University of Pisa
Department of Physics E. Fermi
Master's degree in Physics

Identification of alterations in functional brain connectivity with explainable Artificial Intelligence analysis of multicenter MRI data

Supervisors:
Prof. Alessandra Retico
Prof. Piernicola Oliva

Candidate:
Federico Campo

Academic year: 2021/2022

Contents

Abstract

Autism Spectrum Disorders (ASD) are a diagnostic category of neurodevelopmental disorders manifesting from the early age and involving behavioral and cognitive impairments. So far, no univocal biomarkers have been identified for this disorder. The most accurate diagnostic tool remains the direct behavioural evaluation of the child and the interview of partents/caregivers. Neuroimaging plays nowadays an important role in the identification of neural correlates of this condition.

In this work we focus on the analysis of brain functional connectivity data to study possible differences between patients with ASD and healthy controls. To tackle this challenge, we employed machine learning and deep learning models. Our database is constituted of Magnetic Resonance scans provided by the ABIDE consortium, consisting of structural and functional scans for a total of approximately 2200 subjects, collected from different medical centers. Roughly one half of them belong to healthy individuals (controls) and the other half to people with diagnosed ASD.

Starting from these data, the workflow of this Thesis can be summarised in the following steps:

- MRI and fMRI data preprocessing and registration to a common template, and extraction from the fMRI scans of the temporal series encoding the brain activity of different brain areas for each patient.
- Creation of a connectivity map for each patient. This connectivity matrix is created by computing a measure of Functional Connectivity (FC) for every pair of timeseries of a single patient. The values of FC were computed using two different approaches: the Pearson correlation between the two timeseries, and a measure of coherence among the two signals, based on a time-frequency analysis by means of wavelet transform.
- Implementation of a harmonization procedure to correct for potential biases and distinctive traits in data, linked to the scan acquisition procedures employed in a medical center which can differ between different centers both in acquisition parameters and in hardware components.
- Study and classification of FC matrices with deep learning methods. The two different FC matrices described above are compared to assess which one is able to yield the best separability between controls and subjects with ASD. Classification performances achieved on raw data and harmonized ones by means of different procedures are compared. In addition, a harmonization implemented inside a deep learning model through an adversarial deep learning technique was implemented and tested.
- Deep learning model is compared to a Random Forest classifier, a standard machine learning model, to determine if a deep model brings an actual benefit to this kind of analysis.
- After the definition of the appropriate harmonization procedure, we implemented an interpretation method called SHAP to explain which features contributed the most to the final outcome of machine learning classification, and from them we determined the most relevant brain areas that allow a separation between controls and subjects with ASD.

Organization of contents

BACKGROUND: In this section the relevant theoretical arguments that lay the groundwork to the whole Thesis are briefly presented. In chapter 1 we outline the main problems associated with the Autism Spectrum Disorders, from the biological development to the social issues it generates. We briefly review some studies carried out so far to study this challenging issue. We also provide a brief overview of the main techniques employed in this work for data acquisition, processing and analysis.

Chapter 2 provides an overview of the basic physical principles of MRI imaging and the main acquisition sequences. It also includes a description of the physical and biological principles underlying the functional MRI modality which is the one we will examine in this work.

Chapter 3 provides a brief explanation of machine learning and deep learning models diving into the mathematical formulation of the main concepts. We aim to give an idea of the structure of a machine learning model, the learning procedure, and the important metrics for a proper evaluation of the model performances. Ultimately, we describe a common way to reduce data dimensionality keeping as much information as possible and reducing the number of uninformative or redundant features from data.

Chapter 4 describes the theory behind a specific strategy to inspect a deep learning model and understand the processes that brought it to a certain output given some input data.

MATERIALS & METHODS: This section describes the instruments we used to run the analysis including the dataset, the preprocessing steps, and the data preparation pipeline employed to obtain a measure of the brain functional connectivity.

Chapter 5 describes and analyze the ABIDE dataset, which is a publicly available collection of more than 2200 scans of healthy subjects and patients with ASD, collected from different medical centers.

