to an increase of the local heterogeneity [50]. A MRI scanner, as seen before, is itself a very strong magnet, and it can detect this small difference in the magnetic property of blood. There is a slower decay of the MR signal for brain areas with high concentration of oxyhaemoglobin, which results in higher intensities of T2-weighted images [27].

The behaviour of a BOLD signal obtained from a single concise activation event is described by the hemodynamic response function (HRF) (figure 5). Usually, the BOLD signal peaks approximately 5 seconds after activity onset, decreasing afterwards towards the baseline, and undershooting it around 8 to 12 seconds after onset before the signal stabilization about 20 seconds after the activation event. The initial delay observed is linked to the slow nature of blood flow. [51]

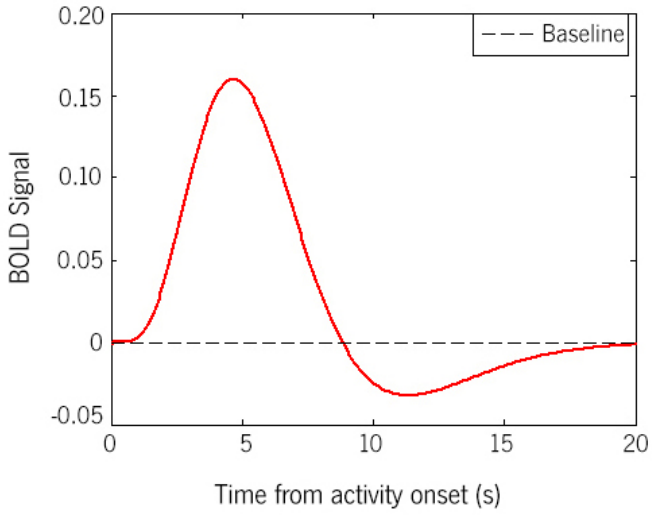


Figure 5: Hemodynamic response function for a single short activation event. Adapted from [51].

The BOLD signal is of low intensity, and therefore must be compensated with repeated acquisition [48]. Also, variations of the BOLD response are very dynamic, hence to study them, several brain volumes must be acquired throughout the time, and with a very short interval in-between [52]. Echo-Planar Imaging (EPI) is a fast imaging technique which can collect all data to reconstruct an entire MR image within a fraction of a second [53, 54]. It allows acquisition of images in 20 to 100 milliseconds, virtually eliminating motion-related artifacts, and a great multisection capability. Single-shot EPI, and multishot EPI are two variations of this technique, in which the first uses single excitation pulse – or shots –, and the second uses multiple excitation pulses. Single-shot EPI provides all the spatial-encoding data of an image after a single RF

excitation. Multishot EPI provides high-quality images with resolution and contrast similar to conventional MR images, in just a matter of seconds. [54]

Representing Images with Numbers and Vice-Versa

During a scanning session, the brain is fragmented into minuscule cube-shaped spaces named *voxels*, each containing tens to hundreds of thousands of neurons [39]. A voxel is the minimum spatial resolution unit, with dimensions varying from 0.5 to 3 millimeters per side, that encloses a signal whose frequency is indicative of the average density of the hydrogen protons within the volume element [39, 55]. Although a voxel by itself does not much provide meaningful information, comparing it to voxels of other brains allows to recognize and establish patterns across different individuals [56]. Brain tissues (gray matter, white matter, and CSF) have each a distinct number of hydrogen nuclei, and after a RF pulse excitation, they transmit a RF signal in form of energy. The different signals are converted from the frequency-space to an image-space using an inverse Fourier transformation, to produce a map of the brain with intensity levels (figure 6) [55].

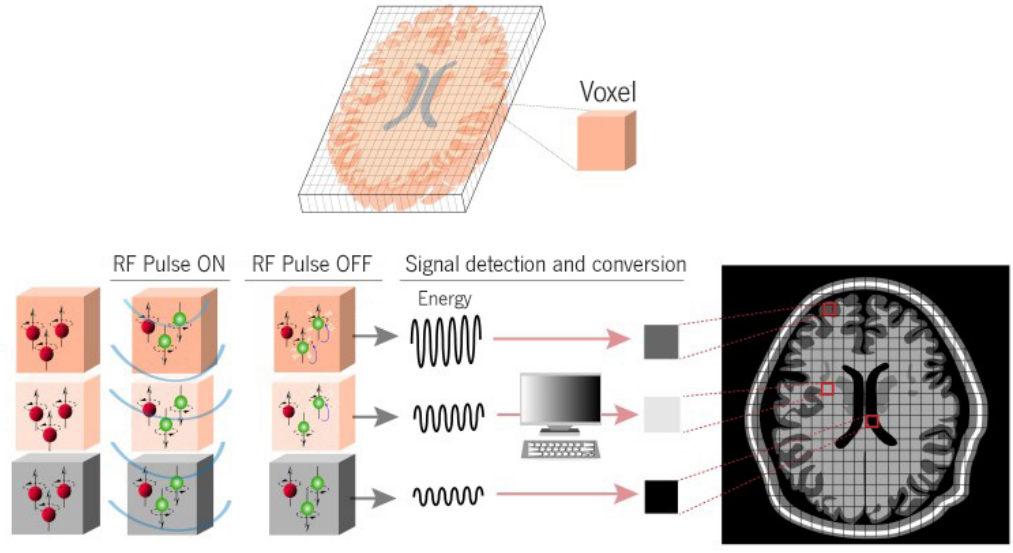


Figure 6: After a RF pulse, brain tissues produce different signals, which are detected and converted to produce a map of the brain. Adapted from [56].

Displaying MR signals in a 2D plane resembles a picture of the brain's anatomy when in fact it is just a representation, which can depict inaccuracies due to spatial distortions or influence of artifacts [41]. A whole-brain clinical MRI scan is usually a stack of 2D slices (i.e. axial brain cuts) at a given moment [39]. Each slice is a grid of *pixels* assigned to a shade of gray [41]. The lighter or darker shades correspond to the strength of the MR signal for that brain unit, i.e. higher or lower BOLD activations, respectively [39].

All pixels in an MR image individually represent one or more numbers, and these values can be plotted as a function of any component of spatial position, or as a function of time [41]. For instance, each voxel (or stack of pixels in the same x and y coordinates over the z axis) can be represented by its time-series denoting the activity level or activation in that brain area across time, for a single subject [55].

Preprocessing Pipeline

As result of fMRI indirect detection of neural activity, acquired data is unavoidably confounded by sources of variation, e.g. subject's head motion, physiologic processes of respiration, cardiovascular functions, and magnetic field inhomogeneity [42, 48]. If not corrected, these undesired fluctuations may conceal intrinsic patterns of neural activity, reduce the detection power of further analysis, or even alter experimental conclusions [42]. As the primary goal of any fMRI acquisition is to obtain the highest possible signal-to-noise ratio (SNR), several computational procedures have been proposed to remove such noise from acquired data [57]. The typical procedures are detailed below, and they are collectively termed as the preprocessing pipeline.

Quality Assurance

This is the first and an indispensable step for a well carried out preprocessing pipeline [58]. It is based on the principle that raw imaging data must be examined to prevent corrupted data from propagating artifacts to the results [42, 58]. The first quality control occurs during acquisition to prevent losing data. This approach includes checking the appearance of the brain being imaged, and detecting blatant head motion or hardware issues, while repeating the acquisition is still feasible [57]. The second quality control is to assure the acquisition protocol has been respected for all participants recruited for the fMRI study, and confirming all images have been accurately imported and sorted according to, for example, the subject, and the session number [57]. To ease the inspection of scans which hold a considerable number of slices, viewing tools such as Osirix, and ImageJ are often used, as they enable imaging dynamic display helpful for detecting visible artifacts [57, 58].

Data Conversion

Clinical imaging data is stored, and transferred across computer networks from hospitals or imaging labs in DICOM format, which stands for Digital Imaging and Communication in Medicine [41, 52, 59]. Although most MRI systems output data in DICOM format, fMRI preprocessing tools use mostly NIfTI (Neuroimaging Informatics Technology Initiative) file format [41, 57]. There is not a standard file format, hence data conversion from DICOM to NIfTI is a fundamental step for futher fMRI data analysis. The

information about the imaged subject contained in the DICOM header is discarded after the conversion, keeping only basic acquisition information, such as TR, TE, FoV and other imaging details [57].

Initial Stabilization

When starting an acquisition, the scanner takes some seconds to fully stabilize its magnetic fields, and so do tissues to achieve the level of excitation required in order to be properly imaged [57]. For that reason, it is recommended to discard the first few scans, typically around the 10 initial seconds, as they may have corrupted information [52, 60].

Slice-Timing Correction

Single brain volumes are assumed to represent data at the same point in time, meaning the BOLD signal of a whole volume is expected to be relative to a specific moment [52, 60]. Most fMRI studies use EPI during acquisition, which causes slices of each volume to be collected sequentially (either in a consecutive or interleaved way), rather than instantaneously [42, 61]. This leads to an accumulation of delays between the first and all remaining slices, creating a lag between real and expected slice acquisition times, even for a fixed and short TR [57, 61]. Slice-timing inconsistencies are commonly corrected using temporal interpolation: the time-course of voxel data in each slice is shifted to temporally align all slices to a reference time-point (typically the mean TR slice), and the amplitude is estimated by interpolating information between the neighbouring voxels within the TR [42, 57, 61].

Head Motion Correction

Head motion during acquisition is a prominent concern in most fMRI studies, for not being conceivable to completely eliminate it, especially when hour-long duration scans are performed or specific populations are being imaged (i.e. young, elderly, and diseased subjects) [42, 58]. It can affect the quality of collected data, as motion mixes signals from neighbouring voxels, and interacts with field inhomogeneity, and slice excitation [42]. To perform a proper data analysis, there must be precise spatial correspondence between voxels, and anatomical areas over time [57]. Strategies to diminish head movement include head-immobilization techniques with fixation devices (e.g. masks, or bite bars) during acquisition, or using a MRI simulator to acquaint the subject to the scanning environment [42, 58]. Motion correction is post-acquisition, and typically performed using a rigid body transformation (x, y, and z rotations and translations) to realign each volume to a reference volume (often the mean volume) [52, 57].

Skull Stripping

A MR image of the brain depicts its both interior and exterior structures, with a high degree of anatomical detail [62]. To study the functional brain, one must isolate the cerebral tissue from non-brain tissues by removing the latter. This process is known as skull stripping or masking, and it helps reducing data size, and improving posterior normalization [52, 57].

Co-registration

Variation between modalities, and differences in MR contrasts (T1-weighted, and T2-weighted) and acquisitions (e.g. slice orientation, image distortion, and resolution) induce clashing between anatomical and functional images of a subject, even if they are collected during the same scanning session [42, 58]. Co-registration is a spatial transformation, in which functional images of an individual are aligned with their anatomical images to map the functional information to the corresponding anatomical space [42, 60]. The alignments can be performed by applying rigid transformations (rotations, and translations), and non-rigid transformations (zooms, and shears) [52, 60]. Optimization functions aim minimizing the discrepancy between the two images [57]. For having higher resolution and being less affected by distortion after interpolation, the MRI is typically resampled into the fMRI spatial resolution through an affine (i.e. linear) transformation, obtaining the co-registration matrix [52, 58]. Following, MRI is warped to a MRI-template using a non-linear transformation, generating the normalization matrix [52, 58]. The resulting composite image is then registered to the reference space [57].

Spatial Normalization

To study a group of individuals and search for functional correlations, one must reduce intersubject variability regarding the shape and size of their brains [42]. The process of aligning images from different brains to a common space is known as spatial normalization [57]. There are two main standard, and yet distinct, coordinate systems, as coordinates in each case do not correspond to the same brain regions or structures, often requiring conversion [57]. The Talairach space is based on a single older brain, and the principal axis corresponds to the Anterior Commissure-Posterior Commissure (AC-PC) line, which is used a reference plane for axial imaging [57, 58]. On the other hand, the Montreal Neurological Institute (MNI) templates are more representative of the population, as they are based on the average of T1-weighted MRI scans of a large number of healthy young subjects [57, 60]. Although MNI developed several templates, MNI152 is the most frequently used [57]. Voxels located in images of different individuals, when normalized using the same template, and defined by the same coordinates, will correspond to the same anatomical structure [52].

Spatial Smoothing

Adjacent brain areas often exhibit functional similarity, meaning there is inherent spatial correlation among those areas [42]. Sudden changes in the BOLD signal for locations of nearby anatomical structures may indict the presence of noise sources. Spatial smoothing enables suppressing the noise, and improving the signal-to-noise ratio (SNR) of functional data, by distributing the intensity of each voxel to its correlated neighbours. This procedure is implemented by convolving the fMRI signal with a Gaussian function of a specific width, whose extent is defined in terms of Full Width at Half Maximum (FWHM) in the Gaussian kernel [57, 61]. Spatial smoothing also contributes to decrease structural and functional intersubject variations [60]. Yet, there should be balance regarding the amount of smoothing to apply, considering it may reduce the resolution of the functional image and the accuracy of slices with small delays and errors, which is undesirable [57, 61, 63].

Temporal Filtering

Most fMRI time-courses are contaminated by low frequencies noise from multiple sources, such as slow scanner drifts ($\sim$ $<$0.01 Hz), and cardiac ($\sim$ 0.15 Hz), and respiratory ($\sim$ 0.3 Hz) phenomena [57, 58]. The final step to the preprocessing pipeline is temporal filtering, which helps to attenuate noise with known or expected frequencies, in order to increase the SNR [57]. Intrinsic neural activity of the brain – the main interest of rsfMRI – is reflected in spontaneous fluctuations of the BOLD signal that occur at frequencies lower than 0.1 Hz [57]. The standard approach for removing equipment-related and physiological noise from raw rsfMRI data is applying a band-pass filter with a 0.01 to 0.08 Hz bandwidth, to restrain the preserved signal to that frequency range [57].

Task and Resting-State fMRI

There are two main types of fMRI experiments: task-related (tfMRI), and resting-state (rsfMRI).

During a tfMRI experiment, sensory stimuli are employed to cue the subject perform a task, while data is being acquired. This temporal allocation of stimuli is a paradigm, and it is used to induce an hemodynamic response or brain activation in the subject [64]. There are two main design categories: block design, and event-related design (figure 7). The block designs use interleaved and equitemporal blocks of activity and rest to elucidate the actual response. In an event-related design, the trials are presented randomly and a specific cognitive event is in focus. As a result, different functional areas of the brain will be represented with more intensity in the magnetic resonance image, suggesting that those areas are functionally related to the performed task [65].

Throughout a rsfMRI experiment, the subject is instructed to remain still and not to perform any intentional activity, while keeping eyes either open or closed for the scan duration [42]. This provides means to get closer with the intrinsic brain network, as it has been confirmed that, during rest, there is information processing and functional connectivity in the brain. This unintentional activity is known to be the spontaneous low frequency fluctuations of the BOLD signal registered in rsfMRI studies [11, 27].

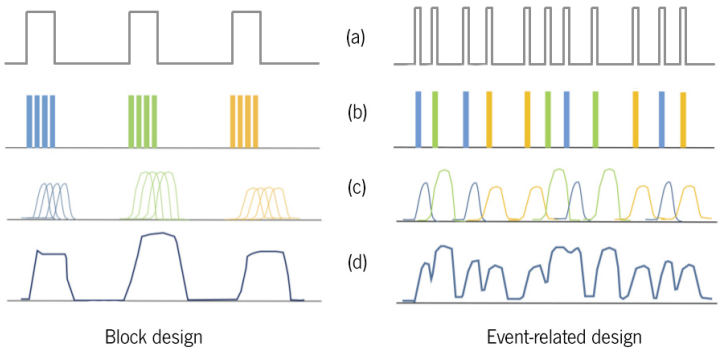


Figure 7: Differences between block design, and event-related design. (a) Paradigm design; (b) Stimulus event; (c) Hemodynamic response to for each stimulus; (d) Sum of all hemodynamic responses. Adapted from [66].

Analysis of Resting State Data

Several statistical methods have been proposed to analyse rsfMRI data, in order to explore the existence and extent of functional connections between brain regions [11]. These methods can broadly be classified into two categories: model-based methods (which require *a priori* knowledge about spatial and temporal patterns of activation, and a model for data generation), and data-driven methods (which do not require previous information nor predefined models) [13, 57, 67, 68].

Model-based methods

Model-based methods select a predefined brain region of interest (ROI) typically known as *seed*, and determine whether other brain regions are functionally connected to the seed, according to metrics established in advance [13, 67]. Correlation is a often used metric, since it is responsive to the shape of HRF, known to vary between brain regions, and across different subjects [13]. Cross-Correlation Analysis (CCA) is a technique that computes the correlation between the BOLD signal time-series of the seed, and all other voxels', resulting in a functional connectivity map (fcMap) [11, 13]. Yet, CCA's performance is questionable, considering noise, such as cardiac or blood vessel activity in the brain, may create an

illusion of strong correlations between regions without blood flow fluctuations [67]. Coherence is the alternative metric, as it is the spectral representation of correlation in the frequency domain. Blood flow fluctuations occur on a time scale of ten seconds, whereas cardiac activity works at a frequency of around 1.25 Hz. Coherence analysis is the technique that allows to distinguish noise from actual functional connectivity, being the spectral interest of 0.1 Hz or lower [13, 67].

Model-based methods are widely used in the detection of functional connectivity due to their simplicity to implement and interpret [29, 67]. The results, nevertheless, depend on the selected seed, as different seeds lead to different fcMaps, limiting the analysis to a restricted set of regions [13, 14]. Moreover, requiring *a priori* knowledge constrains the exploration of functional connectivity to a level, as certain brain regions may be unconsidered if not related to the prior knowledge [67].

Data-driven methods

Data-driven methods arise as an alternative to model-based methods, as the first aim examining functional connectivity patterns on a whole-brain scale, and may unveil unexpected correlations in data [11, 13]. Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Clustering are the three primary techniques, within data-driven methods, to analyse functional connectivity [29, 57].

Both PCA and ICA are often categorized as decomposition techniques, as they ought to express data using a linear combination of spatial and temporal components [14, 67]. PCA uses maximal variance as criteria, whereas ICA searches for statistical independence within components [27]. In PCA, the direction of maximal variance (the principal direction) is collected into the first principal component, and successively less conspicuous directions of variance are saved in subsequent components. These must be orthogonal and altogether they form the principal components [14]. In other words, PCA focuses on finding a set of orthogonal axes that best explain the variability of data, and vectors with small contribution to data variance are often discarded [67]. This allows to separate the relevant information from the noise, obtaining a more refined signal data [57]. For this reason, PCA is frequently used as a dimensionality reduction technique [13, 67]. There are some limitations regarding PCA though, as it is susceptible to sources of signal variation (e.g. physiological noise), and the orthogonality constraint lessens the usefulness and immediate interpretation of information extracted [13]. ICA is an extension of PCA, and the most frequently used statistical method for analysing resting state functional connectivity data [11, 57]. This technique separates linearly mixed signals into their statistically independent sources [57, 69]. Although ICA can be implemented to find either spatially or temporally independent components, ICA is used to decompose resting state data into a set of spatial patterns (maps), and corresponding time-courses [57]. Selecting components of interest is typically performed by correlation

with a predefined RSN or visual inspection [57]. This may be a challenge, considering independent components often contain a more complex representation of data when compared to fcMaps, and for that reason they are more tough to understand, which is an disadvantage to using ICA [11].

Clustering consists of partitioning data into different clusters based on their similarity, using a distance metric [13, 67]. This method enables the grouping of brain voxels with similar connectivity (i.e. similar BOLD time courses) in the same cluster [57, 70]. The top three most used clustering algorithms are k-means, fuzzy clustering (c-means), and hierarchical clustering [70]. k-means divides data into k different clusters, generally using Euclidean distance as metric [18]. This algorithm requires the user to choose beforehand k initial centroids[2] different from each other, and it works by alternating between two steps until convergence [18, 72]. The first step is allocating each data point to the nearest centroid, and the second step is updating the centroid by calculating the average of all data points assigned to that centroid [18, 72]. Each collection of data points assigned to a centroid is basically a cluster, and each observation belongs to a single cluster [72]. Fuzzy c-means is an algorithm which was introduced as an alternative to k-means, as it considers each voxel a member of every cluster with a variable degree of "membership", instead of assigning each voxel to a single cluster [72]. The membership values describe the probability of a voxel belonging to a cluster, ranging from 0 to 1, where 0 means the voxel does not belong to such cluster, and 1 means the voxel belongs to such cluster [67]. The sum of membership values for an individual voxel across all clusters equals 1, which allows such values to be partial [72]. The main idea behind fuzzy c-means is to minimize the function where membership values reside – membership function –, usually defined as the total distance between all patterns and their cluster centres [67]. Cross-correlation, for instance, is a distance metric which may be used in fuzzy c-means clustering to calculate the degree of correlation between two time-courses. For distances between a certain threshold, the corresponding brain regions are considered functionally connected [67]. Nonetheless, this metric is not any close to ideal, considering data might be contaminated with physiological noise. Similarly to k-means, fuzzy c-means also demands setting the number of clusters before initialization, and there is no conviction about how many clusters to chose [13]. Hierarchical clustering begins by considering each voxel as one cluster, and then melds "close" clusters using as distance measurement, for example, a combination of correlation analysis and frequency [67]. The implementation of such approach enables overcoming noise interference, and extracting information reflecting synchronization in CBF and oxygenation between different brain regions [67]. A major downside to hierarchical clustering is being computationally expensive, especially when applied to 3D human brain data, thus limiting its implementation in real world problems regarding medical

[2] Real or imaginary location representing the center of the cluster [71].

imaging [67].