Chapter 6 provides an overview of the most important preprocessing steps to normalize and align all these scans to a common template, to run meaningful statistical analyses and to extract brain timeseries form different brain areas aligned to a common coordinate system. We describe C-PAC, which is the software employed in this Thesis to carry out the image preprocessing and to extract the timeseries.

Chapter 7 describes the two methods we used to calculate the measures of functional connectivity coefficients and create the connectivity matrix for each subject. We computed the connectivity by using the Pearson correlation; we illustrate the formalism of the wavelet transform for time-frequency analysis and the process to extract an alternative connectivity measure from the time-frequency data of two timeseries.

Chapter 8 describes the regression model upon which the multicenter data harmonization procedure is based. This is an analytical method to remove inter-site variability from our data and to avoid biases towards the acquisition source.

Chapter 9 describes the structure and the general idea behind a domain adversarial neural

network, which is a deep neural network that aims to make control vs ASD classification following a procedure that makes data independent from the acquisition site, while learning how to discriminate controls from subjects with ASD.

Chapter 10 describes the main aspects of the SHAP algorithm for machine learning model explanation. We discuss the general idea behind its implementation and the quantities to determine the contribution of a feature to the output of a machine learning model.

IMPLEMENTATION & RESULTS: In this last section, the implementation of the analysis steps illustrated above is described and the results are evaluated and discussed. We evaluated the effect of the harmonization applied to our data and the capability in the ASD vs control discrimination of Pearson-based functional connectivity values vs the wavelet-based ones. We also determined the most relevant features from our data, which played an important role in the problem, and the brain areas that are involved in an altered connectivity.

Chapter 11 provides an overview of the workflow followed for classification problems: the choice of the connectivity measures, the selection of subjects and the structure of different harmonization and the classification schemes.

Chapter 12 presents the results of the analytical harmonization applied to our data, to provide a visual understanding of how this process affects a feature distribution.

Chapter 13 presents the structure deep neural networks employed in classification analysis. Two DNNs are described: a conventional deep neural network and a domain-adversarial neural network.

Chapter 14 compares the classification results obtained with a deep neural network on harmonized and non-harmonized (raw) data with the results of the domain-adversarial neural network, and with the results obtained using a simple Random Forest classifier. The performances achieved also after applying a dimensionality reduction step are presented in order to assess whether this procedure is effective in reducing the dimensionality of this dataset while preserving the informative content of the features.

At the end, chapter 15 shows the implementation of a feature importance assessment procedure. We show which brain connectivity feature contributed the most to the outcome of a prediction, and then to the discrimination between healthy controls and patients with ASD. Subsequently, we identify the brain areas which are mainly involved in this condition.

List of abbreviations

AI: Artificial Intelligence

ANN: Artificial Neural Network

ASD: Autism Spectrum Disorder

AUC: Area Under Curve

BOLD: Blood-Oxigen-Level Dependent

DANN: Domain Adversarial Neural Network

DNN: Deep Neural Network

EPI: Echo Planar Imaging

FC: Functional Connectivity

FIQ: Full Intelligence Quotient

ML: Machine Learning

PC: Principal Component

PCA: Principal Component Analysis

rs-fMRI: resting state functional Magnetic Resonance Imaging

BACKGROUND

Chapter 1

Introduction

1.1 Autism Spectrum Disorder

Autism Spectrum Disorders (ASD) are a diagnostic category of neurodevelopmental disorders that is drawing more and more attention, for its social impact on families and society and for the raise, in the last decades, in the number of diagnosed cases. One in 44 children has been identified with ASD according to a recent study [1]. ASD is classified as a neurodevelopmental behavioural disorder [2] [3] and refers to a broad range of conditions manifesting as deficits in social communication and interaction such as reduced sociability or empathy, repetitive behaviours, resistance to changes, and sometimes even speech difficulties.[4] To the present days, ASD is diagnosed through a comportamental assessment test since there is not a definite biological test. Early signs start appearing by the age of 2 - 3, ages at which children usually start interacting with parents and with other children. Parents are asked to answer a set of questions about every-day behaviour of their children like the checklist for Autism in Toddler [5]. However, are not uncommon cases where ASD is discovered only after the adolescence. Currently ADI-R (Autism Diagnostic Interview - Revised) and ADOS (Autism Diagnostic Observation Schedule) are considered the 'gold standard' tools for diagnosis of ASD [6] [7].