The preference towards model-based methods or data-driven methods highly depends on the data to be studied and the experiment, as none outperforms the other in an all-around way [67]. All referred methods have provided results supporting the notion of the formation of multiple functionally linked networks in the human brain during resting-state [11].

Figure 8 illustrates the mentioned methods for analysing resting state data.

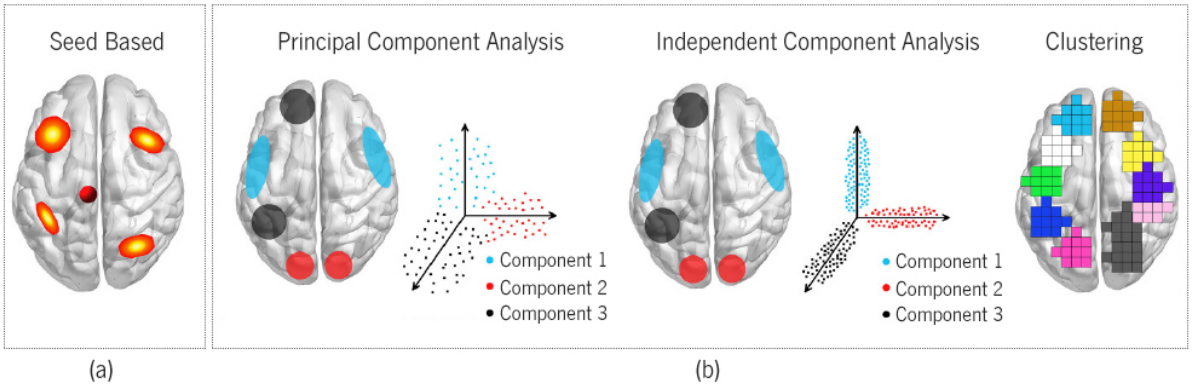


Figure 8: Commonly used analysis methods for resting state fMRI data: (a) model-based method; (b) data-driven methods. Adapted from [57].

Analysis of resting state data often requires brain parcellation, which consists of dividing the brain's spatial domain into a set of non-overlapping regions [73]. Each region comprises a group of voxels displaying homogeneous behaviour or properties observed in one or multiple imaging modalities. Strategies to study function at a region level include the previously mentioned seed analysis, and also the use of a brain atlas, or the use of data-driven parcellations [73]. Brain atlases have a well-defined set of ROIs covering whole-brain volume, representing a labeling of brain structures. There are assorted atlases online, including the Automated Anatomic Labelling (AAL) atlas, or the Harvard-Oxford atlas. Data-driven parcellations provide better representations as it adjusts to data and therefore can maximize the resulting signals, unlike predefined atlases [73]. Brain parcellation helps understanding brain's functional organization, and additionally it is used as a popular data reduction technique for statistical analysis [74].

Graph Theory

Graph theory is based on network sciences, and it has been recently adapted to explore functional connectivity in the human brain, providing unique insights into its topological organization [75]. This technique models the brain as an integrative complex network comprised of n nodes and k edges, with a

high level of spatial detail [11, 75]. The nodes represent either voxels or brain regions based on predefined anatomical atlases, such as the AAL or the Harvard-Oxford atlas [11, 57]. The edges depict the functional connection between the nodes (or regions), which is computed as the correlation or coherence between the time-series of both regions connected [11, 75]. Graph theory is almost an extension of the seed-based analysis where all available seeds are explored [57]. Establishing functional interactions between every possible brain region forms the *functional connectome*, which can be represented as a connectivity matrix or a graph [36, 57]. A connectivity matrix is a two-dimensional matrix representing all possible pairwise connections between neural elements of the brain [36]. For a brain network with n nodes, the matrix is composed by n rows and n columns. Each row and each column corresponds to a different node, and the matrix element positioned in the intersection encodes the connectivity value between those two nodes, and its functional coherence [36, 76]. For instance, diagonal elements of the matrix refer to the connection between a node with itself, and therefore those elements are conventionally set to a common value [36]. A graph can be generated from a connectivity matrix, and the typical stages to yield a graph theoretical representation of functional connectivity are illustrated in figure 9 [36, 75].

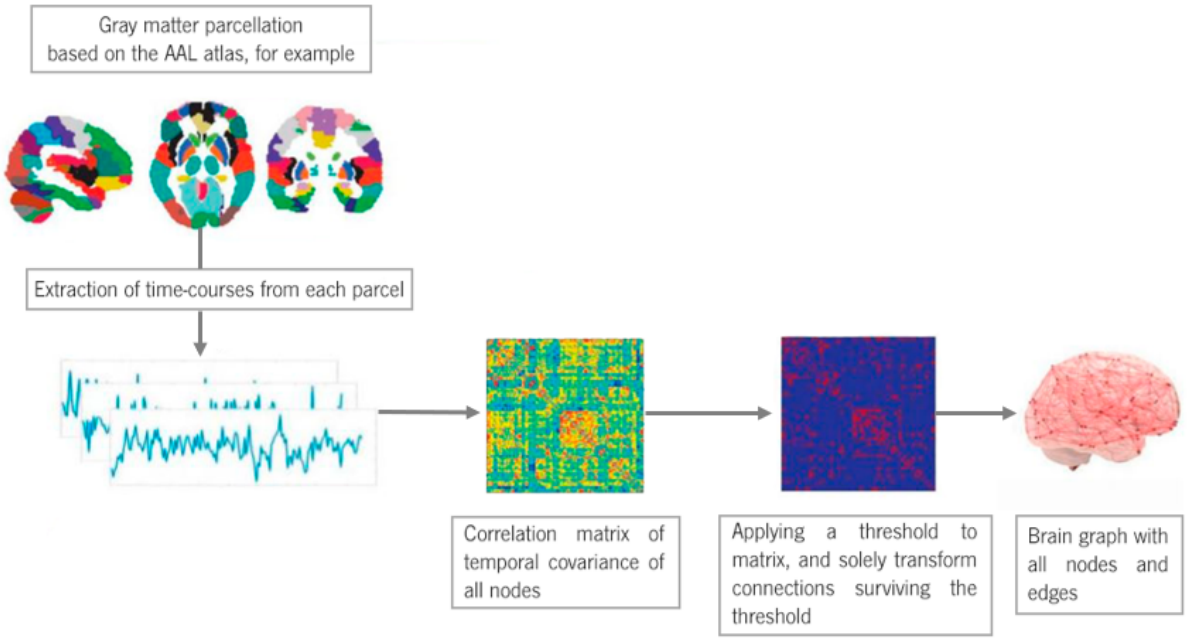


Figure 9: Graph theory analysis of functional connectivity, using an AAL atlas for parcellation, and cross-correlation between time courses. Adapted from [75].

2.3 Brain Connectivity as a Biomarker

A considerable and increasing portion of the population is affected by mental disorders [35]. Neurological diseases involve damage or malfunction of the nervous system (e.g. Alzheimer's disease, Parkinson's disease, and multiple sclerosis). Psychiatric disorders are identified by behaviour and emotion state disturbance (e.g. schizophrenia). Both types affect the functional balance of the human brain, altering brain connectivity between particular brain regions [35].

Recent investigations regarding FC have been focusing on studying the impact of mental disorders on the connections of the brain [35]. In fact, at rest FC has become one of the most important research fields of neuroimaging of the past decade [68]. The clinical utility of fMRI has been greatly increased, since resting state fMRI captures a wide variety of intrinsic mental states, with potential to generate insights about the nature of brain disorders [74].

The detection of functional changes is of great interest for research and medical communities, as it provides means for early diagnosis, and a way to keep up with the evolution of the disease. Besides, by acknowledging the nature and the extent of those functional changes, the treatment will be more appropriated [40].

2.3.1 Alzheimer's Disease

Alzheimer's Disease (AD) is a progressive neurodegenerative disease, and the most common one among patients older than 65 years, affecting nearly 50 million worldwide [77]. AD represents about 70% of dementia cases observed on people aged 60 years and older, thus being the most common form of dementia as well [78]. With an aging society and increased life expectancy, this disease is a major threat and challenge [78]. The typical brain change associated with AD is atrophy, and major symptoms derive from cognitive decline, including episodic memory loss, impaired communication, disorientation, confusion with time or space, and poor judgment [79].

There has been rather consistency regarding findings in AD patients during rest. The hippocampus is a brain region highly affected in AD, and multiple studies have focused on it [78]. For instance, Greicius et al., Sohn et al., and Tahmasian et al., all reported reduced FC between the hippocampus and the PCC in AD patients [80, 81, 82]. The DMN has seen a considerable amount of attention as well. Zhang et al. observed enhanced disconnection of the DMN with the progression of AD, and Demoiseaux et al. found decreased FC in the DMN of all patients, compared to controls [77, 83]. Gili et al. reported functional disconnection between critical areas of the DMN in AD patients, and also in amnestic Mild Cognitive

Impairment (aMCI) patients (i.e. at risk of AD), which itself supports the hypothesis that FC decreases in stages prior to AD [84]. Based on these and other reports of altered connectivity in regions wired by this network, it is now accepted that FC of the DMN is impaired in this neurodegenerative disease [78].

The progression of cognitive impairments in population at risk of AD can be prevented, since there is functional disconnection before onset of AD, and it can be revealed through rsfMRI experiments [85].

2.3.2 Parkinson's Disease

Parkinson's Disease (PD) is the most second common neurodegenerative disorder, with a yearly incidence of 8 to 18 per 100000 individuals, and a prevalence around 1% in people aged 60 and above [78, 86]. It manifests mostly with motor symptoms, such as akinesia, resting tremor, rigidity, and postural instability, but it is also commonly described by cognitive impairment and dementia [78, 87].

Abnormal brain activity in PD is not strictly linked to the performance of self-initiated movements, as recent studies have reported changes in functional connectivity before movement, i.e. during resting-state [88]. For instance, regions associated with the sensorimotor network, including the supplementary motor area, the premotor area, and the primary motor cortex seem to be affected [78, 88]. Although disturbance of connectivity has been exhibited, there is no consensus regarding the direction of such changes (i.e. increase or decrease). As for the DMN, no clear patterns were observed in the literature either. Krajcovicova et al. reported no differences between PD patients and controls, regarding the DMN [89]. Contrarily, Campbell et al. observed increased connectivity in the DMN of PD patients, and yet a decreased FC in their sensorimotor network [90]. Moreover, Putcha et al. found in PD patients an additional connection from the DMN to the right central executive network, not observed in healthy subjects [91]. Studies using graph theory revealed a decrease in efficiency of the brain network [78].

Changes in FC in the long term may be essential for a development of a biomarker for PD. Dubbelink et al. performed a study over a period of 3 years, and demonstrated for PD patients a progressive loss of FC for multiple brain regions, especially sensorimotor regions, and the occipital lobe [92]. This finding supports the role of reduced FC in cognitive decline and the development of dementia in PD [92].

2.3.3 Multiple Sclerosis

Multiple Sclerosis (MS) is a chronic, progressive disease of the central nervous system, leading to physical, mental, or psychiatric problems [85, 93]. MS is identified by the production of widespread lesions in the brain and spinal cord that cause demyelination, and thus inhibition of axonal transmission to

and from the brain [93]. Common symptoms include loss of coordination and balance, weakness, mood changes, visual impairment, and cognitive impairment [94]. More than 2 million people are estimated to live with MS worldwide, and the average age of diagnosis is approximately 30 years [94, 95].

The application of functional imaging to study the relations of brain and behaviour is a rather recent area of research in MS, and it has contributed with new and interesting knowledge so far [93]. For instance, Faivre et al. found increased FC in seven of eight RSN in early MS patients compared to controls, suggesting that RSN reorganization is greatly associated with disability in early MS [96]. Rocca et al. and Bonavita et al. reported correlations between the decrease of FC at DMN and disability in patients with advanced MS, suggesting that a dysfunction of the DMN may be responsible for cognitive deficits in MS [97, 98]. Moreover, Lowe et al. observed in MS patients reduced connectivity in a motor RSN when compared to controls [99]. These findings support the principle that the level of connectivity at rest increases at the onset of MS, before decreasing progressively, which may be related to early compensatory mechanisms that failed afterwards with the progression of MS [96]. By comparing MS patients with healthy controls an additional pattern was observed, as the first group exhibited BOLD responses in regions where no activation was found for the latter [93].

2.3.4 Schizophrenia

Schizophrenia is one of the most common psychiatric disorders, affecting about 1% of the world's population, and also one of the most debilitating with increased mortality, suicide risk and reduced quality of life [75, 100]. Onset typically occurs in late adolescence or early adulthood, with a peak between ages 18 to 25, when the prefrontal cortex is still developing [100] Clinically, it is diagnosed based on paranoid delusions, auditory hallucinations, apathy, and cognitive impairment (mostly in language, memory, attention, and executing function domains) [101, 102].

This mental disorder is believed to arise from disruptions in brain connectivity, as it affects functions that rely on efficient communication between several brain regions [102]. For that reason, many studies have used resting-state experiments to investigate the intrinsic functional organization of the brain, in hopes of finding its correlation with cognitive dysfunctions observed in schizophrenia. Although it has been suggested as a disconnection syndrome, no consistency has been found in carried out studies [103]. Lynall et al. reported decreased FC in people with schizophrenia, whereas increased diversity of functional connections [104]. On the other hand, Whitfield-Gabrieli et al. found increased FC within the DMN in patients in the early phase of the disorder and their young first-degree relatives [105]. Moreover, DMN subcomponents exhibit differential behaviour in schizopherenia, as Skudlarski et al. observed increased

FC within a limited part of the DMN [106]. On a same note, Venkataraman et al. reported that patients with schizophrenia exhibit two distinct patterns in comparison to healthy controls: abnormal increased connectivity between the medial parietal and frontal regions, and decreased connectivity between the medial parietal and temporal regions [102]. Although observed changes in FC are not yet fully understood, these findings confirm that people with schizophrenia are likely to have a less strongly integrated, more diverse profile of brain functional connectivity [104].

2.3.5 Autism

Autism Spectrum Disorder (ASD) is a chronic, neurodevelopmental disorder with average onset at 2 to 4 years of age, nowadays affecting nearly 1% of children worldwide [107, 108]. ASD is characterized by a wide variety of symptoms, including deficits in reciprocal social interactions, abnormal development and use of language, and unusual restricted and repetitive behaviours [109]. These symptoms go from mild to severe, and no two individuals with ASD have the exact same profile, with cognitive abilities ranging from profound disability to exceptional intelligence [110].

A growing number of neuroimaging studies have investigated functional variations in the brains of people with autism at rest, and multiple findings of abnormal FC were observed [67, 110]. For instance, Assaf et al. reported decreased FC in the DMN of ASD patients compared to controls, and demonstrated this loss to be inversely correlated with symptom severity, i.e. greater FC decrease in DMN relates to more severe symptomology [111]. On a same note, Cherkassky et al. revealed a DMN much more loosely connected in ASD participants compared to controls [112]. Shen et al. found significantly weaker FC between the amygdala and brain regions relevant to social communication and language [113]. Sensorimotor and visual regions were reported with increased connectivity to which Chen et al. related the restricted and repetitive behaviours, characteristic of autism [114]. Evidence of atypical sensory processing seen in ASD patients was supported in a study of Cerliani et al. as increased FC between primary sensory and subcortical regions was observed in individuals with ASD compared to controls, suggesting hypersensitivity to sensory stimuli [115]. Li et al. demonstrated the existence of abnormal FC within some cognition networks in children with ASD, reporting increased connectivity in a few of such networks, and decreased connectivity in the others [108].

The heterogeneous nature of ASD makes it challenging to classify this condition using brain connectivity, as a variety of connectivity patterns were observed in individuals with autism [110]. Nonetheless, all of these findings provide evidence of existing dysfunctional networks within the autistic brain, which may constitute biomarkers for the diagnosis of ASD.

3 Technical Background

Summary

Machine Learning is defined as the ability of Artificial Intelligence systems acquiring their own knowledge, by extracting patterns from raw data. Artificial neural networks are computational models inspired by the central nervous system, which have proven to be a promising and powerful tool for understanding information processing in the brain. These are widely used in Machine Learning, given their learning power, and ability to deal with high-dimensional data. When dealing with such data, extracting patterns may not be an easy task. Dimensionality reduction is the transformation of high-dimensional data into a smaller meaningful representation. Finding appropriate tools to work with is a fundamental step, as it prepares the environment to explore techniques and create a solution for a problem.

3.1 Machine Learning

Since the rise of Artificial Intelligence, computers have demonstrated expertise in solving problems that are intellectually difficult for humans, such as tasks described by mathematical rules. The biggest challenge to AI is solving the tasks humans easily perform but have a hard time describing formally for being intuitive and subjective – for example, recognising objects in images. The solution, according to Goodfellow et al. [18], is Machine Learning (ML), an AI approach defined as "the ability of AI systems acquiring their own knowledge, by extracting patterns from raw data". The introduction of ML have allowed computers to behave in an intelligent manner, by making "intuitive" decisions based on the knowledge and the improvement gained from data and experience of real-world environments. [18]

The performance of a ML algorithm relies upon the data being used, and in particular, its representation. To get useful results, data must be represented with meaningful pieces of information known as *features*. *Feature extraction* is the process in where raw input data is preprocessed and transformed into a new set of significant variables [116]. Yet for many tasks, selecting the right features

to extract is often difficult. Representative learning is an approach which uses ML to determine the representation-to-output mapping, but also the representation itself by inferring the relationships between individual features, with minimal human intervention [18, 70]. The classic example of a representative learning algorithm is an autoencoder, which will explored further. The automatic discovery of consistency in data – *patterns* –, through the use of computer algorithms, is also often referred as *pattern recognition* [116].

Raw data is often exposed to non observable sources of variation. The aim is to separate these factors of variation and discard the irrelevant ones, when designing features or algorithms to learn the features. However, extracting such abstract features from data is an intricate task, as it is often difficult to acknowledge them due to their influence over every piece of available information. When such thing happens, obtaining the representation becomes as complex as solving the original problem. [18]

Deep Learning (DL) settles the issue in representative learning, by building complex representations out of other, simpler representations [18]. DL is a particular kind of ML that represents the real world as a nested hierarchy of concepts, which grants it versatility and great power. Unlike traditional ML algorithms, DL does not require a prior feature selection as it finds the most relevant features within data, eliminating human bias and error [17]. The interest in automated systems for disease diagnosis and evolution tracking has set off a growing attention in this AI technology and its use on big healthcare data [70].

Although ML grew out of computer science, and pattern recognition was originated from engineering, both can be seen as two parts of the same field [116]. The main AI approaches are illustrated in figure 10.

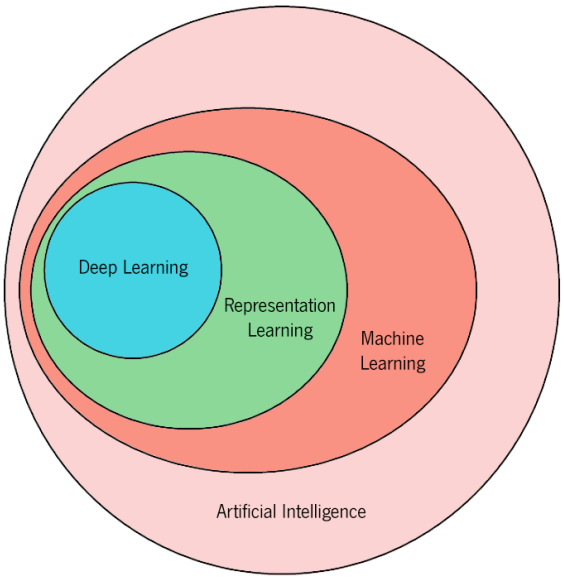


Figure 10: Venn diagram of the relationship between the main AI disciplines. Adapted from [18].

3.1.1 Artificial Neural Networks

The human brain is a collection of an average of 86 billion interconnected neurons responsible for processing and exchanging information [117, 118]. Artificial neural networks (ANN) – or just "nets" – are computational models inspired by the central nervous system [119]. Each ANN is a set of interconnected artificial neurons, structurally similar to biological neurons (figure 11), able to acquire, store and use experimental knowledge [118]. Although current technologies still do not have developed an ANN architecture as tremendous in size and as complex as the one seen in humans, these "nets" have proven to be a promising and powerful tool for understanding information processing in the brain [18].

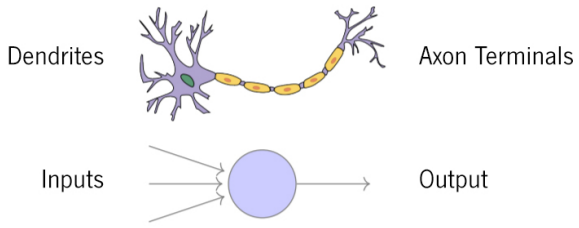


Figure 11: Comparison between the biological neuron and its counterpart in ANN. Adapted from [120].

Perceptron

The perceptron is the simplest configuration of an ANN, as it consists of a single artificial neuron [120]. Given a set of inputs $x_1, x_2, ..., x_n$, also called features, the perceptron outputs y, which is either 0 (no) or 1 (yes) [119]. Each input x_i has a corresponding w_i, expressing the importance of input to the output. To generate the perceptron's output (y), the weighted inputs are summed up and transformed by a non-linear activation function φ. Bias, b, is an additional parameter, often represented as a unitary input, $x = 1$, and a weight, $w_0 = b$. [120]

Mathematically, these operations are described in 1, and the perceptron is illustrated in figure 12.

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{1}$$

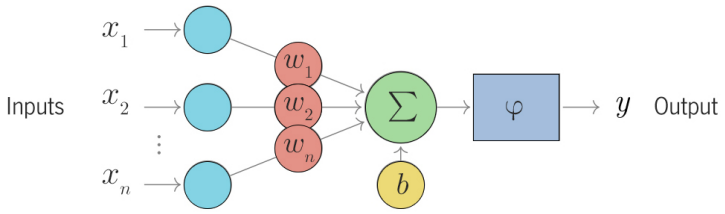


Figure 12: Perceptron. Adapted from [120].

Feedforward Neural Networks, and Recurrent Neural Networks

The topology of a neural network defines how neurons are interconnected, and ANNs display two main organizations based on it: feed-forward, and recurrent (figure 13). In Feedforward Neural Networks (FNN) the information flows in a single direction (unidirectional), i.e. the input information comes from one side, propagates through the net, and comes out as output in the other end. In Recurrent Neural Networks (RNN), there are backloops depicting the feedback connections in neurons, meaning some information can be transmitted backwards [118, 120].

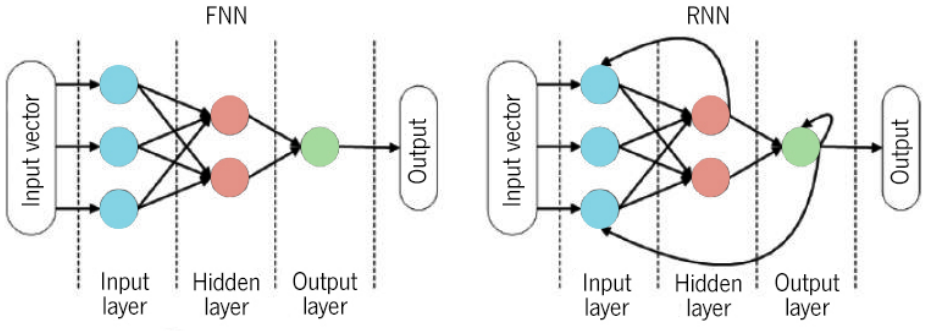


Figure 13: Feed-Forward and Recurrent topologies of an artificial neural network. Adapted from [121].

Multilayer Perceptron

A MLP is a FNN comprising a set of perceptrons, distributed by c layers, $L^{(1)}, ..., L^{(c)}$. The net derives from assembling each layer $L^{(l)}$ to the preceding later, $L^{(l-1)}$, by connecting each neuron from the first layer to all neurons of the second. Therefore, $L^{(1)}$ is the input layer, $L^{(c)}$ is the output layer, and $L^{(2)}, ..., L^{(c-1)}$ are the hidden layers. The size of $L^{(1)}$ and $L^{(c)}$ is usually determined accordingly to the problem: $L^{(1)}$ must have as many neurons as the number of features, while $L^{(c)}$ must have k neurons so that k outputs are obtained. When it comes to hidden layers, their number and size may differ, and they are crucial hyperparameters to the performance of the ANN. The number of hidden layers gives the depth of the model, so a MLP can be also called a Deep Neural Network [18].

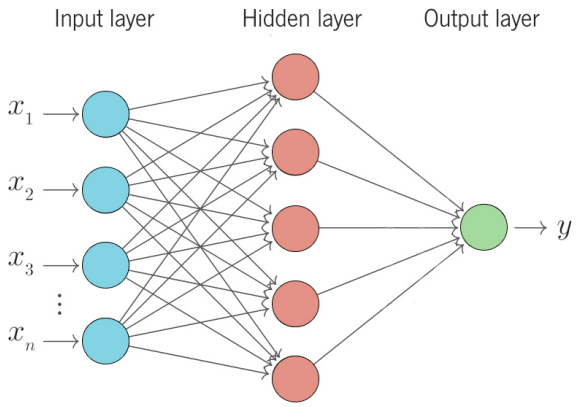


Figure 14: Multilayer perceptron. Adapted from [120].

Convolutional Neural Networks

Convolutional Neural Networks (CNN) are FNN specialized in processing data with spatial dependency between neighbourhood data points [18, 122]. This is also known as data with grid-like topology, and examples include time-series data (1D), and image data (2D). CNN, to be called so, must have at least one convolutional layer and all layers have neurons arranged in three dimensions (width, height and depth) [120, 122]. CNNs are one of the most prominent nets used in Computer Vision, as they share many properties with the visual system of the brain, and perform parameter sharing, meaning they run on any device [123, 124].

Convolutional Layer

The convolutional layer is the main building block of a CNN. It comprises a set of independent *filters* (or *kernels*), and each filter is independently convolved with the input, producing a new *feature map*. A rule to convolution is that filters of a layer must have the same number of channels of the layer's input. The filter is essentially a feature detector, extracting distinctive features at each location of the input data [70]. The neurons in a convolutional layer are solely connected to a small, local region of the input or the preceding layer – *receptive field* – in order to avoid the waste of fully-connected neurons [123].

An example of convolution where both the input and the filter have a single channel is illustrated in figure 15. The purple area represented in the input is the receptive field, defined by the size of the filter, and it is where the convolution is taking place. After sliding the filter over the input, the element-wise matrix multiplication is calculated for every location comprised in the receptive field. The summed up value is included in the feature map, and the filter then slides to the following position, repeating this procedure.

The movement of the filter is defined by the stride parameter. If the stride is 1, then the filter moves one unity; if it is 2, the filter moves two unities. [124]

The convolution operation within the first receptive field is described as:

$$3 \times 1 + 0 \times 0 + 2 \times 0 + 1 \times (-1) = 2$$

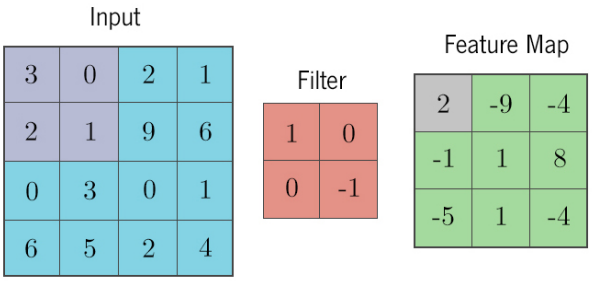


Figure 15: Convolution of a 4×4 input with a 2×2 filter, and the resulting 3×3 feature map. Adapted from [120].

Each of these units corresponds to the activation of a neuron which captured a feature within the receptive field. The presence of high values means the filter found a strong connection, and the codified feature is partially represented. If not, the connection is weaker. Deeper networks increase the chances of recognizing more complex features in the feature maps, and learn a better representation of the input data [18].

In 3D convolution, multiple convolutions are performed on the input, each using a different filter, and resulting in a different feature map. The output of the convolutional layer is all the feature maps stacked together. [124]

Activation Function

For a CNN to be robust, it must include non-linearity. Convolution is a linear operation, and repeating this step produces a set of linear activations that need to undergo an activation function [18]. Simply put, an activation function is a non linear transformation operated over the output of a layer [125]. It determines whether a neuron should be activated or not, depending on the information received by the neuron being relevant or not, respectively.

The three major types of activation functions are Sigmoid, Hyperbolic Tangent (TanH), and Rectified Linear Unit (ReLU), and represented in figure 16.

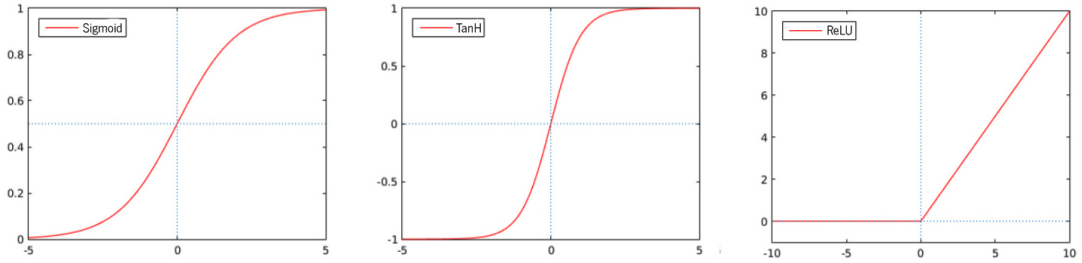


Figure 16: Activation functions: Sigmoid, TanH, and ReLU. Adapted from [125].

Sigmoid and TanH functions have similar S shapes, but differ on the intervals, as the first constrains the output of neurons onto the interval $[0, 1]$, and on the second the output of neurons range from -1 to 1. The problem with these functions is that their outputs can easily get saturated to the interval boundaries, which greatly slows down training a neural network, and makes it more possible to trap into local minimum [125].

ReLU is currently the preferred activation function, due to its computational efficiency when compared to the rest. It converts all negative inputs to zero, meaning few neurons are activated per time, converging six times faster than Sigmoid and TanH functions. In addiction, ReLU does not get saturated at the positive region, because there is no interval constraint.

Nevertheless, there are drawbacks related to using ReLU as activation function. The gradient at the negative region is zero, which implies that all the weights will not be updated during backpropagation (later explained). Also, ReLU functions are not zero-centered. This means that for it to get to its optimal point, it will have to use a zig-zag path which may be longer.

When S-shaped non-linearities are used, TanH is often preferred over the Sigmoid function, because it is zero-centered. [123, 125]

Pooling Layer

A pooling layer is strategically positioned between convolutional layers, as it reduces the dimensions of a feature map by pooling its pixels together. It aims making convolutional layers less sensitive to small shifts or distortions by remaining robust to position and shape of the features [70].

Spatial extent and stride are the two parameters that describe this type of layer. The first defines the window size to apply, and the second gives information on how many units the window moves [123].

Among all pooling operations, the most widely used in CNNs are *max pooling*, and *average pooling*. Max pooling takes the maximum value in the pooling window, while average pooling propagates the mean

of the values within the pooling window. Taking the example illustrated in figure 17, for a 2×2 window and stride 2: each window is colored differently, because the window size and stride are the same value, so the windows will not overlap. The size of the *feature map* is downsized by half in both dimensions, preserving the important information. Thus, the pooling layer helps reducing the computational cost and combating overfitting, by reducing the number of parameters to learn. [124, 126]

For a feature map with three dimensions, the depth does not change. Both overlapping and non-overlapping pooling windows are used in CNNs [125].

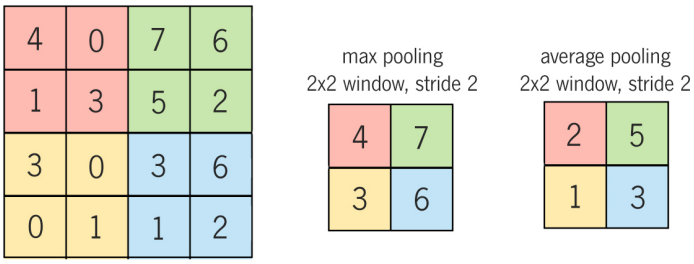


Figure 17: Max pooling and average pooling using a 2×2 window and stride 2. Adapted from [70].

Dense Layer

A typical CNN is wrapped up with, at least, a layer of fully-connected (FC) neurons, also known as dense layer – or simply MLP, as previously introduced. The name *fully connected* upsurges from the neurons in this layer having total connection to all the nodes of the previous layer. The output of both convolution and pooling layers are 3D volumes, and a FC layer expects a 1D vector, requiring an arrangement known as *flatten*.

This architecture of a CNN is fundamentally a combination of two parts. The convolution and pooling layers operate feature extraction in the first part, by extracting local features from the input and combining them to achieve higher level features. Dense layers take charge in the second part, being responsible for integrating the feature responses, and provide the final results [70, 125]. Figure 18 resumes, in illustrated form, all layers comprising a CNN.

Considering the amount of layers a CNN can incorporate – even more if it is a deep CNN – there are a ton of weight parameters to estimate. This requires a lot of data samples for model training and parameter tuning, which will be discussed in the following subsection. [70]

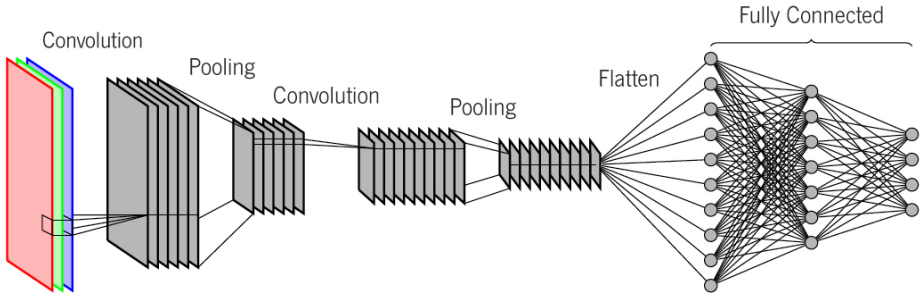


Figure 18: Representative architecture of a Convolutional Neural Network. Adapted from [120].

Optimization

Considering w as the parameter vector describing all weights of the model, the goal is to determine the optimal values for w. Most ML algorithms include some sort of optimization, which consists on either minimizing or maximizing a function $f(x)$ – designated *objective function* – by modifying x. The function wished to be minimized is also common referred as *cost function* or *loss function*, E. [18]

Cost Function

In simple terms, the cost function is a measure of how well a ML model performs. Selecting a proper cost function is essential for training an accurate model, as each cost function has its own properties helping the model to learn a specific way.

Mean Squared Error

The Mean Squared Error (MSE) function is the simplest, and yet one of the most commonly used. Considering $\hat{y}_i$ gives the results, and y_i is the target, MSE – or cost function, E – can be mathematically described by the formula 2.

$$E = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2}$$

Using MSE, E produces only positive values, and the error decreases to zero when $\hat{y}_i = y_i$, On the other hand, E increases whenever the Euclidean distance between the results and the targets increases. The aim is selecting w such that E is as close to zero as possible.

Since MSE sets larger weights on huge errors, this function is great for ensuring that the model has no outlier predictions with those errors. Yet, upon a bad prediction, the squaring part of the function magnifies the error, which may represent a disadvantage [18, 123].

Minimizing the Cost Function with Gradient Descent

Understanding the concept of minimizing a cost function is easier when representing in 2D, thus suppose $y = f(x)$ is cost function, and x and y are both real. The derivative, $f'(x)$, reveals how to change x in order to make a small decrease in y (figure 19).

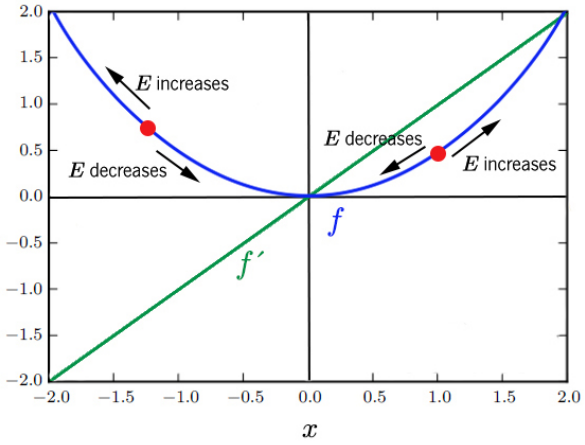


Figure 19: Optimization of a cost function. Adapted from [18].

For x smaller than zero, $f'(x)$ is negative, so moving to the right decreases f. On the opposite, for x greater than zero, $f'(x)$ is positive, thus moving to the left increases f. In case that $f'(x)$ is null, there is no information about which direction to move – these are known as critical points, and include local minimum, local maximum, and saddle points (figure 20).

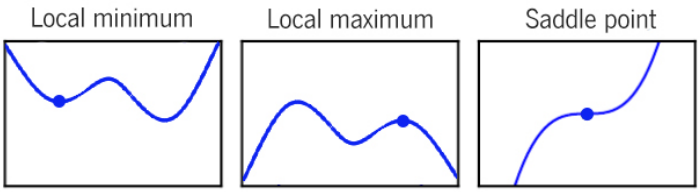


Figure 20: Critical points in a cost function. Adapted from [18].

The absolute lowest value of the cost function is a global minimum, and ideally all functions would have one. Within a ML context, it is possible to come across functions with many not optimal local minima, and many saddle points with a neighbour flat region. Optimization becomes very difficult, especially for a multidimensional input. The training usually ends when finding a very low value of f, as long they correspond to a significantly low value of the cost function, even it isn't truly minimal.

This very popular iterative technique goes by the name of *gradient descent*, and it represents a simple and effective method of optimization for neural networks. By moving x in small steps with opposite sign

of the the derivative, the cost function is reduced. [18]

A convex cost function, E, in 3D, would be similar to the smooth representation on figure 21a. However, most cost functions are non-convex, as seen on figure 21b, and there is a possibility to get stuck in a local minimum, and never converging to the global minimum of the function. There are methods to fix this, by changing the learning rate, or using momentum.

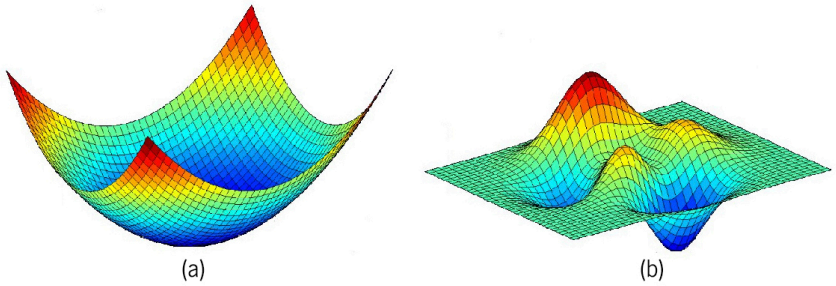


Figure 21: Convex cost function (a), and non-convex cost function (b). Adapted from [127].

Most real model implementations optimize E with multiple inputs, so the derivative concept must be generalized with respect to a vector. Thus, the gradient of the cost function, $\nabla E[\vec{w}]$, consists of a vector with all partial derivatives of E regarding each component of the parameter vector w (e.g. element 0 of the gradient is the partial derivative of E with respect to w_0). Mathematically, this vector derivative is described as:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n}\right] \tag{3}$$

The gradient descent starts with an arbitrary initial vector w, and repeatedly updates it in small steps. At each step, by computing the formula 3, and interpreting it as a vector in weight space, the gradient specifies the direction producing the steepest increase in E. Hence, by moving in the direction of the negative gradient, E is reduced. Eventually, a set of w values will be found to minimize E to an acceptably low value, and cease the training [18, 128].

Learning Rate

This size of steps taken to reach the minimum of the cost function is called *learning rate*. A substantial value of learning rate means covering more area at each step, and a faster way to get closer to the minimum, but it also means making drastic changes to the values of the weights and bias, and a big chance of overshooting the global minimum (figure 22a). With a small learning rate value there is no risk of always wandering around the global minimum, but, it will take longer to converge and an extended

training time, as with shorter "steps" more steps will be needed to reach the lowest point (figure 22b). Also, if the function is non-convex, it is possible to get trapped in a local minimum (figure 22c). [129]

There is no generic value for learning rate, as it depends solely on experimentation.

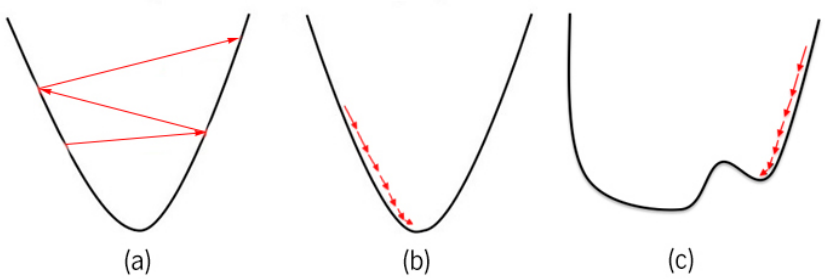


Figure 22: Effect from big learning rate (a), and a small learning rate on a convex function (b), and a small learning rate on a non-convex function (c). Adapted from [129].

An Algorithm with Adaptive Learning Rate

The learning rate is one of the hyperparameters most difficult to set, since it has a significant impact on model performance [18]. Optimizers are used to search for parameters that minimize the loss function, and there are algorithms that adapt learning rates throughout the course of learning, e.g. the Root Mean Square Propagation (RMSprop) algorithm. RMSprop restricts the oscillations in the y-direction, concentrating the direction of the steps in the x-direction [129]. Hence, the learning rate can be increased, and RMSprop can take larger steps in the x-direction, converging faster after finding a convex bowl. RMSprop has been proven to be an effective and practical algorithm for deep neural networks [18].