Nevertheless, besides these behavioural test, to achieve a more accurate diagnosis, biological information should be taken into account as well. For a better treatment of this condition, it would be of great importance to make an early diagnosis, in order to provide in time the help and support people need. Hopefully, treating children when this disorder is still it in its early stage, could bring to an improvement in their quality of life.

The main biological factors that bring to the developing of autism are not very clear so far, and is acquiring more consensus between neuroscientists and medical doctors that there is not just a single cause, but a concatenation and coexistence of various triggering factors. Among them, the most strongly suspected are identified to be genetic and environmental, the latter including air pollution.

Genetic origin of this disorder is associated with a rare gene mutation, such as a deletion, duplication or inversion, and even though most of the mutation that increase the risk of developing autism are not been traced. It has been assessed though, that it has high inheritance traits. From studies on twins it has been assesses that between omozygotes twins there is around 90 % or probability that both of them develop ASD, while for heterozygote this probability falls to around 7% for men and 1-2% for women [8]

From a neurological point of view, ASD may affect a brain both in its structure and in its functionality.

Structural studies usually focus on volumentric and morphometric analyzes to examine differences in brain anatomy. It has been studied how ASD could alter the symmetry between the two emispheres [9] of the brain. In children it has been observed an increase in total brain volume as well as an enlargement of the left superior temporal gyrus. However but this trend is not well defined for older ages [10].

Functional neuroimaging researches mainly focus on impaired functionality. Different studies pointed out a reduced information processing due to synaptic dysfunction that manifests in a reduced or altered brain functional connectivity. Functional connectivity measures aim to describe statistical dependencies between brain areas in time, by studying temporal correlation between different brain parts even between those which are spatially separated. Altered brain activity is found by different but controversal studies on functional connectivity (FC) both hyper-FC and hypo-FC were detected in ASD patients. The coexistence of a hyper- and hypo-FC traits has also been found between different areas of the brain. In particular it seems that FC abnormalities have an age-dependent trend: over connectivity is usually observed in young children while under connectivity in adolescences and adults [11] [12].

1.2 fMRI as a functional connectivity investigation tool

In the last decades, different approaches to study ASD have been proposed, to find a relation between different brain areas involved in lower or higher functional connectivity. Different diagnostic tools are being used to investigate this matter, from different perspectives and at different scales, such as Magnetic Resonance Imaging (MRI), functional-MRI (fMRI), or electroencephalography (EEG).

Magnetic resonance imaging is employed to obtain images of brain both for structural studies and for functional through the technique of functional MRI (fMRI). An other tool employed to study temporal signals of the brain is the EEG which differs from fMRI because of the type of signal it is sensitive to and because of the different resolution in both signal and spatial domains. With EEG it is possible to study signals in a time scale comparable with the neuronal activation time (timescale $\sim \mu s$), while in fMRI is reported a signal due to the hemodynamic response following a neuronal activation (timescale $\sim s$).

fMRI is a diagnostic tool that investigates physiological changes linked to blood flow variations across the whole brain structure. Measures of blood properties are strictly linked with measures of the neuronal activity: an increase of neuronal activity leads to a boost in blood flow and an increase of vessel size within a specific brain area, because of the bigger demand of oxigen and other nutrients to the neurons in that area.

The different neural connections are identified by measuring the blood-oxigen-level dependent (BOLD) fluctuations, from different areas of the brain, using the signal difference between oxy (diamagnetic) and deoxy-hemoglobin (paramagnetic). The spontaneous fluctuations in the BOLD signal during resting state are considered a strong indicator for the assessment of the properties of the brain system.

fMRI can be acquired during the resting state of a patient or during a task. We refer to these procedures as rs-fMRI and task-based fMRI. While task-based fMRI is a good instrument to investigate the local activation of a single brain area during a task, rs-fMRI was proven to be suitable for examining functional connectivity among all the brain regions, and highlight the difference between a normal brain and one affected by a disorder. Resting state fMRI is drawing attention for the investigation of a number of disease evolutions including ASD.