3.1.2 Unsupervised Learning

Unsupervised learning refers to learning inherent and useful properties of the data structure without relying on output data, because the training data is a set of unlabeled inputs [18, 70]. Unsupervised algorithms aim discovering groups of similar examples within data, and/or projecting data from a high-dimensional space down to two or three dimensions, which is the purpose of data visualization. Common unsupervised learning algorithms include k-means clustering, PCA, and autoencoders. The model performance cannot be evaluated since no labels are provided, thus the accuracy will not be measured. [116, 130]

Autoencoders

An autoencoder is a type of neural network that attempts to replicate its input to its output [18]. Its feedforward architecture allows the information to move in a single direction in the network, from the input nodes to the output nodes [131], and it consists of the two following parts: encoder, and decoder. The encoder, f, is responsible for converting the input data, x, into a different representation, h, known as code or latent-space representation. This code is where important features are preserved. The decoder, g, then proceeds to reconstruct the input on the autoencoder's output, r, by converting the new representation, h, back into the original input format [18, 131]. The general structure of an autoencoder is illustrated in figure 23.

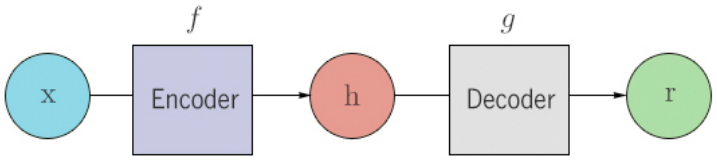


Figure 23: General structure of an autoencoder. Adapted from [18].

The input copying task may sound pointless, especially if an autoencoder was capable of outputting an exact, perfect replica of the input. Although autoencoders are trained to preserve as much information as possible, the game changer is that they are, in fact, designed to copy the input only approximately to resemble the training data. By being forced to prioritize which aspects of the input should be copied, autoencoders are trained to make the new representation, h, taking on useful properties of the training data. The autoencoder must balance between being sensitive to the inputs enough to build a proper and accurate reconstruction, and insensitive enough to the inputs which the model does not memorize or overfit the training data instead [132]. This trade-off forces the model to keep only the variations in the data required to reconstruct the input, without holding on to redundancies within the input [132]. Rather than setting the output of the decoder as target, the interest is ultimately on training the autoencoder to learn a sparse set of features to truthfully describe the original data [133, 134]. In other words, the challenge is getting the model to learn a generalizable and meaningful latent space representation [132].

Undercomplete Autoencoders

A potential and viable way to force the autoencoder to capture important features of the data is constraining the hidden layer, h, to have a smaller dimension than the input, x [18]. An autoencoder

which produces a compressed representation of the input is known as undercomplete. Its architecture is the simplest out of all types, and it is distinctive for its bottleneck in the middle limiting the amount of information flowing through the network [132] (figure 24).

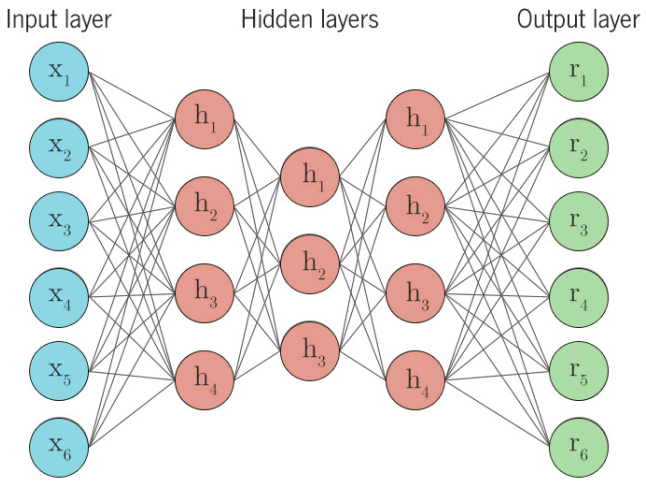


Figure 24: Undercomplete autoencoder. Adapted from [132].

For this autoencoder to perform a reasonable result, the input data must have internal correlation or dependency on its features, otherwise it will do a poor job on the compression and subsequent reconstruction [131, 132]. Taking into example data with digits "0" and "1", digit "0" is circular, while digit "1" contains more of a straight line – the autoencoder will learn this fact and encode it in a more compact form [131].

The learning procedure for this type of autoencoder is described as minimizing a loss function,

$$L(x, g(f(x))) \tag{4}$$

where L denotes a loss function penalizing $g(f(x))$ for being dissimilar from x, which translates into measuring the differences between the original input, x, and its reconstruction, r. In case of the decoder being linear and L being mean squared error, an undercomplete autoencoder will learn the principal subspace of training data, which is the same as PCA's, as a side-effect [18]. Neural networks are capable of learning non-linear relationships, and it is possible to take advantage of it. If both encoder and decoder functions are non-linear, then a more powerful non-linear generalization of PCA is achievable using autoencoders. Figure 25 shows the different between the two [132]. However, if the encoder and the decoder are given too much capacity, the model will learn to memorize the training data, and consequently the autoencoder will fail on extracting useful information about the distribution of the data [18].

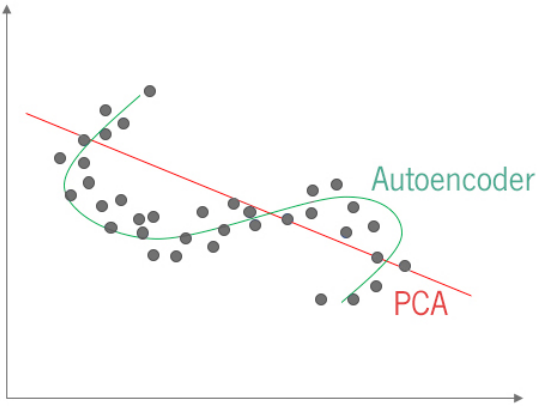


Figure 25: Linear (PCA) and non-linear (autoencoder) dimensionality reduction. Adapted from [132].

Regularized Autoencoders

An alternative method for imposing an information bottleneck and not compromising data generalization is offered with the use of regularized autoencoders. Instead of limiting the model capacity and keeping the code size small, regularization terms can be comprised in the loss function to achieve desired properties and discourage overfitting. These properties include sparsity of the representation, and robustness to noise or to missing inputs. [18, 132, 135]

Sparse Autoencoders

A sparse autoencoder explores an interesting approach towards regularization, as it uses a sparsity constraint to penalize activations within a layer, in (h), on the hidden layer, h, is added to the loss function, L, as seen in the formula,

$$L(x, g(f(x))) + \Omega(h) \tag{5}$$

where $h = f(x)$ denotes the encoder output, and $g(h) = g(f(x))$ denotes the decoder output. This regularization forces h to selectively activate regions of the network, thus representing each input as a combination of a small number of nodes per data sample. If the value of the node is close to 1, it is activated. If not, the node is deactivated and its output to the following layer is zero [135]. Since all activated nodes are data-dependent, different inputs result in activation of different nodes, and the network is compelled to condense and store only important features of the data [132, 135]. Unlike undercomplete autoencoders which use the entire network for every instance, regularized autoencoders are compelled to condense and store only important features of the data. A generic sparse autoencoder is illustrated

in figure 26. The opacity of each node corresponds to whether the node is activated (full opacity) or not (slightly transparent) in the network [132].

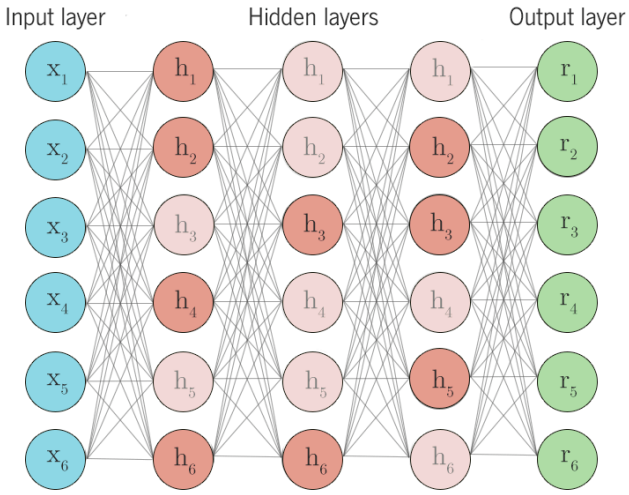


Figure 26: Sparse autoencoder. Adapted from [132].

A sparse autoencoder is forced to selectively activate regions of the network depending on the input data, whereas an undercomplete autoencoder uses the entire network for every instance. As a result, the network's capacity to memorize the input is restricted, without its ability to extract features from the data being limited. Sparse autoencoders are typically used to learn features for another task, such as classification. The penalty can be seen as a regularized term added to a feedforward network whose primary task is to copy the input to the output, and possibly perform some supervised task that depends on these sparse features.

Denoising Autoencoders

So far, all mentioned autoencoders exhibited different techniques to prevent the overfitting problem, but every one of them fall into the concept of reconstructing an output identical to the input. Another approach towards developing a generalizable model is to add random noise to the input data but train the autoencoder to predict the original, noise-free data as its output. This is called a denoising autoencoder (DAE), as its task is to remove the added noise and produce the underlying meaningful data of the input. DAE are trained to minimize a loss function similar to a undercomplete autoencoder's, but with a particularity on the reconstruction error,

$$L(x, g(f(\widetilde{x})) \tag{6}$$

where x denotes the input, and $\widetilde{x}$ denotes a copy of x which was corrupted by noise. The trick of developing a mapping to memorize the training data does not work, as the input and the output are not the same anymore [132]. For this reason, the network is forced to undo this corruption and reconstruct the unadulterated input, rather than just copying it [18, 135]. DAE is trained to reconstruct the clean input, x, from its corrupted version, $\widetilde{x}$. This technique results in a robust neural network, as this one becomes immune to noise in input, up to a certain extent, while the decoder function learns to resist small changes in the input [135].

Contractive Autoencoders

For small changes to the input data, it is expected the encoded state to be preserved. The autoencoder model can be trained accordingly, by demanding a small value on the derivative of the hidden layer activations regarding the input [132]. This model is typical of contractive autoencoders (CAE), compelling the feature extraction function to withstand infinitesimal perturbations of the input. In other words, a CAE is forced to learn how to shrink/narrow a neighbourhood of inputs into a smaller neighbourhood of outputs. As illustrated in figure 27, similar inputs (blue) are contracted to a constant output (green function), based on what the model observed during the training phase (red). It's noticeable how the slope of the reconstructed data, $r(x)$, is substantially zero for neighbourhoods of input data, x [132].

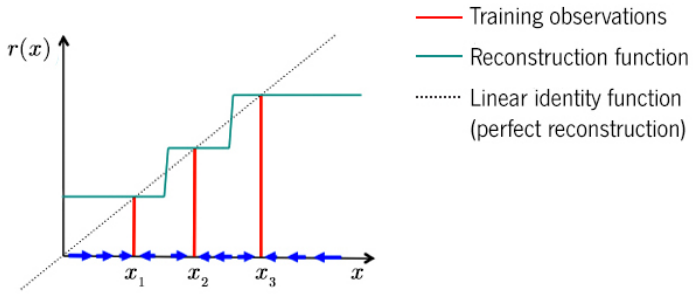


Figure 27: Training on Contractive Autoencoders. Adapted from [132].

This is achievable by penalizing instances where a small change in the input leads to a large change in the encoding space. In terms of the loss function, in addition to the reconstructing error L, a penalty term is added to deal with the large value of derivative of the encoding function,

$$L(x, \widetilde{x}) + \lambda \sum_i \left\| \nabla_x a_i^{(h)}(x) \right\|^2 \tag{7}$$

where $\nabla_x a_i^{(h)}(x)$ defines the gradient field of the hidden layer activations with respect to the input, x, summer over all i training examples [132].

Variational Autoencoders

Variational autoencoders (VAE) are based on non-linear latent variable models. In a latent variable model, it is assumed that observable inputs, x, are generated from hidden variables, y, whom contain important properties about the data. VAE consist of two neural networks, the first being for learning the latent variable distribution, and the second for generating the observables from a random sample obtained from latent variables distribution [135]. They are highly popular for generating images.

t-SNE

A popular technique for exploring high-dimensional data was introduced back in 2008 by van der Maaten and Hinton [136]. The name, t-SNE, arises from this method being a variation of Stochastic Neighbour Embedding [137]: it uses a symmetrized version of the SNE cost function with simpler gradients and it uses a t-Student distribution rather than a Gaussian one to compute the similarity between two points in the low dimensional space [137]. This way it is easier to optimize and produces better visualizations by reducing the tendency to crowd points together in the centre of the map. t-SNE has been emerging in the ML field as it has proven to be better than other existing techniques at creating a two-dimensional map using a set of points in a high-dimensional space [136].

3.1.3 The Curse of Dimensionality

Many ML problems become difficult when the number of dimensions is high, and such phenomenon is know as the curse of dimensionality [18]. For a growing number of features or dimensions, the amount of data needed to accurately generalize grows exponentially. Dimensionality reduction is the transformation of high-dimensional data into a smaller meaningful representation [138]. This is based on the assumption that the dimension of data is artificially inflated, and the number of parameters needed to represent data is much lower. Hence, to reduce the size of a representation, the dimensionality reduction algorithm must identify and remove redundancies [18]. Dimensionality reduction eases compression and visualization of high-dimensional data, and techniques include PCA, autoencoders, and t-SNE [18, 138].

PCA is by far the most popular unsupervised linear technique for dimensionality reduction [138]. It finds the directions of maximum variance in high-dimensional data, and projects data onto a new linear

subspace with fewer dimensions than the original, while still retaining most of the information [139]. PCA aims to identify patterns in data, based on its features strong correlations.

Real-world data, however, is likely to assemble a highly non-linear manifold[1], and non-linear techniques may be one big step ahead of PCA [138].

Autoencoders have been also successfully applied to dimensionality reduction, and it can be looked at as a way to transform representation [18]. As previously seen, if restricting the number of hidden layer nodes to be less than the number of original input nodes, it will result in a compressed representation of the input [140]. In case of the size of the hidden layer becoming smaller than the intrinsic dimension of data, it will result in loss of information. Still, the decoder might be able to learn to map the latent space to a specific input, as autoencoders are highly non-linear. With appropriate dimensionality, and sparsity constraints, autoencoders can learn data projections that are more interesting than PCA's [18]. In addiction, autoencoders might be beneficial to feature extraction, as the latent-space representation can be fed to other algorithms, e.g. for clustering purposes [18].

Another non-linear technique for dimensionality reduction which adapts to the underlying data is t-SNE. It embeds high-dimensional data into a lower dimensional space by preserving the neighborhood structure of data [141]. Moreover, t-SNE attempts to find patterns in data, being also well-suited for visualization [142]. ML often requires working with up to millions of dimensions, and since visualizing high-dimensional data is a challenge to humans, they would greatly benefit from using t-SNE [143].

3.2 Working Environment

Before searching the solution for a problem in Medical Informatics, setting up the working environment is a requirement. It consists of a set of software which enables writing programs for a particular language or platform. Within the context of this dissertation, this environment includes a operating platform, an Integrative Development Environment (IDE), and a DL framework. Choosing the proper working tools involves comparing options available for each case.

3.2.1 Operating Platform

The environment on which computer programs can run is the operative platform. Virtualization has become an important tool in computer system designing, and operating platforms have different abstraction levels [144].

[1] Connected region of points [18].

Virtual Machine

A Virtual Machine (VM) is a software computer that runs an Operating System (OS) and applications. The VM, also known as *guest*, is created within a full, persistent execution environment comprised of a *hypervisor* and a *host* [144]. The host is the underlying platform supporting the VM, i.e. the hardware. The hypervisor is the virtualizing software with access to all the hardware resources and responsible for managing them.

A single-host can support several VMs simultaneously, and each VM is completely sandboxed from the rest of the system, as the VM technology provides isolation between multiple guests running concurrently on the same hardware platform [144]. This is an important feature because in case of a guest's security being compromised, or a guest's OS failing, the software running on other guests is not affected. Additional advantages of VM include being easily managed, and widely available, and allowing multiple users to pick a different OS and a variety of applications for each VM to run on the shared host. Despite advantages, VMs run rather slow and have performance issues [145]. The classic implementation of a VM system has four main tiers, starting from physical hardware, a virtualizing software, a copy of an OS, and applications [145]. For several VMs, each runs a virtual copy of all the hardware the OS needs to run, there are multiple copies of OS, and even more applications, creating a quite heavy and less efficient system, adding up a lot of RAM and CPU cycles.

Docker

Docker implements OS virtualization to isolate software applications from the environment they are running on, using containers for the purpose [146, 147]. Each container packages its code and dependencies together (e.g. libraries, and configurations), solving "works on my machine" common issues and ensuring compatibility across platforms [146].

Despite appearances, Virtual Machines (VM) and Docker containers work differently. The latter are more lightweight considering they share the OS kernel with the host machine, not requiring an OS per application unlike VMs (figure 28) [146, 148]. Launching a Docker container takes about one second, whereas VMs are more time consuming and demand more resources to function. Moreover, Docker containers are the strongest for default isolation in the industry [146].

Docker makes it easier to create, deploy, and run applications, providing portability, scalability, efficiency and security at once, and such properties have skyrocketed its popularity [146].

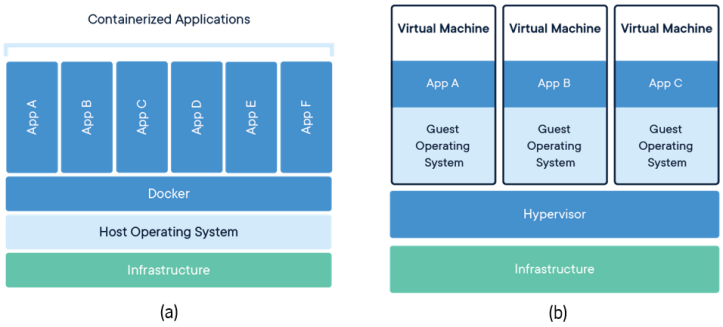


Figure 28: Docker containers virtualize the OS (a), whereas Virtual Machines virtualize the hardware (b). [146].

3.2.2 Deep Learning Framework

Nowadays there is a myriad of open source Deep Learning frameworks intended to simplify the implementation of large-scale and complex DL models [149]. They provide a succinct approach for defining models using a collection of pre-built and optimized components, without the need to develop a model from scratch [149]. These frameworks are built in different ways and for diverse purposes. As DL continues to grow, DL frameworks are improving, earning users' attention rapidly, and adapting to their needs.

Tensorflow and Keras

Tensorflow is an open source computational framework for designing, building, and training DL models, and it was released in 2015 by Google Brain Team [123]. Despite being implemented in C++, this framework can be accessed and managed using other programming languages such as Python, Java [150]. Built to be deployed at scale and accessible for everyone, Tensorflow includes different application programming interfaces (API) for both FNN or RNN architectures, and runs on multiple CPUs or GPUs, and also mobile operating systems [151]. Another interesting feature is the tensorboard, which allows the developer to monitor either graphically and visually Tensorflow's current tasks, which is particularly useful for program debugging. This framework operates with a static computation graph for efficiency, which means it must be defined and compiled before executing [150].

Keras is a minimalist library written in Python, developed for quickly and effortlessly implementing artificial neural networks. It is used as a high-level API which runs over Tensorflow backend (figure 29 [150]. Keras can execute on CPUs and GPUs, and it provides a wide range of highly powerful, and abstract building blocks to build complex models [152]. This library is ideal for DL newbies, as it provides a much more readable and succinct code [150].

By the end of 2018, Tensorflow had the most GitHub activity, ArXiv articles, Google searches, and books on Amazon, and most developers are using it [153]. It is considered the most popular DL framework today [150]. Keras is also a favourite, for being lightweight, minimizing user actions, and making it easier to understand DL models [149].

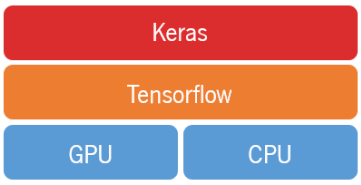


Figure 29: Software and hardware stack using Keras and Tensorflow.

PyTorch and FastAI

PyTorch is a DL framework implemented as a Python package, yet shares some C++ backend with Torch, a Lua-based framework for DL as well [149, 154]. PyTorch is ideal for small-scale or academic projects, as the process of training a neural network is straightforward and simple [150]. An interesting feature to this framework is supporting dynamic computation graphs, meaning it enables making changes to the architecture during runtime [149, 150].

PyTorch has seen increasing growth and it is considered a competitor to Tensorflow [154].

FastAI is an API that sits on top of PyTorch, and it was inspired by Keras and requires less code for strong results [153].

3.2.3 Integrated Development Environment

An IDE is a program meant for software development, which allows to write, test, and debug code easily. Although most IDEs support many programming languages, the focal point will be over Python-specific IDEs.

Spyder

Spyder stands for Scientific Python Development Environment, and it is an open-source IDE for Python available for Windows, MacOs and Linux [155]. This IDE contains a multi-language editor with robust syntax highlighting, real-time code analysis tools, automatic code completion, and debugging [155, 156]. Spyder includes IPython, an interactive console, that allows to execute commands and interact with the results, e.g. visualizing data [156, 157]. Advantages of using Spyder include being a lightweight IDE, and easy

to work with, and thus interesting for Python beginners. Not being as advanced as some other IDEs out there, on the other hand, makes it less attractive for experts. In addiction, the easiest way to get Spyder is installing the Anaconda package, as Spyder is not delivered as a standalone executable, which itself is a downside [157].

PyCharm

PyCharm is a very powerful and full-featured IDE, available in community and professional editions, both working on Windows, Mac OS X, and Linux platforms [158]. It supports Python development directly, featuring a smart code editor, code analysis, error highlighting, and a debugger with graphical interface [155, 158]. The potential of using PyCharm appeals to all kinds of Python developers, from the newcomers to data scientists, which is a major point in its favor [157]. There is a list of additional plugins that can be downloaded to the IDE, expanding its features [155, 159]. Yet, these are exclusive for the professional users, which limits the open-source version in terms of proficiency. For instance, integration with Docker is available for the professional edition only [157]. Another drawback concerning PyCharm is loading slower for being a heavier IDE, and more difficult to install in Ubuntu when compared to Spyder [158].

Jupyter Notebook

Jupyter Notebook is an open-source web application that enables users to combine raw text, markdowns, code and its output in a single document known as *notebook* [155]. Notebook documents are human-readable documents as well as executable documents, making Jupyter Notebook a promising alternative to the traditional IDEs [160]. The code contained in a notebook is executed by a kernel, which changes accordingly to the programming language [160]. These notebooks can be exported in editable formats, such as *.ipynb* (notebook format) or *.py* (python modules), or as PDF or HTML files. Moreover, notebooks help saving time running code, by splitting long experiments into smaller ones that can be executed independently [161]. Nonetheless, Jupyter Notebook is lacking a good debug functionality.

Notebooks are often used to get started with Keras, and for DL experiments [161]. Moreover, integrating notebooks with Docker containers provides real time script editing, running and visualizing the results in a web browser, without the need to get inside the container.

3.2.4 Medical Imaging Archiving

Imaging and related data are being produced at a growing rate. Sharing data resources at different scales, from large studies to individual laboratories, have triggered an interest in storing solutions that make data available in a continuous and protected way [162].

Network Attached Storage

Network Attached Storage (NAS) refers to storage devices directly connected to a network which provide remote file access services to any host or client connected to the same network [163]. A NAS system comprises one or more storage devices for storing data, and an engine implementing the file access services [163]. The communication between NAS and users on a local area network (LAN) is via Ethernet, and each NAS is defined by its unique Internet Protocol (IP) address [164].

The NAS implementation simplifies storage management by centralizing storage under a consolidated place, and increases data availability, since both hardware and software in a NAS are developed and tested to run together, and free from user configurations [164]. Moreover, NAS improves sharing of data among users, as it is accessible to all of them at any time. When in need for more storage space, the total disk space of a NAS system can be expanded by adding more storage devices in connected arrays, for them to work as a single unit. Yet, a concerning drawback of using NAS is data integrity and privacy being more vulnerable to malicious users of the network, and even more if clinical data is involved [164].

XNAT

The eXtensible Neuroimaging Archive Toolkit (XNAT) is an open source data management tool developed by the USA-based Biomedical Informatics Research Network (BIRN), which has been increasingly adopted by worldwide biomedical imaging researchers [162]. XNAT is used for importing, storing, processing, analysing, searching, filtering, and securely distributing imaging and related data to multiple users (figure 30) [165].

The XNAT workflow was inspired on the common practices that neuroimaging labs use to manage their data from acquisition to publication [166]. The workflow begins by loading metadata and imaging data into the archive, using web-based forms or XML documents for the first, and standard transfer tools or an upload tool for the second. A quality control process follows to check integrity of captured data, which is temporarily withhold until inspected and approved by an authorized user. At this point, imaging data is still raw, and XNAT provides tools to preprocess a set of scans, and an online image viewer to analyse

intermediate and postprocessed results. Data can be easily browsed, sorted, searched, and downloaded through XNAT's web-based interface, e.g. for local use. Additionally, XNAT enables collaboration between research groups or other entities, and promotes data sharing with the general community.

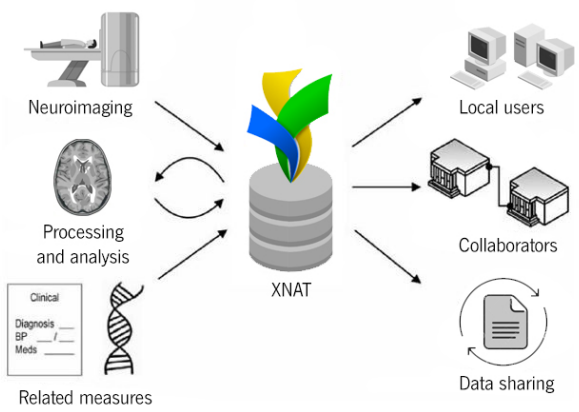


Figure 30: XNAT captures data from multiple sources and distributes it to a variety of end users. Adapted from [166].

The XNAT framework relies on eXtensible Markup Language (XML) and XML Schema to model stored data [165]. XML Schema is the standard language for delineating open and extensible XML data formats, and grants XNAT's extensibility [166]. This feature is crucial in a rapidly advancing field like neuroimaging where managed data types are likely to change and evolve quickly.

The basis of XNAT's data model rests upon three abstract data-types defined in a base XML Schema Document (XSD): *Projects*, *Subjects*, and *Experiments* [166]. From these, concrete data types may be derived, by defining their relationship to the abstract data types. A *Project* is used to define a collection of data stored in the server, and all data must be assigned to a project. Users are either owners, members, or collaborators of the project they are included. This grants data security in two ways: if included in a project, users are given permissions which define who can read, write or delete data of that project; if not included in a project, users are not given access to data of any kind within that very project. Each project can comprise multiple subjects, and a *Subject* is anyone who participates in a study. A subject record may include fields of information regarding, for instance, the age or gender of the subject. A subject can display numerous experiments. An *Experiment* is an event by which imaging or non-imaging data of a subject (or more) is acquired. An experiment that is intrinsically associated with a subject is a *Subject Assessment*, and by default, XNAT provides solely one extension of such data-type, which is imaging session. Hence, an *Imaging Session* is a specific type of subject assessment, and it is used to capture the data acquired in the normal workflow of imaging studies, i.e. scanned, preprocessed and processed stages. *Scans* represent

individual scan series from a single imaging session, and usually contain multiple imaging scans. A single imaging session can comprise several image reconstructions, and assessments. *Reconstructions* are frequently the result of a preprocessing stream over data, whereas *Assessments* represent processed data and its analysis, and may contain both generated images and statistics. Scans, reconstructions, and assessments all depend on the type of imaging session, e.g. MR session, PET session, or CT session.

XNATUM is a Python package that interacts directly with any XNAT instance, providing users direct access to all imaging data stored in XNAT [167]. XNATUM uses Python classes to create an initial persisting server configuration, that holds the login and server information. When the session is established, it enables to list and download data, using functions designated for the purpose.

4 Experiment Setup

Summary

The materials used for this experiment setup include resting state fMRI data from two sources: In-House data provided by the Life and Health Sciences Research Institute (Portugal), and HCP data from the Human Connectome Project initiative (USA). The methods are subdivided in Data Download, Data Preprocessing, Dimensionality Reduction, and Analysis. A case study was performed using the In-House data to evaluate the performance of these methods, as it is concerns of a smaller number of cases and lower dimensionality.

4.1 Domain Knowledge

The first stage to the present experiment setup was domain knowledge in neuroimaging and machine learning. Domain knowledge is described as the understanding of a particular domain, along with the environment in which the domain operates. In order to contribute such fields of knowledge, one must be first introduced to tools and technologies that are used within it, and becoming familiar with them.

Throughout the development of this experiment setup, it was necessary to gather with experienced users of the technologies implemented and concepts explored, for a more introspective analysis, and clearer decisions.

4.2 Materials

The materials of this dissertation are intended to provide means for designing and developing an experiment setup for extracting FC at a voxel level, and finding patterns on it. The data and tools used will be described in the present section.

The resting state fMRI data presented was obtained from two different sources, and named accordingly. In-House data comes from a study performed in a sample of a cohort of subjects recruited

for the SWITCHBOX Consortium project, and it was provided by Life and Health Sciences Research Institute (ICVS). It is stored in a XNAT server, XNAT@DI, with restricted access. HCP data comes from a global initiative named WU-Minn Human Connectome Project (HCP) consortium, whose aim acquiring, analysing and freely sharing information on brain connectivity obtained by imaging healthy adults [168]. It is stored in a XNAT server and publicly available in a platform named ConnectomeDB[1].

4.2.1 In-House Data

This sample comprises 120 healthy participants (57 males, 63 females, mean age 65.94, minimum age 51, maximum age 87). The study was conducted in accordance with the principles expressed in the Declaration of Helsinki and was approved by the Ethics Committee of Hospital de Braga (Portugal).

Data Acquisition

Imaging sessions were performed at Hospital de Braga on a clinical approved 1.5 T Siemens Magnetom Avanto MRI scanner (Siemens, Erlangen, Germany), using a Siemens 12-channel receive-only head coil. A functional BOLD sensitive EPI scan was collected, with the following parameters: 30 slices, TR = 2000 ms, TE = 30 ms, flip angle (FA) = 90°, slice thickness = 3.5 mm, slice gap = 0.48 mm, voxel size = 3.5 x 3 x 3.5 mm, FoV = 1344 mm, and 180 volumes.

During acquisition, subjects were instructed to remain still, awake, with their eyes closed, as motionless as possible and to think of nothing in particular. None of the participants fell asleep during acquisition, and each session took a total of 6 minutes to complete the acquisition.

Data Preprocessing

Data stored had been already preprocessed, and the preprocessing pipeline included the following methods: (i) format conversion from DICOM to NIfTI, using the dcm2nii[2] converting tool; (ii) skull stripping on the structural data using the Brain Extraction Tool (BET) from FSL[3], with a 0.18 threshold; (iii) linear and non-linear registration of the structural space to a standard space, and resampling to 2 mm isotropic voxel size; (iv) removal of the first 10 seconds of fMRI acquisition for signal stabilization (TR = 2s, first 5 volumes removed); (v) slice timing correction and motion correction applying the rigid body alignment of each volume with the mean volume using MCFLIRT[4]; (vi) mask applied to all volumes resulting solely in

[1] https://db.humanconnectome.org/

[2] https://www.nitrc.org/plugins/ mwiki/index.php/dcm2nii:MainPage

[3] Library of analysis tools for fMRI, MRI, and DTI brain imaging data. https://fsl.fmrib.ox.ac.uk/fsl/fslwiki

[4] https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/MCFLIRT

cerebral tissue acquisitions; (vii) coregistration of the functional data using FLIRT[5], with structural data as reference; (viii) spatial normalization of the native structural acquisition for the MNI152 standard space, and a resampling to 2 mm isotropic voxel size; (ix) motion scrubbing, firstly using a mask of the white matter and CSF regions to calculate the mean values of such regions, and afterwards using a subject-level General Linear Model (GLM) to perform a linear regression of motion parameters and remove residual high motion-induced fluctuations; (x) temporal filtering using a band-pass filter with a bandwidth from 0.01 to 0.08 Hz to remove confounding factors.

4.2.2 HCP Data

The HCP project contains, at the moment, fMRI data on 1096 healthy individuals (500 males, 596 females; 229 participants aged 22-25, 478 aged 26-30, 376 aged 31-35, and 13 aged 36+). These subjects records belong to several data releases, dating from 2013 (Q1) to 2015 (S1200), which are continuously updated to provide data with greater quality, and additional resources. For instance, the last update dates April 2018. This study was approved by the Institutional Review Board at the Washington University in St. Louis (USA).

Data Acquisition

Imaging sessions were performed at Washington University in St. Louis (WashU) on a 3T Siemens Magnetom Connectome Skyra MRI scanner, using a Siemens 32-channel receive head coil. The BOLD signal was extracted using the gradient-echo EPI with the following parameters: 72 slices, TR = 720 ms, TE = 33.1 ms, FA = 52°, slice thickness = 2.0 mm, voxel size = 2.0 x 2.0 x 2.0 mm, FoV = 208 x 180 mm, echo spacing = 0.59 ms, and 1200 volumes.

During acquisition, subjects were instructed to keep their eyes opened, with relaxed fixation on a projected bright cross-hair on a dark background, and presented in a darkened room. Each session took approximately 15 minutes.

Data Preprocessing

Although HCP provides the unprocessed NIfTI data for all scans, the HCP data included in this experiment setup had been preprocessed with the HCP functional pipelines, *fMRIVolume* and *fMRISurface*. Both implement different steps of the standard preprocessing pipeline. The *fMRIVolume*

[5]https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FLIRT

pipeline performs (i) slice-timing correction using interpolation; (ii) motion correction by realigning volumes to compensate for subject motion; (iii) coregistration of the fMRIdata to the structural data; (iv) spatial normalization of the native volume to the MNI152 standard space; and (v) masking data with the brain mask to extract the brain. The *fMRISurface* pipeline is solely responsible for (vi) spatial smoothing, using a Gaussian surface algorithm and a 2 mm FWHM [169].

Further preprocessing, i.e. motion scrubbing and temporal filtering are not included in the HCP functional pipelines for removing significant amounts of information. For the purpose, the motion parameters are provided alongside to the NIfTI preprocessed data, and they must be used to finish the preprocessing of the HCP data.

4.2.3 Working Station Configuration

The programming language chosen is Python, and the technical specifications of the machine used to preprocess data, and develop the methods are enunciated in table 2. Given the options to the working station configuration for this experiment setup, and based on advantages and disadvantages depicted in subsection 3.2, the selected tools are the following: Docker as Operating Platform; Tensorflow and Keras as DL Framework; Jupyter Notebook as IDE; XNAT as Medical Imaging Archiving.

Table 2: Specifications of the machine used for the experiment setup.

OS	Ubuntu 14.04 LTS (64 bit)
CPU	Intel Xeon CPU E5-1650 V2 @ 3.5 GHz – 6 Cores
RAM	64 GB
Disk space	2 Disks of 2 TB 1 Disk of 512 GB
GPU	NVIDIA QUADRO P6000 (24 GB of GDDR5X dedicated memory) Cuda Parallel-Processing Colors 3840 FP32 Performance 12 TFLOPS

4.3 Methods

Besides domain knowledge and materials, an experiment setup foresees a series of methods. For the purpose of this dissertation, four methods were developed: Data Download, Data Preprocessing, Dimensionality Reduction, and Analysis. Each method employs several functions to achieve its outcome, and it is supported by Python packages and modules. These include NumPy, Nibabel, Matplotlib, ScikitLearn, Seaborn, Glob, and OS, and a brief description for each is presented below.

- **NumPy** [170] – Library for scientific computing with Python, mainly focused on array representation and processing. Images can be represented as multidimensional arrays of numbers, and NumPy arrays are often used for storing imaging data. This package supplies a broad range of functions to perform easy and fast computations between such arrays. NumPy forms the base of all packages used in this experiment setup, as brain scans are saved in n-dimensional NumPy arrays, and everything else is built on top of that.
- **NiBabel** [171] – Package for handling neuroimaging data, which provides reading access to a collection of neuroimaging file formats, e.g. ANALYZE, NIfTI, AFNI BRIK, and Philips PAR/REC. It comprises designated functions for retrieving and saving the header information contained in the file, and the image itself. This package was mostly used on this experiment to load images, and extracting data from them.
- **Matplotlib** [172] – Library for 2D plotting of arrays, which offers a wide variety of plots with great quality to chose from, e.g. line plots, histograms, bar charts, and scatterplots. This package is very helpful for representing data in visually easier to grasp figures, to quickly detect and understand patterns, and make correlations in data.
- **Scikit-Learn** [173] – ML library that features several classification, regression, and clustering algorithms for predictive data analysis. It is designed to interoperate with NumPy. This package was used for implementing the t-SNE algorithm.
- **Seaborn** [174] – Library for data visualization, based on matplotlib, that provides a high-level interface for drawing informative statistical graphics. This package was used to visualize data resulting from the t-SNE computation.
- **Glob** [175] – Module that finds and lists all pathnames matching a pattern specified within a string. This pattern can be either absolute, or contain shell-style wildcards (i.e. *). If using wildcards, and defining an extension in the pattern, all files with such format will be retrieved independently of the directory their in. This functionality was useful to get the paths of all brain scans being studied.

- **OS** [176] – Module that provides functions to interact with the underlying operating system that Python is running on, i.e. Ubuntu. These functions include creating, renaming, and deleting directories, fetching files, and manipulating paths. In the context of this experiment setup, it was useful for splitting a path into its head and tail, to get the name of each file.

The pipeline of methods for this experiment setup is illustrated in figure 31, and all of them will be described right after.

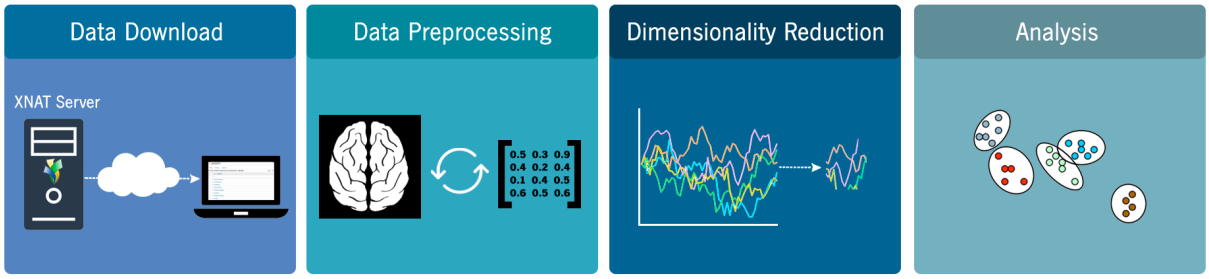


Figure 31: Methods for this experiment setup.

4.3.1 Data Download

When data is stored in a XNAT server, it must be downloaded to perform a series of transformations accordingly to what the experiment requires. This method enables the connection directly to the server where data is stored, and implements a set of functions based on the XNATUM package to extract the desired data. The *getSubjects* and *visualizeSubjects* functions work together to retrieve the list containing the IDs of all available subjects within a project. To download all sessions of a subject, the *getSessionsFromSubject* function takes as input the ID of the selected subject, and *checkDownloadedSessions* displays all sessions the user has downloaded until that moment. If the aim is to analyse imaging data from all subjects within a project, the *getAllProjectSessions* function is used to download all available sessions within the instantiated project. Both *getSessionsFromSubject* and *getAllProjectSessions* functions download data to a designated directory.

4.3.2 Data Preprocessing

After downloading all scans, the *searchData* function enables confirming the files within the destination directory, which was defined in the downloading functions. It recursively searches for all files matching the scan file format, i.e. NIfTI, and returns a list containing all paths and filenames. To retrieve how many files are read and splitting the path and name for each file, the *readScans* function was developed.

Loading and transforming imaging data are fundamental steps to make such data comprehensible to ML algorithms. Hence, all scans must be imported as a multidimensional array of numbers. The *loadScan* function loads a brain scan within a NIfTi file, and extract its data to a NumPy array, giving as argument its path, and the subject's ID. To confirm a scan's dimensions, the *checkScanShape* function was implemented as it retrieves the scan's shape. Inspecting slices of a brain volume is a good practice to check how they are being represented. For the purpose, the *visualizeBrain* function plots slices of a scan for the selected instant.

The region of interest of each MRI scan is the brain, and everything else is considered noise. An approach to isolate the cerebral tissue is selecting voxels confined within the brain area, using a binary brain mask. This mask is generated by assigning all non-zero voxels of a normalized brain scan to the value 1, which is known as binarization. This process builds a brain mask that perfectly delimits and fills in the brain area of all standardized scans, and it can be used to retrieve the coordinates for all brain voxels. Firstly, one must load the mask into a numpy array, using the function *loadMask*, which has a similar arrangement to *loadScan*. The shape of the mask is returned using *checkMaskShape*, and it determines the range of points for each axis. All dimensions of the mask are iterated using a function named *coordsArray*, which adds in a list the x, y, z coordinates for each point where the value met is 1. The length of this list reveals how many voxels are confined within the brain, and for a simpler representation, each set of x, y, y, is assigned to a number from 1 to n. For the purpose, the *checkCoods* function was developed to return the set of coordinates given a point of choice within the brain.

The activation of each voxel changes over time as it depends on the neural activity with respect to the brain region the voxel is within. To analyse such variations across several brain areas and individuals, as well as comprising multiple examples, data must be represented by individual. A suggestion is to build a structure per subject denoting the activation values for each coordinate over time, based on observations of the subject's brain scans. The *createNumpyArray* function creates this structure by defining a NumPy array of zeros with a total of elements equal to the obtained quantity of voxels within the brain $\times$ the number of brain volumes acquired per subject. The zeros of the structure are replaced with the BOLD activation value observed for each set of coordinates, and the final structure is saved in a *.npy* file to a designated directory. This process is implemented by the function *saveNumpyArrayPerSubject*, and repeated for each subject.