Resting state fMRI is an fMRI acquisition carried out while the patient is in a relaxed state, and is not performing any active task. The patient is simply asked to stay still with eyes closed

or open while fixating a reference. Following this setup, the brain is at rest and its activity is not perturbed by active tasks, such as moving an arm, or passive actions, like being exposed to different visual stimuli, which are common activities for task-based functional connectivity. Resting state setup revealed to be a powerful tool to investigate the intrinsic generated brain activity and study the altered functional connectivity networks in subjects with mental disorders.

1.3 Machine learning to deal with non-linear problems

During the study of characteristic traits between controls and ASD patients, different attributes are involved and affect data in a strong and evident way, the most important of which are age, sex and the full intelligence quotient (FIQ). Age, for example, affects both structural and functional data, with an observed overgrowth of the brain volume and hyper-connectivity, in toddlers. Both traits tend to decrease with increasing age. Moreover, on control subjects, even though it is not fully characterized, brain structure and functional connectivity seem to decrease with age [13].

Sex is the other factor that gives characteristic traits to brain features. Some brain areas exhibit an increased functional connectivity in female subjects, and, from a combined analysis of both age and sex, it appears that in men some brain structures show more pronounced aging effect than women [14].

In addition, if we limit our focus to functional data, the eye status at scan plays an important role. Functional connectivity with open steady eyes results to be different from that when data are acquired with closed eyes, with strong differences and higher connectivity in different brain areas between the two cohorts of subjects [15]. Furthermore, an other aspect to be take into account regarding patients with closed eyes, is that during the scan acquisition, they may fall asleep, and this would heavily modify the functional brain activity, resulting in a modified functional connectivity pattern.

It is clear that the distinction of ASD patients from healthy subjects is not a straightforward task and a simple univariate analysis would not be sufficient, because of the presence of different confounding factors contributing to the characterization of data.

To take on this challenge, one of the most promising tools that allow us to deal with complex, non-linear problems is the use of machine learning algorithms to study data extracted from fMRI such as pairwise correlations between different regions of the brain Machine learning belongs to artificial intelligence (AI) tools and make use of algorithms that learn from data, and modify their parameters with the aim of recognizing distinctive properties from data and make predictions or classification, on new unseen data, trying to gradually reduce the error and make more accurate predictions.

A relevant aspect that makes machine learning and artificial neural networks so popular is the ability to deal with non-linear problems, by introducing several non-linear functions during training. This allows to obtain non-linear outputs from each input data, which would hopefully help in the distinction between characteristic traits of healthy people and of ASD patients.

A drawback of machine learning is that, to perform well, an algorithm needs to be trained on datasets of large dimension, because the bigger the dataset, the more the algorithm is able to generalize information and the better are its performances on new data. This is true for all kinds of machine learning algorithms and especially for Deep Neural Networks (DNNs) which are the principal family of algorithms we are going to employ in this work.

1.4 The need for a data harmonization procedure

To achieve the goal of obtaining a large dataset, particularly in medical field where data are not easily available from a single center or are not in a sufficient amount to perform a large-scale analysis, data from different acquisition centers need to be put together. In the last years, this procedure is becoming increasingly popular and several multicenter medical datasets such as the Human Connectome Project, the Alzheimer's Disease Neuroimaging Initiative (ADNI)[1] or Autism Brain Imaging Data Exchange (ABIDE)[2] were created to obtain large collections of data to make significant statistical analysis. Unfortunately, this procedure brings with it a downside regarding the unavoidable bias towards the site that data belong to. This bias is a result of hardware and scan procedure differences between centers and it is not feasible to ask to require that all the medical centers across the world use exactly the same instrumentation and uniform to a single common acquisition protocol.

To work out this problem and try to mitigate the inter-scan variability, we need a harmonization procedure to remove site-dependent information, leaving all the rest of important information unchanged.

In this work, two different approaches are proposed: an analytical harmonization and a deep learning approach.

Analytical harmonization is a procedure that modifies features extracted from images acquired at multiple sites with shifts and rescaling to remove only inter-site related effects, while trying to preserve all other information, as biological-related effects.

The second approach which is based on deep learning, tries to carry out the classification of control/ASD data extracting from input data both the information related to the class (control/ASD) and to the site. Then it uses the latter to remove the bias linked to sites before discriminating subjects with ASD from controls.

[1] https://adni.loni.usc.edu/

[2] https://fcon_1000.projects.nitrc.org/indi/abide/

Chapter 2

Principles of Magnetic Resonance Imaging

In this chapter we report some basic principles that underlie medical imaging diagnostic modality based on the Nuclear Magnetic Resonance phenomenom. We only provide a glimpse on what are the main principles behind this technique and the main acquisition sequences employed in diagnostics, whereas we remind to textbook for a more complete description [16].

2.1 Physical principles

Magnetic Resonance Imaging (MRI), is a non-invasive imaging technique widely employed in radiology to obtain anatomical images or to study physiological processes, taking advantage of its flexible sensitivity to different tissues.

When a nucleus is subjected to an external magnetic field, the interaction would make the nuclei align with the magnetic field, but, since protons and nuclei have an intrinsic spin, and hence an angular moment $\overrightarrow{\mu}$, they start precessing.

Precession occurs because of the torque created by the interaction of a static magnetic field with the angular moment of a spinning nucleus. The relation is given by

$$\overrightarrow{\tau} = \overrightarrow{\mu} \times \overrightarrow{B_0} \tag{2.1}$$

Precession occurs around the symmetry axis given by the direction of the magnetic field $\overrightarrow{B_0}$ as schematically shown in figure 2.1.

When a nucleus starts this rotatory movement, the characteristic frequency ω_0 is uniquely determined by the properties of the nucleus itself and by the strength of the magnetic field; the relation is given by the Larmor equation and the frequency ω_0 is called **Larmor frequency**

$$\omega_0 = \gamma B_0 \tag{2.2}$$

where γ is a constant called *gyromagnetic factor* and is it characteristic of each nucleus. The gyromagnetic factor of a

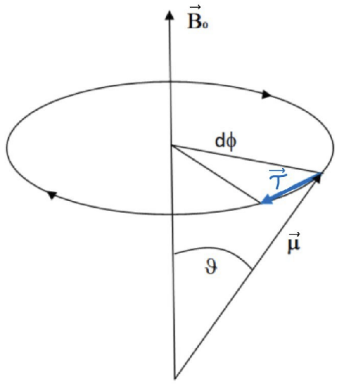


Figure 2.1: *Representation of a angular moment vector $\vec{\mu}$ under a static magnetic field $\vec{B}_0$ and the resulting moment $\vec{\tau}$ which causes the precession*

proton in water has a value of $\gamma = 2.7 \cdot 10^8 \frac{rad \cdot s}{T}$. For a typical scanner with a magnetic field of 2-3 T, the Larmor frequency assumes values of tens of MHz i.e. in the radio frequency (RF) range.

When protons in a body are exposed to a magnetic field, they can assume two states in relation to the direction of the external field: parallel and anti-parallel (as an example in figure 2.1 a parallel configuration is represented).

Proton with different orientation with respect to the magnetic field have different energies. The potential energy of a particle with magnetic moment $\vec{\mu}$ immersed in a magnetic field $\vec{B_0}$ is given by

$$U = -\vec{\mu} \cdot \vec{B_0} \tag{2.3}$$

If we suppose $\vec{B_0}$ along the z axis, it will have only component along z, so the scalar product will just concern the z component of the magnetic moment μ_z. Recalling the relation between the angular moment and the magnetic moment $\vec{\mu} = \gamma \vec{J}$, and the quantization of the angular moment $J_z = m_s \hbar$. m_s represents the spin of the proton and can assume values $-\frac{1}{2}$ or $+\frac{1}{2}$. The potential energy of a proton when subjected to a magnetic field can be written as:

$$U = -\vec{\mu} \cdot \vec{B_0} = -\mu_z B_z = -\gamma m_s \hbar B_z = -m_s \hbar \omega_0 = \pm \frac{1}{2} \hbar \omega_0 \ . \tag{2.4}$$