Time-courses are typically used to describe activity of neurons within voxels, making it easier to detect variations, and helping in the further step of analysing brain connectivity. The function *plotTimeCoursePerSubject* generates a plot for each subject depicting BOLD fluctuations across multiple

locations of their brain over time. The number of locations and time interval to display can be defined by the user in the input of this function.

An additional function, namely *checkMinMax*, was developed to verify both minimum and maximum values observed on each NumPy array. These values are used as reference to normalize data within the interval [0,1], through the *MixMaxNormalization* function.

4.3.3 Dimensionality Reduction

Sample size in fMRI studies is typically around the order of ten or hundreds, making it harder to find patterns from original dimensional data [74]. To fix this, dimensionality reduction techniques come forward. Traditional approaches to analyse FC usually foresee an atlasing step, in which the brain is parcellated into specific regions of interest, before assessing connectivity [177]. Although efficient, brain parcellation is not suitable for the purpose of this dissertation. Reasons include exploring FC at region level, where voxel's time-courses are spatially averaged, and existing a wide variety of atlases which is itself an indicator of having no consensus in the research community on how to properly parcellate all brain regions. PCA, as a data-driven technique, overcomes the problem of parcellation, but still requires setting critical parameters beforehand, such as the number of components to estimate, influencing the resulting network configurations [177]. In addition, PCA is a linear transformation, and it has been described as performing poorly compared to non-linear dimensionality reduction techniques [18]. The remaining techniques studied, i.e. autoencoder and t-SNE, are non-linear, adapting better to data, and thus both are proposed for this experiment setup.

Autoencoder

A typical autoencoder comprises an encoder, and a decoder, and its development begins with modelling these two components. To model the encoder, the *createEncoder* function was created. This function takes as input a NumPy array returned by the *MinMaxNormalization* function from the previous method, and multiple layers are added to structure the encoder. These layers are used to reduce the dimensionality of the input data, until reaching the latent space vector, whose dimension is defined by the user. The *createDecoder* function models the decoder, taking as input the latent space vector of the *createEncoder* function. Layers are then added to build the decoder, and output data with increasingly higher dimension. To create the autoencoder model, the *createAE* function was defined. This function calls the *createEncoder* and *createDecoder* to create the encoder and decoder respectively, and defines the latent space vector as being the output of the *createEncoder* function. The output of the autoencoder

results from feeding the latent space vector to the *createDecoder* function. The autoencoder then tries to model the input data to the output.

The model is compiled using RMSprop as optimizer, and MSE as loss function, through the *compileModel* function. Aside from defining the optimizer and loss function, parameters such as number the epochs, the batch size, callbacks are defined in this area. The number of epochs refers to how many times the training samples are brought back and forward while training. The batch size defines the number of samples given to the model at each time. For instance, larger batch sizes results in requiring more memory space to accommodate the data. The model performance is influenced by these parameters, and therefore, changing their values may produce improvements in the results. The callback included was ModelCheckPoint, which is responsible for saving the best weights of the model's training in a *.h5* format from the h5py[6] package. This file is then saved in the disk for further reference.

After training the model, it must be evaluated to understand its performance. Accuracy is typically chosen as evaluation metric as it retrieves the percentage of correctly predicted outputs. Despite its concept, this is not a suitable metric for evaluating the performance of this model, since accuracy does not understand the values it is dealing with in context of this problem. Chances are the activation value of a voxel will not be the same in the input and the output, as the autoencoder produces a close but not perfect representation of the input. Accuracy analyses both expected and predicted values and compares them, and outputs if the value was correct or not, regardless of close values still being accepted as a good prediction. Having this said, this model performance is measured based on the model's loss during the training phase. MSE is the selected loss function, and for values close to 0, it means the output is a very similar representation to the input. There is no validation phase, as the target is not to produce a generalizing model, but rather minimizing the error, and having a model capable of representing data properly.

t-SNE

Brain voxels comprised in the same brain region are expected to have similar time-courses of activation. The t-SNE technique is proposed to display data points in the 2D plane, in order to find aggregations of data points.

Firstly, the *selectData* function accepts as input a NumPy array comprised of spatial and temporal data and loads it. Data portions of interest may be selected by picking an interval for spatial data, and also for temporal data of the array. Within each interval, interleaved data points can be chosen to narrow the

[6] h5py is a Pythonic interface to the HDF5 (Hierarchical Data Format) binary data format.

quantity of data points to compute.

Defining and setting parameters of the t-SNE algorithm is done in the *runTSNE* function. Parameters include perplexity, early exaggeration, learning rate, and number of iterations, and all of them must be set by the user. The perplexity refers to the number of nearest neighbors, and values should be comprised between 5 and 50. For large datasets, a large perplexity is typically required. The early exaggeration parameter controls the space between natural clusters in the embedded space. For instance, a higher value of early exaggeration will translate to natural clusters being more apart from each other. As for the learning rate, values should vary between 10 and 1000. Low learning rates result in a dense cloud with few outliers, whereas high learning rates make any data point approximately equidistant from its nearest neighbours. The number of iterations corresponds to the maximum number of iterations for the optimization, and it should be at least 250.

At last, the *visualizeData* function enables inspecting the distribution of data in the 2D plane.

4.3.4 Analysis

Data is now ready to be analysed using traditional algorithms for pattern recognition in order to identify clusters of functionally coupled regions in the brain. Studies using unsupervised learning techniques have reported meaningful variations of resting state FC during the course of a resting state acquisition [74]. The configuration of brain networks and their rearrangement over time is explored through dynamic FC (dFC), and it can be expressed using different approaches [177].

The most common scenario uses sliding-windows correlations, in which time-courses are divided in successive, partially overlapping windows [74, 177]. FC is then estimated for each of these temporal windows, and a sequence of connectivity matrices are yielded (one for each window). To identify patterns in this sequence, clustering techniques are used [74]. For instance, multiple studies have found recurring dFC patterns, also known as "states", through k-means clustering of windowed connectivity matrices [74, 178]. However, there are some limitations regarding the use of sliding windows, for instance choosing the window size, or its shape [179].

An alternative scenario to estimate time-varying FC involves modeling the rsfMRI time-series directly, using Hidden Markov Models (HMMs) [74, 180]. HMMs are unsupervised learning methods for sequential data that assume time-series data can be described using a sequence of finite number of states (figure 32). These "states" represent unique brain networks of distinct FC which repeat at different points in time. HMMs estimate a number of states shared by all subjects, as well as a specific state time-course for each subject, expressing when each state is active [181]. Each state is defined by its mean activation, and

connectivity matrix [181]. To describe a fMRI time-series using an HMM, the following parameters must be defined beforehand: number of observations (i.e. time samples), number of states, state-transition probability distribution, observation probability distribution for each state, and an initial state distribution [182]. Studies adopting HMMs have revealed that transitions between states occur as a hierarchically organized sequence [74, 181]. Moreover, a recent study demonstrated HMMs were able to distinguish MCI patients from controls [183]. HMMs appear to be a promising technique to identify patterns of temporal organization or reconfigurations in RSNs. For that reason, it is proposed as pattern recognition technique for the last method of this experiment setup, and ought to be implemented in a future work.

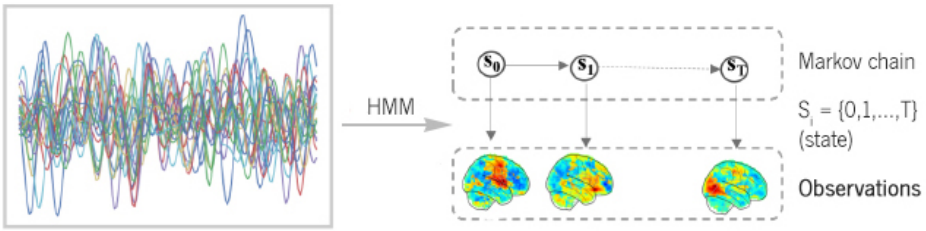


Figure 32: HMM models time-series data as a sequence of finite number of states. Adapted from [74].

4.3.5 Case Study Results

The performance of the previous methods was tested using the In-House data, as it concerns a smaller number of cases. The main results regarding each method are presented below.

Data Download

In-House data is stored within a XNAT project whose ID is *ConnectBrain*, and it includes both data and metadata. The *metadata.cvs* file describes each subject in terms of gender (male or female), year of birth, and number of MR sessions (undefined, 1 or 2). All participants are identified with a corresponding ID (e.g. *SW0033C*, *SW0045C*, *SW3944C*) to preserve their privacy and data. Despite having a total of 120 subjects, 13 of them do not have any assigned scan file. Data is comprised of a total of 183 scans, in which 107 concern the first MR session (labeled as REST1), and the remaining 76 were acquired in the second MR session a year after (labeled as REST2).

After defining *sessions* as the target directory in the *getAllProjectSessions*, all 183 scans were downloaded and arranged in folders, creating a structure based on XNAT's. This arrangement is illustrated in figure 33, and it displays the generated directories. A set of folders is created for each MR session, identifying the subject and the MR session (REST1 or REST2). T1 and T2 labeled folders give information on the scan type. All scan files were in a *.gz* compressed format.

Figure 33: Folder organization based on the XNAT data structure for the In-House data.

Data Preprocessing

To exemplify results obtained within this method, solely the first subject from the list will be considered. After loading and extracting data from its brain scan to a NumPy array using the *loadScan*, the *checkScanShape* retrieved a dimension of 91 $\times$ 109 $\times$ 91 $\times$ 175 (first three for the x, y, and z axis, and last dimension for instant t). The *visualizeBrain* function displays the brain in interleaved slices for the selected point in time of the acquisition. For instance, figure 34a illustrates the first brain volume, i.e. t = 0, and depicts contrast in the intensities between tissues. The brain mask loaded echoes the MNI152 standard space, and it has a shape (91, 109, 91) as given by the *checkMaskShape* function. The *visualizeMask* function produced the result depicted in figure 34b, and opposed to what was observed in figure 34a, no features are observed inside the brain mask.

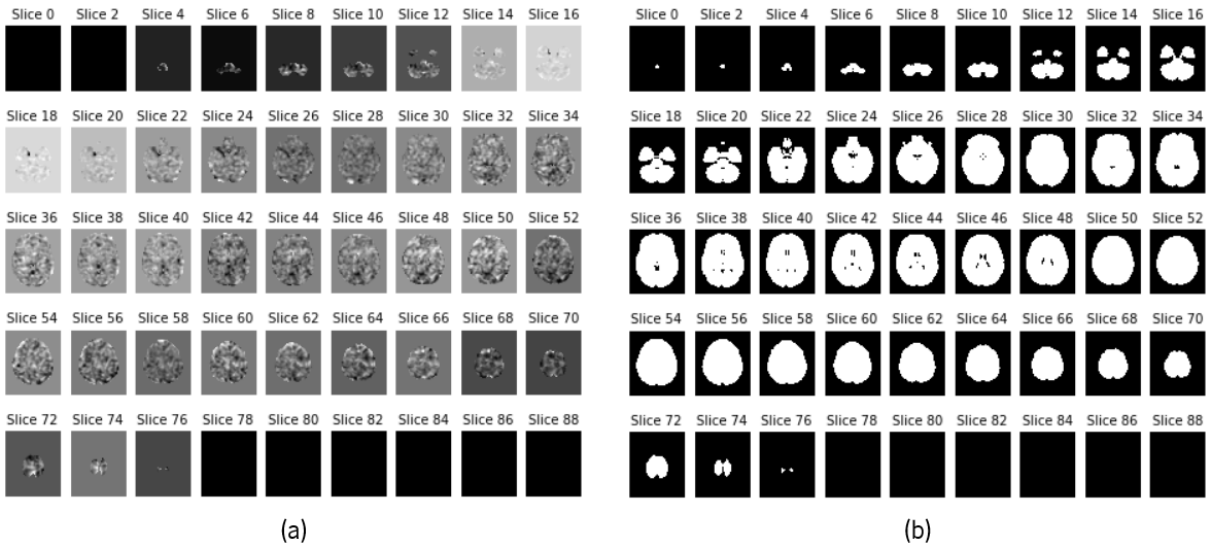


Figure 34: (a) Brain displayed in slices for a subject, at t = 0, using the *visualizeBrain* function; (b) brain mask displayed in slices using the *visualizeMask* function.

Using the *coordsArray* followed by the *checkCoords*, the value 228483 was retrieved as being the total of voxels captured within the brain. As a result, the NumPy array of zeros created for each subject using the *createNumpyArray* was set to a dimension of 228483 $\times$ 175, wherein the first value denotes the number of voxels, and the second value depicts the number of slices per scan. After filling in the array of zeros with the BOLD activation values of the voxels within the subject's brain area through the *saveNumpyArrayPerSubject*, time-course plots could be generated. The plot depicted in figure 35a considers the first 5 voxels of the array, whereas the plot illustrated in figure 35b considers voxels interleaved with a value of 50000. Each color represents the activity of a voxel, and although both plots have the same cardinality (dimension of 175 $\times$ 5), and represent the same subject, the BOLD effect behaves differently. The variation of activation levels among contiguous areas is almost imperceptible, while distant areas display very different BOLD signals. This may be explained by the voxels selected for the plot on the left perhaps belonging to the same brain region, thus behaving similarly, whereas the voxels selected for the plot on the right might be representative of distinct brain regions, and therefore display very unlike behaviours.

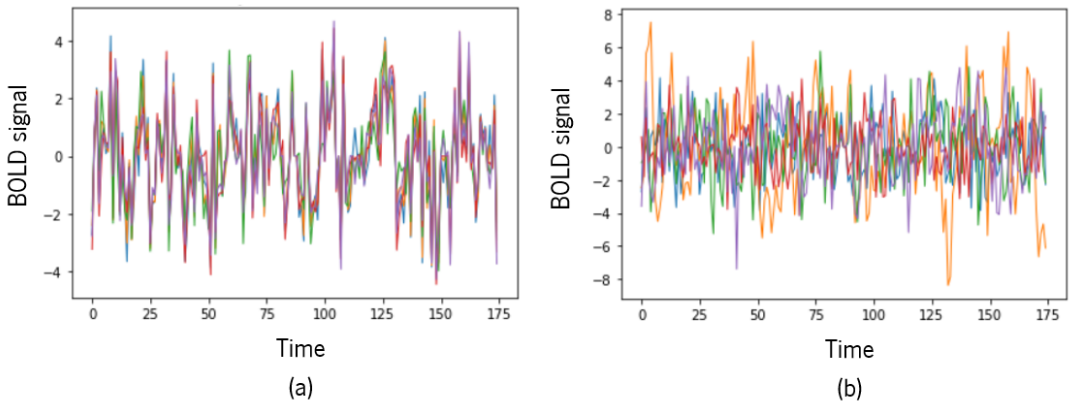


Figure 35: (a) Time-courses plot displaying variations of the BOLD signal for contiguous voxels; (b) time-courses plot displaying variations of the BOLD signal for distant voxels.

Dimensionality Reduction

The architecture of the autoencoder tested for the In-House data will be firstly depicted in its two parts, encoder and decoder models, to go over each layer and justify the choices made.

The encoder is structured in a total of eight layers, and each layer is described in table 3, along their output shapes, and number of trainable parameters. The first layer (Input Layer) receives as input an array with size 228483 $\times$ 175. A reshape is performed to include the channel dimension, which in this case is 1, as In-House data are grayscale images. The shape is (228483, 175, 1) at that point (i.e. a

39984525-dimensional array), and for that reason, following layers will essentially aim to reduce initial dimension to a much smaller latent space. The MaxPooling2D layer uses a pooling window size of 2 $\times$ 2, and a stride of 2, reducing both spatial and temporal data by a half. The output shape is (114242, 88, 1), and although much smaller than the original input, there is still room for improvement. To remove the channel dimension, needed for a further operation, a reshape is used again. The MaxPooling1D layer uses a pooling window size of 8, and a stride of also 8, to reduce spatial data by 1/8, and the number of voxel locations went from 114242 to 14281. Both MaxPooling layers include zero padding, so that border elements are covered and computed more. A flatten layer, as the name suggests in used to flatten the input tensor to a shape equal to the number of elements contained in the tensor. All layers described until this point do not have learning properties, but rather operate feature reduction, hence zero parameters for all of them. Data is prepared to be fed to a Dense layer (i.e. MLP), to drastically reduce the dimension of the input from 1256728 to 4 units, using ReLU as activation function, since no negative values are present. There is a total of 5026916 trainable parameters within the first MLP. At last, a second and much simple Dense layer is used to output the latent space of the autoencoder with size 2, with 10 trainable parameters.

The decoder tries to symmetrically reproduce the encoder to build the output from the latent space to be as close to the input as possible. Hence, the decoder is structured in a total of nine layers, as displayed in table 4. The Input Layer takes the latent space with size 2, and a Dense layer follows to raise the dimension from 2 to 4, making a total a 12 parameters to train. Another Dense layer proceeds to increase greatly the input, to a output dimension of 1256728, requiring the network to train 6283640 parameters. Both Dense layers use ReLU for activation function, giving its properties. To decompress the input, a reshape is performed transforming the 1D vector of 1256728 elements to a matrix with dimension 14281 $\times$ 88. The UpSampling1D layer uses a sampling factor of 8, to repeat each spatial step 8 times along the axis to increase the number of voxel locations, and outputs a dimension of 14248 $\times$ 88. Another reshape is used to include the channel dimension (1 again), with a output shape of (114248, 88, 1). The UpSampling2D layer has a similar behaviour to the previous upsampling layer, as both scale up the given input with a nearest neighbor algorithm for interpolation. Yet UpSampling2D repeats the rows and columns of data 2 times each due to the chosen factor being 2, and not just one of the dimensions as UpSampling1D does. At this point, data has a shape (228496, 176, 1), which is very close to In-House data's original size. Hence, an adjustment is done with the Cropping2D layer using a tuple of two tuples, i.e. (7, 6), (1, 0), to crop 7 rows on top, and 6 bottom rows for the spatial axis, and 1 row on the left and no rows on the right for the temporal axis. After a reshape to remove the channel dimension, the output

shape of the decoder is now the same as the encoder's input, i.e. (228483, 175). Aside from the two Dense layers, none of the layers of the decoder have trainable parameters, as they just intend to transform the dimensions from the input to the output.

Table 3: The layers of the encoder.

Layer Type	Output Shape	Parameters
Input Layer	(None, 228483, 175)	0
Reshape	(None, 228483, 175, 1)	0
MaxPooling2D	(None, 114242, 88, 1)	0
Reshape	(None, 114242, 88)	0
MaxPooling1D	(None, 14281, 88)	0
Flatten	(None, 1256728)	0
Dense	(None, 4)	5026916
Dense	(None, 2)	10

Table 4: The layers of the decoder.

Layer Type	Output Shape	Parameters
Input Layer	(None, 2)	0
Dense	(None, 4)	12
Dense	(None, 1256728)	6283640
Reshape	(None, 14281, 88)	0
UpSampling1D	(None, 114248, 88)	0
Reshape	(None, 114248, 88, 1)	0
UpSampling2D	(None, 228496, 176, 1)	0
Cropping2D	(None, 228483, 175, 1)	0
Reshape	(None, 228483, 175)	0

The summarized autoencoder is depicted in table 5, comprising its input layer, encoder and decoder models, their output shape, as well as the number of trainable parameters. Although data dimensions were highly reduced, this model still had an overwhelming number of 11,310,578 parameters to train. Several tries were made regarding on how to handle and process such high-dimensional data with the autoencoder, but due to the machine's memory constraints, solely the present architecture provided results. For instance, the lowest MSE error obtained was 0.09431 (i.e. an absolute error of 0.3071). This is a rather acceptable error, considering the aim is not to reproduce exactly the input, but rather a close representation.

Table 5: The autoencoder model.

Layer	Output Shape	Parameters
Input Layer	(None, 228483, 175)	0
Encoder (Model)	(None, 2)	5026926
Decoder (Model)	(None, 228483, 175)	6283640
Total parameters: 11,310,578		

The t-SNE algorithm was also tested for the In-House data, and results are described below.

The result of the computation of t-SNE for the first 10,000 coordinates, and first 10 brain volumes using the *visualizeData* function is illustrated in figure 36. The parameters set were the following: iterations = 5000, perplexity = 50, learning rate = 700, and early exaggeration = 30. A few clusters can be identified, and they were circled. Each cluster might represent voxels comprising a certain brain region, and given their similaries, t-SNE groupped them.

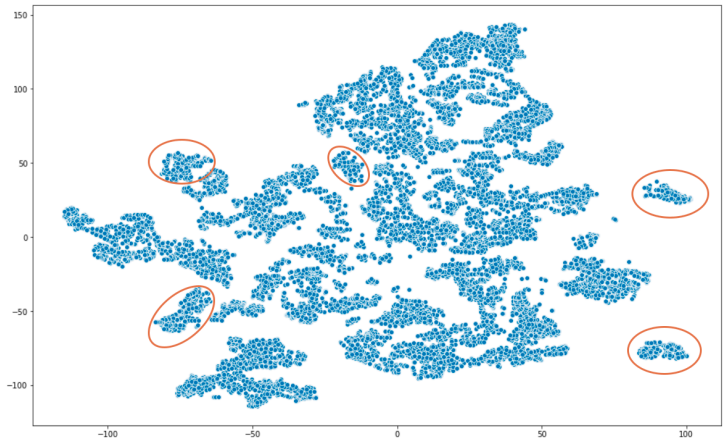


Figure 36: t-SNE algorithm computation for the first 10,000 coordinates, and first 10 brain volumes.

Using the same parameters, t-SNE was tested for a total of 2285 coordinates with an interval of 100 coordinates between each two, and the first 50 brain volumes. The result, illustrated in figure 37, is similar to a Gaussian distribution, and no clusters are observed. This might be due to having selected a group of coordinates that do not represent the same brain region. This does makes sense considering coordinates were picked one every 100 coordinates, and no coherent behaviour has been found.

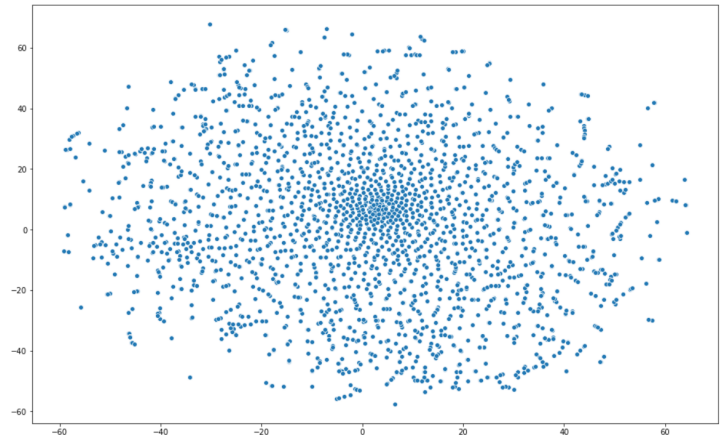


Figure 37: t-SNE algorithm computation for one every 100 coordinates, and the first 50 brain volumes.

To observe how the learning rate affects the computation, all parameters where kept the same, except for learning rate. It was expected that a lower learning rate would result in a denser cloud, but instead, data points became more disperse. This might be justified by the fact that high-dimensional data is being visualized. When having too many features, observations are harder to cluster, as too many dimensions cause every observation to appear equidistant from all the others. Figure 38 illustrates the computation with the same parameters as previous computations, but with a learning rate of 10. Except for the very center, no meaningful clusters appear to be formed.

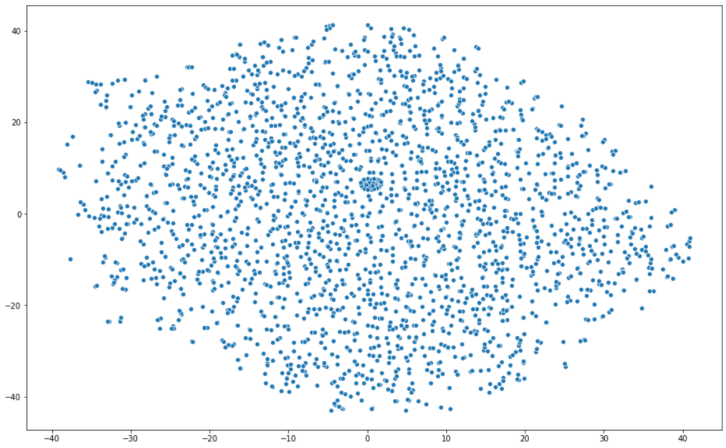


Figure 38: t-SNE algorithm computation for one every 100 coordinates, and the first 50 brain volumes, with a lower learning rate.

5 Conclusions and Future Work

5.1 Conclusions

The main goal of this dissertation was designing and developing an experiment setup to analyse resting-state functional connectivity at a voxel level, in order to find functional patterns. To achieve this, the work developed through out this dissertation comprised several smaller stages.

Initially, it was required an in-depth domain knowledge on the functional human brain, and techniques used to access it. This was fundamental to understand the typical workflow from acquisition, to analysis of functional connectivity, and outline a pipeline for this experiment setup. Next, resting-state fMRI data was gathered, and potential tools for the working environment were investigated. The result was a configured working station to start experimenting with data. This led to explore ways of transforming data, using methods from several Python packages, such as NiBabel, NumPy, matplotlib, and scikit-learn, and aid enhancing knowledge on Python. In addition, the integration of Jupyter Notebook and Docker containers provided real-time script editing, running, and protection, as well as user-friendly visualization of results. To reduce the dimensionality of the already preprocessed data, an autoencoder was designed with building blocks imported from the Keras library. These included several layer options, optimizers, loss functions, and other interesting features. Tensorflow and Keras have proved to be a powerful combination to quickly developing and training ML and DL models. An additional dimensionality reduction technique was studied, t-SNE, which was particularly interesting for data visualization.

The previously depicted stages enabled to pipe this experiment setup, resulting in four methods: Data Download, Data Preprocessing, Dimensionality Reduction, and Analysis. Each one of these methods comprised functions to support their process from input to output. The exception is Analysis, which was based on exploring rather than developing. To evaluate the performance of this experiment setup, a case study was performed using the In-House data for concerning a smaller number of subjects to study. Data was successfully downloaded from a XNAT server using the Data Download method, and Data Preprocessing provided methods to transform data. Using the autoencoder strategy on the

Dimensionality Reduction method, the best obtained Mean Squared Error was 0.094. However, given the drastic reduction in data, no clusters could be identified. The alternative technique, t-SNE, allowed to represent data into a 2D plane, and some clusters were observed, despite the high dimension of data. Limitations concerning the memory of the machine used restricted some aspects of this experiment setup's testing. For instance, HCP data did exhibit results during Data Download, and Data Preprocessing, but failed to proceed, as the machine would not have the capacity to allocate such data.

Overall, the pipeline was successful at delivering results. With appropriate resources, this experiment setup may support the process of analysing and extracting patterns from any resting state FC data, and aid in the detection of mental disorders.

5.2 Future Work

There are some aspects of this experiment setup which could be enhanced in a future work.

Preprocessing is an essential step to prepare data, and may determine the validity of future results. Hence, several techniques can be tried out for improvement. The normalization selected for this experiment setup enables activation values to range from 0 to 1. A different interval could be used to normalize the values, for instance, between -1 and 1, since activation values can be positive or negative. This might be interesting to further compare how these two normalization approaches influence the results. In this experiment setup, each case was considered independent from each other, and normalization was performed at subject level. Although this is a suitable approach for this dissertation's goal, if the aim is to place subjects at a same level of comparison, normalization should be done at a dataset level.

The primary aspect to improve would be using materials not to reduce data as much, and to make the analysis computationally possible for any dataset without using brain parcellation or common statistical methods. In-House data comprised 39,984,525 data points per subject, and it was reduced more than 95% just with MaxPooling layers in the autoencoder technique due to the machine's memory constraints. This might have severely affected crucial features present in data. Perhaps the autoencoder model would have benefited from using convolutional, and deconvolutional layers as they search for features that best represent data, instead of just using basic maxpooling and upsampling layers. If more features were preserved, clustering techniques could have been used on the resulting latent space vector to find patterns, instead of relying on data visualization methods. Additionally, the Analysis method would greatly benefit from the implementation of a Hidden Markov Model to find functional patterns across brain regions.

Fixing the dimensionality issue would open doors for experimenting with larger datasets. For instance,

HCP data could be explored to its full potential, as it comprises more subjects to study, a much larger number of brain volumes per subject, and a better data quality when compared to In-House data. Moreover, an interesting addition to this experiment setup would be introducing resting state fMRI data from a group of subjects with mental disorders. This would contribute to explore FC patterns on subjects with mental disorders *versus* healthy controls, and possibly validate literature findings concerning the one complex integrative system known as the brain.

References

[1] Morris F Collen. Origins of medical informatics. *Western Journal of Medicine*, 145(6):778, 1986.

[2] William R Hersh. Medical informatics: improving health care through information. *Jama*, 288(16):1955–1958, 2002.

[3] Stefanie Gehring and René Eulenfeld. German medical informatics initiative: unlocking data for research and health care. *Methods of information in medicine*, 57(S 01):e46–e49, 2018.

[4] Reinhold Haux. Medical informatics: past, present, future. *International journal of medical informatics*, 79(9):599–610, 2010.

[5] Marjan Laal. Innovation process in medical imaging. *Procedia-Social and Behavioral Sciences*, 81:60–64, 2013.

[6] Siemens Healthineers Global. Medical imaging. https://www.healthcare.siemens.com/medical-imaging. [Online. Accessed: 2018-10-12].

[7] Michael Brammer. The role of neuroimaging in diagnosis and personalized medicine – current position and likely future directions. *Dialogues in Clinical Neuroscience*, 11(4):389, 2009.

[8] M Symms, HR Jäger, K Schmierer, and TA Yousry. A review of structural magnetic resonance neuroimaging. *Journal of Neurology, Neurosurgery & Psychiatry*, 75(9):1235–1244, 2004.

[9] Martin A Lindquist and Tor D Wager. Principles of functional magnetic resonance imaging. *Handbook of Neuroimaging Data Analysis*, 2014.

[10] Michel A Hofman. Evolution of the human brain: when bigger is better. *Frontiers in neuroanatomy*, 8:15, 2014.

[11] Martijn P Van Den Heuvel and Hilleke E Hulshoff Pol. Exploring the brain network: a review on resting-state fmri functional connectivity. *European neuropsychopharmacology*, 20(8):519–534, 2010.

[12] Linda Geerligs, Mikail Rubinov, Richard N Henson, et al. State and trait components of

functional connectivity: individual differences vary with mental state. *Journal of Neuroscience*, 35(41):13949–13961, 2015.

[13] Elmar Wolfgang Lang, Ana Maria Tomé, Ingo R Keck, JM Górriz-Sáez, and Carlos García Puntonet. Brain connectivity analysis: a short survey. *Computational intelligence and neuroscience*, 2012:8, 2012.

[14] Baxter P Rogers, Victoria L Morgan, Allen T Newton, and John C Gore. Assessing functional connectivity in the human brain by fmri. *Magnetic resonance imaging*, 25(10):1347–1357, 2007.

[15] David D Cox and Robert L Savoy. Functional magnetic resonance imaging (fmri) "brain reading": detecting and classifying distributed patterns of fmri activity in human visual cortex. *Neuroimage*, 19(2):261–270, 2003.

[16] Hayit Greenspan, Bram Van Ginneken, and Ronald M Summers. 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, 2016.

[17] Bradley J Erickson, Panagiotis Korfiatis, Zeynettin Akkus, and Timothy L Kline. Machine learning for medical imaging. *Radiographics*, 37(2):505–515, 2017.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[19] V Soares, J Dos Santos, and M Naldi. Visualization in big data: a tool for pattern recognition in data stream. *Revista de Sistemas de Informacao da FSMA*, 15:30–39, 2015.

[20] Maria Fayez, Soha Safwat, and Ehab Hassanein. Comparative study of clustering medical images. In *2016 SAI Computing Conference (SAI)*, pages 312–318. IEEE, 2016.

[21] Tableau. Data visualization beginner's guide: a definition, examples, and learning resources. https://www.tableau.com/learn/articles/data-visualization. [Online. Accessed: 2018-10-17].

[22] Nicolas Kruchten. Data visualization for artificial intelligence, and vice versa. https://medium.com/@plotlygraphs/data-visualization-for-artificial-intelligence-and-vice-versa, 2018. [Online. Accessed: 2018-10-17].

[23] SAS. Data visualization for artificial intelligence, and vice versa. https://www.sas.com/en_us/insights/big-data/data-visualization. [Online. Accessed: 2018-10-17].

[24] Shelby Blitz. How machine learning improves data visualization," sisense. https://www.sisense.com/blog/how-machine-learning-improves-data-visualization/, 2018. [Online. Accessed: 2018-10-18].

[25] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information*

systems, 24(3):45–77, 2007.

[26] John Nolte. *Essentials of the human brain e-book: With student consult online access*. Elsevier Health Sciences, 2009.

[27] Radu Mutihac. Essentials in brain connectivity. *Journal of Neurology and Clinical Neuroscience*, 1(1), 2017.

[28] Institute for Quality and Efficiency in Health Care (IQWiG). How does the brain work? https://www.ncbi.nlm.nih.gov/books/NBK279302/. [Online. Accessed: 2019-10-22].

[29] Karla Batista-García-Ramó and Caridad Ivette Fernández-Verdecia. What we know about the brain structure–function relationship. *Behavioral Sciences*, 8(4):39, 2018.

[30] Qawi K Telesford, Sean L Simpson, Jonathan H Burdette, Satoru Hayasaka, and Paul J Laurienti. The brain as a complex system: using network science as a tool for understanding the brain. *Brain connectivity*, 1(4):295–308, 2011.

[31] Lorraine K Tyler. The resilient brain: Cognition and ageing. https://www.thebritishacademy.ac.uk/sites/default/files/Resilient%20Brain%20Tyler.pdf, 2011. [Online. Accessed: 2019-10-24].

[32] Olaf Sporns. Structure and function of complex brain networks. *Dialogues in clinical neuroscience*, 15(3):247, 2013.

[33] Oxford Reference. Neuronal pools. https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100230228. [Online. Accessed: 2019-10-22].

[34] Paulo Marques. *A network approach to brain aging through multimodal neuroimaging*. PhD thesis, 2015.

[35] Aina Frau-Pascual, Morgan Fogarty, Bruce Fischl, Anastasia Yendiki, Iman Aganj, Alzheimer's Disease Neuroimaging Initiative, et al. Quantification of structural brain connectivity via a conductance model. *Neuroimage*, 189:485–496, 2019.

[36] Alex Fornito, Andrew Zalesky, and Edward Bullmore. *Fundamentals of brain network analysis*. Academic Press, 2016.

[37] Gaetano Cariello and Sebastiano Stramaglia. Deep learning for pattern recognition on resting state fmri data. 2017.

[38] Silvia A Bunge and Itamar Kahn. Cognition: An overview of neuroimaging techniques. 2009.

[39] Geoffrey K Aguirre. Functional neuroimaging: technical, logical, and social perspectives. *Hastings Center Report*, 44(s2):S8–S18, 2014.

[40] Richard DeCharms. Methods for measurement and analysis of brain activity, August 16 2007. US Patent App. 11/738,967.

[41] Patrick W Stroman. *Essentials of functional MRI*. CRC Press, 2016.

[42] Jingyuan E Chen and Gary H Glover. Functional magnetic resonance imaging methods. *Neuropsychology review*, 25(3):289–313, 2015.

[43] Edson Amaro Jr and Gareth J Barker. Study design in fmri: basic principles. *Brain and cognition*, 60(3):220–232, 2006.

[44] Richard Bitar, General Leung, Richard Perng, Sameh Tadros, Alan R Moody, Josee Sarrazin, Caitlin McGregor, Monique Christakis, Sean Symons, Andrew Nelson, et al. Mr pulse sequences: what every radiologist wants to know but is afraid to ask. *Radiographics*, 26(2):513–537, 2006.

[45] Robert W Brown, Y-C Norman Cheng, E Mark Haacke, Michael R Thompson, and Ramesh Venkatesan. *Magnetic resonance imaging: physical principles and sequence design*. John Wiley & Sons, 2014.

[46] David C Preston. Magnetic resonance imaging (mri) of the brain and spine: Basics. https://casemed.case.edu/clerkships/neurology/Web%20Neurorad/MRI%20Basics.htm, 2006. [Online. Accessed: 2019-10-24].

[47] John R Hesselink. Basic principles of mr imaging. *Clinical magnetic resonance imaging, 2nd ed. Philadelphia: WB Saunders Company*, 1996.

[48] Gary H Glover. Overview of functional magnetic resonance imaging. *Neurosurgery Clinics*, 22(2):133–139, 2011.

[49] O Gosseries, A Demertzi, Q Noirhomme, J Tshibanda, M Boly, M de Beeck Op, R Hustinx, P Maquet, E Salmon, G Moonen, et al. Functional neuroimaging (fmri, pet and meg): what do we measure? *Revue médicale de Liège*, 63(5-6):231–237, 2008.

[50] Patrícia R Nunes, Sandra R Tecelão, and Rita G Nunes. Ressonância magnética funcional: mapeamento do córtex motor através do efeito bold. *Saúde & Tecnologia*, pages 11–18, 2015.

[51] Dustin Stansbury. fmri in neuroscience: The basics. https://theclevermachine.wordpress.com/2012/11/23/fmri-in-neuroscience-the-basics/, 2012. [Online. Accessed: 2019-10-15].

[52] Ricardo José Silva Magalhães. *Modelação de padrões de conectividade cerebral funcional*. PhD thesis, 2013.

[53] Robert R Edelman, Piotr Wielopolski, and Franz Schmitt. Echo-planar mr imaging. *Radiology*, 192(3):600–612, 1994.

[54] Mehdi Poustchi-Amin, Scott A Mirowitz, Jeffrey J Brown, Robert C McKinstry, and Tao Li. Principles and applications of echo-planar imaging: a review for the general radiologist. *Radiographics*, 21(3):767–779, 2001.

[55] Martha Skup. Longitudinal fmri analysis: A review of methods. *Statistics and its interface*, 3(2):235, 2010.

[56] Knowing Neurons. The brain's building blocks: Of protons and voxels. https://knowingneurons.com/2017/09/27/mri-voxels/. [Online. Accessed: 2019-10-29.

[57] José M Soares, Ricardo Magalhães, Pedro S Moreira, Alexandre Sousa, Edward Ganz, Adriana Sampaio, Victor Alves, Paulo Marques, and Nuno Sousa. A hitchhiker's guide to functional magnetic resonance imaging. *Frontiers in neuroscience*, 10:515, 2016.

[58] Stephen C Strother. Evaluating fmri preprocessing pipelines. *IEEE Engineering in Medicine and Biology Magazine*, 25(2):27–41, 2006.

[59] Xiangrui Li, Paul S Morgan, John Ashburner, Jolinda Smith, and Christopher Rorden. The first step for neuroimaging data analysis: Dicom to nifti conversion. *Journal of neuroscience methods*, 264:47–56, 2016.

[60] Hussain A Jaber, Hadeel K Aljobouri, İlyas Çankaya, Orhan M Koçak, and Oktay Algin. Preparing fmri data for postprocessing: Conversion modalities, preprocessing pipeline, and parametric and nonparametric approaches. *IEEE Access*, 7:122864–122877, 2019.

[61] David B Parker and Qolamreza R Razlighi. The benefit of slice timing correction in common fmri preprocessing pipelines. *Frontiers in neuroscience*, 13:821, 2019.

[62] P Kalavathi and VB Surya Prasath. Methods on skull stripping of mri head scan images—a review. *Journal of digital imaging*, 29(3):365–379, 2016.

[63] Stuart Clare. Functional mri: methods and applications. *University of Nottingham*, page 155, 1997.

[64] Jija S James, PG Rajesh, Anuvitha VS Chandran, and Chandrasekharan Kesavadas. fmri paradigm designing and post-processing tools. *The Indian journal of radiology & imaging*, 24(1):13, 2014.

[65] Kai Lawonn, Noeska N Smit, Katja Bühler, and Bernhard Preim. A survey on multimodal medical data visualization. In *Computer Graphics Forum*, pages 413–438. Wiley Online Library, 2018.

[66] Juan E Arco, Carlos González-García, Paloma Díaz-Gutiérrez, Javier Ramírez, and Maria Ruz. Influence of activation pattern estimates and statistical significance tests in fmri decoding analysis. *Journal of Neuroscience Methods*, 308:248–260, 2018.

[67] Kaiming Li, Lei Guo, Jingxin Nie, Gang Li, and Tianming Liu. Review of methods for functional brain connectivity detection using fmri. *Computerized Medical Imaging and Graphics*, 33(2):131–139, 2009.

[68] Katell Mevel, Gaël Chételat, Francis Eustache, and Béatrice Desgranges. The default mode network in healthy aging and alzheimer's disease. *International Journal of Alzheimer's Disease*, 2011, 2011.

[69] Vince D Calhoun and Tülay Adali. Unmixing fmri with independent component analysis. *IEEE Engineering in Medicine and Biology Magazine*, 25(2):79–90, 2006.

[70] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570–584, 2017.

[71] Michael J. Garbade. Understanding k-means clustering in machine learning. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1, 2018. [Online. Accessed: 2019-11-23].

[72] Salam Al-Augby, Sebastian Majewski, Agnieszka Majewska, and Kesra Nermend. A comparison of k-means and fuzzy c-means clustering methods for a sample of gulf cooperation council stock markets. *Folia Oeconomica Stetinensia*, 14(2):19–36, 2014.

[73] Bertrand Thirion, Gaël Varoquaux, Elvis Dohmatob, and Jean-Baptiste Poline. Which fmri clustering gives good brain parcellations? *Frontiers in neuroscience*, 8:167, 2014.

[74] Meenakshi Khosla, Keith Jamison, Gia H Ngo, Amy Kuceyeski, and Mert R Sabuncu. Machine learning in resting-state fmri analysis. *Magnetic resonance imaging*, 2019.

[75] Brent C Munsell, Guorong Wu, Leonardo Bonilha, and Paul Laurienti. *Connectomics: Applications to Neuroimaging*. Academic Press, 2018.

[76] Yong Fan and Christos Davatzikos. Pattern recognition of functional brain networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6309–6313. IEEE, 2017.