In the formula above, energy with negative sign (associated to proton with spin $+1/2$) corresponds to proton aligned in parallel with the magnetic field, while energy with positive sign belongs to proton in an anti-parallel state. For a thermodynamic system, with a given temperature T, the probability to find particles with energy ϵ is given by the Boltzmann distribution

$$P(\epsilon) = \frac{1}{Z} e^{-\frac{\epsilon}{kT}} \tag{2.5}$$

where k is the Boltzmann's constant, and Z is the partition function of the system $Z = \sum_\epsilon e^{-\epsilon/kT}$. Using this relation we can compute the difference between the number of proton in the two states parallel and anti-parallel

$$\Delta N = N^+ - N^- = \frac{N}{Z}(e^{-\frac{\hbar\omega_0}{kT}} - e^{\frac{\hbar\omega_0}{kT}}) \approx \frac{N}{2} \frac{\hbar\omega_0}{kT} \tag{2.6}$$

where the approximation of the exponential since for human body is $\hbar\omega_0 \ll kT$. What we obtain with this formula is that the number of parallel and anti-parallel protons is not the same, and the reason behind this comes from energetic considerations since protons with spin parallel to the magnetic field have a lower potential energy. This imply that when a body is exposed to a magnetic field, this small excess of number of protons in an parallel state cause the body to have an intrinsec magnetization $\vec{M}$.

This net magnetization can be represented by a single vector pointing parallel to the direction of the external field and if we want to detect a signal related to this magnetization, we have to perturb this vector from its equilibrium state. The perturbation would cause the precession of this magnetization vector around the direction of the magnetic field, with its own Larmor frequency. This precession will produce a changing magnetic flux which can be detected by an external coil. To kick this vector away from its equilibrium state, a second external magnetic field is employed under the form of a radiofrequency pulse, with a frequency resonating with the Larmor frequency of the precessing spins.

This radiofrequency pulse is usually applied to rotate spins in a direction ortogonal to the static magnetic field $\vec{B_0}$ which we can suppose to be along the z axis $\vec{B_0} = B_0 \hat{z}$

We create this way a magnetization component onto the x-y plane, aligned with the direction of the radiofrequency pulse. This magnetization vector on transverse (x-y) plane is called *transverse*

magnetization, and can be denoted as $\vec{M}_\perp = M_x\hat{x} + M_y\hat{y}$. After this radio frequency pulse, protons will start rotating in the x-y plane, but, since this pulse is not persistent, after it is switched off they gradually lose their initial energy and tend to realign to the z-axis.

This process of realigning to the z-axis is called longitudinal relaxation (longitudinal with respect to the original $B_0\hat{z}$ direction).

The evolution of the longitudinal and transverse magnetization are modeled with the introduction of two time constant, T_1 and T_2, respectively. Longitudinal relaxation occurs because the system goes from higher energy state (when it is flipped on the x-y plane) to a state of thermodynamic equilibrium with its surrondings, regaining its previous magnitude.

This process follows an exponential decay in time shown in equation 2.7, with a time constant T_1, which for the physical reason underlying it, is also called spin-lattice relaxation time:

$$M_z(t) = M_0(1 - e^{-t/T_1}) \ . \tag{2.7}$$

The transverse magnetization vector follows a different evolution. When studying the evolution of $\vec{M}_\perp$, an other effects has to be taken into account: the spin-spin interaction between protons. If we consider a physical system where protons are immersed in a magnetic field, each proton interacts with this external field plus all the small fields created by the surrounding protons and their associated spins. This leads to the presence of a local field for each proton, slightly different from the external magnetic field $\vec{B}_0$. These different local fields affect the evolution of the transverse magnetization because they cause a relative dephasing of protons among each other. According to this, if immediately after the RF pulse all the spins can be imagined as aligned to the x axis, they start to fan out. This effect speeds up the loss of the total transverse magnetization on the x-y plane.

If we study this evolution from the rotating reference frame, (rotating with the same Larmor frequency of protons) the module of the transverse magnetization follows an exponential decay in the form of

$$M_\perp(t) = M_\perp(0)e^{-t/T_2} \ . \tag{2.8}$$

.

Since T_2 constant comprises the spin-spin interaction process plus the spin-lattice interaction which cause the realignment of spins along the $\hat{z}$ axis, it is always smaller than T_1. For protons in human tissue, T_1 ranges from 10 to 100 milliseconds, while T_2 is usually of the order of 10 milliseconds.

In a real phyisical system, however, there is and additional dephasing source, coming from external field inhomogeneities. This effect is often taken into account by the introduction of a different decay time T_2' which along with T_2 bring to a overall time costant given by

$$\frac{1}{T_2*} = \frac{1}{T_2} + \frac{1}{T_2'} \tag{2.9}$$

2.2 Image acquisition and k-space

The physical process underlying signal acquisition is Faraday induction, according to which an electrical potential, called electromotive force (emf), is generated by the variation of a magnetic flux over time, $emf = -\frac{d\Phi}{dt}$, being Φ the varying flux through the receiving coil.

A simple setup in MRI to generate a varying flux over time is the application of a single RF pulse, which is able to make the magnetization rotate by an angle of $\pi/2$ (flip angle) with respect to the direction of the magnetic field $\vec{B}_0$. The variation of magnetic flux occurs while tipped spins

are rotating in the x-y plane and the magnetization vector is relaxing towards the longitudinal axis. This induced emf signal can be detected by properly oriented and tuned RF coils. This simple experiment is referred as Free Induction Decay (FID) and is usually performed in any MRI scan to tune RF coils and optimise system response.

Usually the varying flux we are interested in is the one along the transversal plane. The reasons behind this are mainly two: 1. the weak signal along the longitudinal axes would be saturated by the strong magnetic field of the static magnetic field $\vec{B_0}$. 2. The signal coming from the transverse magnetization $M_\perp$ is representative of all the main information we need for the analysis of a material: T_1, T_2 and the proton density ρ.

To properly create an image based on these information we need to spatially encode each signal received by the changing magnetic flux. In order to relate a signal with a spatial position, in addition to the initial static magnetic field $\vec{B_0}$ we have to place a second field which causes a controlled local modification of the magnetic field $\vec{B_0}$. This additional field $\vec{B'}$ needs to be non-uniform and to have a lower magnitude of $\vec{B_0}$ for each point.

Its distribution follows a spatial gradient so that the total magnetic field along $\hat{z}$-axis is given by the sum of this two contributes, and the signal contains space-varying frequency components according to equation 2.2 which can be rewritten as

$$\omega(z) = \gamma B(z) \tag{2.10}$$

being z the spatial coordinate and where $B(z)$ is the total magnetic field now given by the relation

$$B(z,t) = B_0 + z \cdot G(t) \qquad G(t) = \partial B'/\partial z \tag{2.11}$$

This spatial changes in magnetic field causes different parts of the body along the z axis to have different Larmor frequencies. This gradient along the longitudinal direction of the static field $\vec{B_0}$ is usually referred as Slice Selection gradient G_{ss}. Choosing the z axis as the direction of the slice selection, we acquire images on the transversal x-y plane.

When acquiring an image, data are collected under the form of a matrix called k-space. K-space is a coordinate system used to store spatial frequencies information. From these information we can retrieve the usual MRI image (containing spatial, anatomical information) by applying the inverse 2D Fourier transform.

In a k-space matrix, low spatial frequencies, corresponding to large object across the whole real image, are encoded at the center of the matrix and high spatial frequencies corresponding to small objects and finer details, are encoded in the peripheries.

The construction of k-space is done step by step in relation to the combination of applied RF pulse and fields time by time, producing a trajectory on the k-space.

To acquire spatial information in the x-y plane, we need to introduce two new gradients (one for each direction): a readout (or frequency encoding) gradient and a phase encoding gradient.

The frequency encoding gradient acts on one direction in the transversal plane; let us identify this direction as the x axis for a clearer visualization. Just like the slice selection gradient, it consists of a linear changing magnetic field to modify the Larmor frequency of the spins along the x axis. According to the acquisition technique we want to perform, it can be applied forward and after a while, reversed. This allows a refocusing of all the dephased spins due to spin-spin interaction; we will see more in detail this concept when we will discuss acquisition sequences. For now, we just need to know that the refocusing of all the dephased spins occurs after a time interval called echo time (TE).