[77] Hong-Ying Zhang, Shi-Jie Wang, Bin Liu, Zhan-Long Ma, Ming Yang, Zhi-Jun Zhang, and Gao-Jun Teng. Resting brain connectivity: changes during the progress of alzheimer disease. *Radiology*, 256(2):598–606, 2010.

[78] Christian Hohenfeld, Cornelius J Werner, and Kathrin Reetz. Resting-state connectivity in neurodegenerative disorders: Is there potential for an imaging biomarker? *NeuroImage: Clinical*, 18:849–870, 2018.

[79] Joseph Gaugler, Bryan James, Tricia Johnson, Allison Marin, and Jennifer Weuve. 2019 alzheimer's disease facts and figures. *Alzheimers & Dementia*, 15(3):321–387, 2019.

[80] Michael D Greicius, Gaurav Srivastava, Allan L Reiss, and Vinod Menon. Default-mode network activity distinguishes alzheimer's disease from healthy aging: evidence from functional mri. *Proceedings of the National Academy of Sciences*, 101(13):4637–4642, 2004.

[81] William Seunghyun Sohn, Kwangsun Yoo, Duk L Na, and Yong Jeong. Progressive changes in

hippocampal resting-state connectivity across cognitive impairment: a cross-sectional study from normal to alzheimer disease. *Alzheimer Disease & Associated Disorders*, 28(3):239–246, 2014.

[82] Masoud Tahmasian, Lorenzo Pasquini, Martin Scherr, Chun Meng, Stefan Förster, Satja Mulej Bratec, Kuangyu Shi, Igor Yakushev, Markus Schwaiger, Timo Grimmer, et al. The lower hippocampus global connectivity, the higher its local metabolism in alzheimer disease. *Neurology*, 84(19):1956–1963, 2015.

[83] Jessica S Damoiseaux, Katherine E Prater, Bruce L Miller, and Michael D Greicius. Functional connectivity tracks clinical deterioration in alzheimer's disease. *Neurobiology of aging*, 33(4):828–e19, 2012.

[84] Tommaso Gili, Mara Cercignani, Laura Serra, Roberta Perri, Federico Giove, Bruno Maraviglia, Carlo Caltagirone, and Marco Bozzali. Regional brain atrophy and functional disconnection across alzheimer's disease evolution. *Journal of Neurology, Neurosurgery & Psychiatry*, 82(1):58–66, 2011.

[85] Waldemar Karwowski, Farzad Vasheghani Farahani, and Nichole Lighthall. Application of graph theory for identifying connectivity patterns in human brain networks: A systematic review. *Frontiers in Neuroscience*, 13:585, 2019.

[86] Lonneke ML De Lau and Monique MB Breteler. Epidemiology of parkinson's disease. *The Lancet Neurology*, 5(6):525–535, 2006.

[87] Carl D Hacker, Joel S Perlmutter, Susan R Criswell, Beau M Ances, and Abraham Z Snyder. Resting state functional connectivity of the striatum in parkinson's disease. *Brain*, 135(12):3699–3711, 2012.

[88] Tao Wu, Xiangyu Long, Liang Wang, Mark Hallett, Yufeng Zang, Kuncheng Li, and Piu Chan. Functional connectivity of cortical motor areas in the resting state in parkinson's disease. *Human brain mapping*, 32(9):1443–1457, 2011.

[89] L Krajcovicova, M Mikl, R Marecek, and Irena Rektorova. The default mode network integrity in patients with parkinson's disease is levodopa equivalent dose-dependent. *Journal of neural transmission*, 119(4):443–454, 2012.

[90] Meghan C Campbell, Jonathan M Koller, Abraham Z Snyder, Chandana Buddhala, Paul T Kotzbauer, and Joel S Perlmutter. Csf proteins and resting-state functional connectivity in parkinson disease. *Neurology*, 84(24):2413–2421, 2015.

[91] Deepti Putcha, Robert S Ross, Alice Cronin-Golomb, Amy C Janes, and Chantal E Stern. Altered intrinsic functional coupling between core neurocognitive networks in parkinson's disease.

NeuroImage: Clinical, 7:449–455, 2015.

[92] Kim TE Olde Dubbelink, Menno M Schoonheim, Jan Berend Deijen, Jos WR Twisk, Frederik Barkhof, and Henk W Berendse. Functional connectivity and cognitive decline over 3 years in parkinson disease. *Neurology*, 83(22):2046–2053, 2014.

[93] Nancy D Chiaravalloti and John DeLuca. Cognitive impairment in multiple sclerosis. *The Lancet Neurology*, 7(12):1139–1151, 2008.

[94] Heather Gilmour, Pamela L Ramage-Morin, and Suzy L Wong. Multiple sclerosis: Prevalence and impact. *Health Rep*, 29(1):3–8, 2018.

[95] Ülkü Türk Börü, Arda Duman, Ahmet Şükrü Kulualp, Neşe Güler, Mustafa Taşdemir, Ümit Yılmaz, Recep Alp, and Cem Bölük. Multiple sclerosis prevalence study: The comparison of 3 coastal cities, located in the black sea and mediterranean regions of turkey. *Medicine*, 97(42), 2018.

[96] Anthony Faivre, Audrey Rico, Wafaa Zaaraoui, Lydie Crespy, Françoise Reuter, Delphine Wybrecht, Elisabeth Soulier, Irina Malikova, Sylviane Confort-Gouny, Patrick J Cozzone, et al. Assessing brain connectivity at rest is clinically relevant in early multiple sclerosis. *Multiple Sclerosis Journal*, 18(9):1251–1258, 2012.

[97] MA Rocca, P Valsasina, Martina Absinta, G Riccitelli, ME Rodegher, P Misci, P Rossi, A Falini, G Comi, and M Filippi. Default-mode network dysfunction and cognitive impairment in progressive ms. *Neurology*, 74(16):1252–1259, 2010.

[98] Simona Bonavita, Antonio Gallo, Rosaria Sacco, Marida Della Corte, Alvino Bisecco, Renato Docimo, Luigi Lavorgna, Daniele Corbo, Alfonso Di Costanzo, Fabio Tortora, et al. Distributed changes in default-mode resting-state connectivity in multiple sclerosis. *Multiple sclerosis journal*, 17(4):411–422, 2011.

[99] Mark J Lowe, Micheal D Phillips, Joseph T Lurito, David Mattson, Mario Dzemidzic, and Vincent P Mathews. Multiple sclerosis: low-frequency temporal blood oxygen level–dependent fluctuations indicate reduced functional connectivity—initial results. *Radiology*, 224(1):184–192, 2002.

[100] Thomas R Insel. Rethinking schizophrenia. *Nature*, 468(7321):187, 2010.

[101] Jennifer Fitzsimmons, Marek Kubicki, and Martha E Shenton. Review of functional and anatomical brain connectivity findings in schizophrenia. *Current opinion in psychiatry*, 26(2):172–187, 2013.

[102] Archana Venkataraman, Thomas J Whitford, Carl-Fredrik Westin, Polina Golland, and Marek Kubicki. Whole brain resting state functional connectivity abnormalities in schizophrenia. *Schizophrenia research*, 139(1-3):7–12, 2012.

[103] Michael Greicius. Resting-state functional connectivity in neuropsychiatric disorders. *Current*

opinion in neurology, 21(4):424–430, 2008.

[104] Mary-Ellen Lynall, Danielle S Bassett, Robert Kerwin, Peter J McKenna, Manfred Kitzbichler, Ulrich Muller, and Ed Bullmore. Functional connectivity and brain networks in schizophrenia. *Journal of Neuroscience*, 30(28):9477–9487, 2010.

[105] Susan Whitfield-Gabrieli, Heidi W Thermenos, Snezana Milanovic, Ming T Tsuang, Stephen V Faraone, Robert W McCarley, Martha E Shenton, Alan I Green, Alfonso Nieto-Castanon, Peter LaViolette, et al. Hyperactivity and hyperconnectivity of the default network in schizophrenia and in first-degree relatives of persons with schizophrenia. *Proceedings of the National Academy of Sciences*, 106(4):1279–1284, 2009.

[106] Pawel Skudlarski, Kanchana Jagannathan, Karen Anderson, Michael C Stevens, Vince D Calhoun, Beata A Skudlarska, and Godfrey Pearlson. Brain connectivity is not only lower but different in schizophrenia: a combined anatomical and functional approach. *Biological psychiatry*, 68(1):61–69, 2010.

[107] Guiomar Oliveira, Assunção Ataíde, Carla Marques, Teresa S Miguel, Ana Margarida Coutinho, Luísa Mota-Vieira, Esmeralda Gonçalves, Nazaré Mendes Lopes, Vitor Rodrigues, Henrique Carmona da Mota, et al. Epidemiology of autism spectrum disorder in portugal: prevalence, clinical characterization, and medical conditions. *Developmental Medicine & Child Neurology*, 49(10):726–733, 2007.

[108] Meng Li, Lei Wei, Jian Liu, Peng Li, Yunfan Wu, et al. Altered functional connectivity in children with low-function autism spectrum disorders. *Frontiers in neuroscience*, 13:806, 2019.

[109] David G Amaral, Cynthia Mills Schumann, and Christine Wu Nordahl. Neuroanatomy of autism. *Trends in neurosciences*, 31(3):137–145, 2008.

[110] Jocelyn V Hull, Lisa B Dokovna, Zachary J Jacokes, Carinna M Torgerson, Andrei Irimia, and John Darrell Van Horn. Resting-state functional connectivity in autism spectrum disorders: a review. *Frontiers in psychiatry*, 7:205, 2017.

[111] Michal Assaf, Kanchana Jagannathan, Vince D Calhoun, Laura Miller, Michael C Stevens, Robert Sahl, Jacqueline G O'Boyle, Robert T Schultz, and Godfrey D Pearlson. Abnormal functional connectivity of default mode sub-networks in autism spectrum disorder patients. *Neuroimage*, 53(1):247–256, 2010.

[112] Vladimir L Cherkassky, Rajesh K Kana, Timothy A Keller, and Marcel Adam Just. Functional connectivity in a baseline resting-state network in autism. *Neuroreport*, 17(16):1687–1690, 2006.

[113] Mark D Shen, Deana D Li, Christopher L Keown, Aaron Lee, Ryan T Johnson, Kathleen Angkustsiri,

Sally J Rogers, Ralph-Axel Müller, David G Amaral, and Christine Wu Nordahl. Functional connectivity of the amygdala is disrupted in preschool-aged children with autism spectrum disorder. *Journal of the American Academy of Child & Adolescent Psychiatry*, 55(9):817–824, 2016.

[114] Heng Chen, Jia Wang, Lucina Q Uddin, Xiaomin Wang, Xiaonan Guo, Fengmei Lu, Xujun Duan, Lijie Wu, and Huafu Chen. Aberrant functional connectivity of neural circuits associated with social and sensorimotor deficits in young children with autism spectrum disorder. *Autism Research*, 11(12):1643–1652, 2018.

[115] Leonardo Cerliani, Maarten Mennes, Rajat M Thomas, Adriana Di Martino, Marc Thioux, and Christian Keysers. Increased functional connectivity between subcortical and cortical resting-state networks in autism spectrum disorder. *JAMA psychiatry*, 72(8):767–777, 2015.

[116] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[117] Steven Mithen. Our 86 billion neurons: She showed it. *The New York Review of Books*.

[118] Ivan Nunes Da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, and Silas Franco dos Reis Alves. Artificial neural networks. *Cham: Springer International Publishing*, 2017.

[119] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[120] Anna Gummeson. Prostate cancer classification using convolutional neural networks. *Master's Theses in Mathematical Sciences*, 2016.

[121] K Andrej, B Janez, and K Andrej. Introduction to the artificial neural networks. *Artificial Neural Networks-Methodological Advances and Biomedical Applications*, 2011.

[122] Loris Nanni, Stefano Ghidoni, and Sheryl Brahnam. Ensemble of convolutional neural networks for bioimage classification. *Applied Computing and Informatics*, 2018.

[123] Nikhil Buduma and Nicholas Locascio. *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. O'Reilly Media, Inc., 2017.

[124] Arden Dertat. Applied deep learning - part 4: Convolutional neural networks. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2, 2017. [Online. Accessed: 2019-10-22].

[125] Chenyou Fan. Survey of convolutional neural network.

[126] Manli Sun, Zhanjie Song, Xiaoheng Jiang, Jing Pan, and Yanwei Pang. Learning pooling for convolutional neural network. *Neurocomputing*, 224:96–104, 2017.

[127] Kalin Kolev. Convexity in image-based 3d surface reconstruction. 2011.

[128] Tom M Mitchell et al. Machine learning. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[129] Rohith Gandhi. A look at gradient descent and rmsprop optimizers. https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b, 2018. [Online. Accessed: 2019-10-22].

[130] Devin Soni. Supervised vs. unsupervised learning. https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d, 2018. [Online. Accessed: 2019-10-23].

[131] Arden Dertat. Applied deep learning - part 3: Autoencoders. https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798/, 2017. [Online. Accessed: 2019-05-10].

[132] Jeremy Jordan. Introduction to autoencoders. jeremyjordan.me/autoencoders/, 2018. [Online. Accessed: 2019-05-12].

[133] How autoencoders work: Intro and implementation. https://www.kaggle.com/gauravkoradiya/how-autoencoders-work-intro-and-implementation, 2018. [Online. Accessed: 2019-05-12].

[134] James L Crowley. Pattern recognition and machine learning. http://www-prima.inrialpes.fr/Prima/Homepages/jlc/Courses/2016/PRML/ENSI3.PRML.S9.pdf, 2017. [Online. Accessed: 2019-05-13].

[135] Ashwini Kumar Pal. Dimension reduction - autoencoders. https://blog.paperspace.com/dimension-reduction-with-autoencoders/, 2018. [Online. Accessed: 2019-05-13].

[136] How to use t-sne effectively. https://distill.pub/2016/misread-tsne/. [Online. Accessed: 2019-05-02].

[137] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[138] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

[139] Principal component analysis. https://plot.ly/python/v3/ipython-notebooks/principal-component-analysis/. [Online. Accessed: 2019-12-14].

[140] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[141] Ashwini Kumar Pal. https://blog.paperspace.com/dimension-reduction-with-t-sne/. [Online. Accessed: 2019-12-14].

[142] Manish Pathak. Introduction to t-sne. https://www.datacamp.com/community/tutorials/introduction-t-sne. [Online. Accessed: 2019-12-14].

[143] Christopher Olah. Visualizing mnist: An exploration of dimensionality reduction. http://colah.

github.io/posts/2014-10-Visualizing-MNIST/, 2014. [Online. Accessed: 2019-05-02].

[144] James E Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, 2005.

[145] Charles Anderson. Docker. *IEEE Software*, 32(3):102–c3, 2015.

[146] Docker. https://www.docker.com/. [Online. Accessed: 2019-12-05].

[147] Maximiliano Osorio, Carlos Buil Aranda, and Hernán Vargas. Dockerpedia: a knowledge graph of docker images. In *International Semantic Web Conference (P&D/Industry/BlueSky)*, 2018.

[148] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[149] Pulkit Sharma. 5 amazing deep learning frameworks every data scientist must know (with illustrated infographic). https://www.analyticsvidhya.com/blog/2019/03/deep-learning-frameworks-comparison/, 2019. [Online. Accessed: 2019-11-13].

[150] Oleksii Kharkovyna. Top 10 best deep learning frameworks in 2019. https://towardsdatascience.com/top-10-best-deep-learning-frameworks-in-2019-5ccb90ea6de, 2019. [Online. Accessed: 2019-11-13].

[151] What is tensorflow? introduction, architecture & example. https://www.guru99.com/what-is-tensorflow.html. [Online. Accessed: 2019-10-20].

[152] Nikhil Ketkar et al. *Deep Learning with Python*. Springer, 2017.

[153] Jeff Hale. Deep learning framework power scores 2018. https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a, 2018. [Online. Accessed: 2019-11-13].

[154] Mitul Makadia. Top 8 deep learning frameworks. https://dzone.com/articles/8-best-deep-learning-frameworks, 2018. [Online. Accessed: 2019-12-05].

[155] Paulo Henrique Vasconcellos. Top 5 python ides for data science. https://www.datacamp.com/community/tutorials/data-science-python-ide, 2018. [Online. Accessed: 2019-12-04].

[156] Spyder. https://www.spyder-ide.org/. [Online. Accessed: 2019-12-04].

[157] Serdar Yegulalp. Review: Six python ides go to the mat. https://www.computerworld.com/article/3132925/review-six-python-ides-go-to-the-mat.html. [Online. Accessed: 2019-12-04].

[158] John Fincher. Python ides and code editors (guide). https://realpython.com/python-ides-code-editors-guide/. [Online. Accessed: 2019-12-04].

[159] JetBrains. Pycharm professional plugins. https://plugins.jetbrains.com/pycharm. [Online. Accessed: 2019-12-04].

[160] What is the jupyter notebook? https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/

what_is_jupyter.html. [Online. Accessed: 2019-12-04].

[161] F Chollet. Deep learning with python, vol. 1. *Greenwich, CT: Manning Publications CO*, 2017.

[162] Yannick Schwartz, Alexis Barbot, Benjamin Thyreau, Vincent Frouin, Gaël Varoquaux, Aditya Siram, Daniel Marcus, and Jean-Baptiste Poline. Pyxnat: Xnat in python. *Frontiers in neuroinformatics*, 6:12, 2012.

[163] Michael Padovano. System and method for accessing a storage area network as network attached storage, 2003. US Patent 6,606,690.

[164] Garth A Gibson and Rodney Van Meter. Network attached storage architecture. *Communications of the ACM*, 43(11):37–45, 2000.

[165] Rick Herrick, Michael McKay, Timothy Olsen, William Horton, Mark Florida, Charles J Moore, and Daniel S Marcus. Data dictionary services in xnat and the human connectome project. *Frontiers in neuroinformatics*, 8:65, 2014.

[166] Daniel S Marcus, Timothy R Olsen, Mohana Ramaratnam, and Randy L Buckner. The extensible neuroimaging archive toolkit. *Neuroinformatics*, 5(1):11–33, 2007.

[167] Rogério Moreira. Xnatum. https://github.com/rgllm/xnatum. [Online. Accessed: 2019-12-10].

[168] David C Van Essen and Matthew F Glasser. The human connectome project: progress and prospects. In *Cerebrum: the Dana forum on brain science*, volume 2016. Dana Foundation, 2016.

[169] Matthew F Glasser, Stamatios N Sotiropoulos, J Anthony Wilson, Timothy S Coalson, Bruce Fischl, Jesper L Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R Polimeni, et al. The minimal preprocessing pipelines for the human connectome project. *Neuroimage*, 80:105–124, 2013.

[170] Numpy Documentation. Numpy. https://numpy.org/. [Online. Accessed: 2019-12-04].

[171] NiBabel Documentation. Neuroimaging in python. https://nipy.org/nibabel/. [Online. Accessed: 2019-12-04].

[172] Matplotlib Documentation. Python plotting. https://matplotlib.org/. [Online. Accessed: 2019-12-04].

[173] scikit-learn: machine learning in python. https://scikit-learn.org/. [Online. Accessed: 2019-12-04].

[174] seaborn: statistical data visualization. https://seaborn.pydata.org/. [Online. Accessed: 2019-12-04].

[175] Python Documentation. glob - unix style pathname pattern expansion. https://docs.python.org/2/library/glob.html. [Online. Accessed: 2019-12-04].

[176] Python Documentation. os - miscellaneous operating system interfaces. https://docs.python.org/3/library/os.html. [Online. Accessed: 2019-12-04].

[177] Maria Giulia Preti and Dimitri Van De Ville. Eigenmaps of dynamic functional connectivity: Voxel-level dominant patterns through eigenvector centrality. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 988–991. IEEE, 2016.

[178] Elena A Allen, Eswar Damaraju, Sergey M Plis, Erik B Erhardt, Tom Eichele, and Vince D Calhoun. Tracking whole-brain connectivity dynamics in the resting state. *Cerebral cortex*, 24(3):663–676, 2014.

[179] Maria Giulia Preti, Thomas AW Bolton, and Dimitri Van De Ville. The dynamic functional connectome: State-of-the-art and perspectives. *Neuroimage*, 160:41–54, 2017.

[180] Gemeng Zhang, Biao Cai, Aiying Zhang, Julia M Stephen, Tony W Wilson, Vince D Calhoun, and Yu-Ping Wang Wang. Estimating dynamic functional brain cconnectivity with a sparse hidden markov model. *IEEE transactions on medical imaging*, 2019.

[181] Diego Vidaurre, Stephen M Smith, and Mark W Woolrich. Brain network dynamics are hierarchically organized in time. *Proceedings of the National Academy of Sciences*, 114(48):12827–12832, 2017.

[182] Rong Duan, Hong Man, Wei Jiang, and Wen-Ching Liu. Activation detection on fmri time series using hidden markov model. In *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.*, pages 510–513. IEEE, 2005.

[183] Heung-Il Suk, Chong-Yaw Wee, Seong-Whan Lee, and Dinggang Shen. State-space model with deep learning for functional dynamics estimation in resting-state fmri. *NeuroImage*, 129:292–307, 2016.