The phase encoding gradient acts the exact same way of the previous one, but it acts on the y axis. It is switched on and acts by modifying the Larmor frequency but it is just a temporary change. When it is switched off, all the spins along this direction continue precessing all at the same frequency, but with a relative phase between them.

The combined action of these two gradients allow us to acquire different lines of the k-space.

2.3 Acquisition sequences

Depending on the type of analysis we want to carry out, we have to focus on some magnetic properties rather than others. For example, for a an anatomical (structural) image, we may be interested in a good contrast between different tissues, while in a functional scan we are interested in detecting temporal signal changes.

Two main parameters drive the differences between different acquisition sequences. For the moment we define them, but a contextualized use of them can help to clarify their meaning. This can be found in the following sections describing different acquisition sequences.

- **Repetition time (TR)** is the time interval between one RF pulse and the next.
- **Echo time (TE)** is the time interval between a RF pulse and the echo peak

Changes in TR, TE and RF pulses characteristics (such as angle or number of pulses), allow to perform different acquisition sequences and to emphasise one of the three fundamental contrasts between different tissues based on the parameters T_1, T_2 or the proton density ρ.

As a general rule:

- T_1-weighted images are acquired with small values of TR. This avoid that all nuclei are back to their longitudinal position, allowing a better tissue contrast. TE has to be very short to avoid contribution due to T_2-related effects.
- T_2-weighted images are acquired with high values of TR ($TR \gg T_1$) and values of TE comparable with T_2 time constant.
- To enhance the spin density contrast, TR has to be chosen as long as possible while TE needs to be short.

2.3.1 Spin-echo sequence

Spin echo (SE) sequence is one of the most used one. It lets us retrieve lost information due to spin dephasing caused by magnetic field inhomogeneities.

The spin echo method employs two RF pulses: the first flips protons by an angle of $\pi/2$ with respect to the $\vec{B_0}$ direction and the second of an angle π in relation to the direction of the first pulse. All the sequence can be summarized by the following steps:

1. The first radio-frequency pulse is applied with an angle of $\pi/2$ in relation to the direction of the external magnetic field $\overrightarrow{B_0}$. This process as mentioned in section 2.1 causes the spin flip onto the transversal plane x-y. Just as an example, we imagine they are flipped into the x direction. They gradually start dephasing both because of spin-spin interactions and small external field inhomogeneities, which are different from point to point. Because of this, they start fanning out, because some rotate faster and some are delayed with respect to the average magnetization vector.

2. At the instant t = τ the second RF pulse is sent. It is created in order to flip protons by angle of π with respect to their direction after the first impulse, so if the first impulse flips them along the $\hat{x}$ direction, this overturns them along $-\hat{x}$. This has the effect of turning all spins with a phase ϕ to a phase $\pi - \phi$.

3. Dephasing spins continue accumulating a phase, however, since they were flipped by 180 degrees, the same process according to which they were previously fanning out, push them to converge. In fact, in an interval Δt two spins have accumulated a phase $\Delta\phi(t+\Delta t) = -\gamma\delta B\Delta t$, due to an inhomogeneity in magnetic field δB_0, after they are turned over, this relative phase becomes $\Delta\phi(t + \Delta t) = +\gamma\delta B\Delta t$. δB_0, though, continue causing a dephasing, that after an further interval Δt results in a total dephasing $\Delta\phi(t + 2\Delta\phi) = +\gamma\delta B\Delta t - \gamma\delta B\Delta t = 0$. This rephasing results in a recreated transverse magnetization vector (echo) in the opposite direction of the first one created after the $\pi/2$ pulse.

With this procedure, it is possible to recover the loss of transversal magnetization due to inhomogeneities of the magnetic field.

The time interval covering the application of the first impulse $\pi/2$ to the instant at which phases turn back to zero is called **echo time** TE. Between the application of the π pulse and the echo time, the acquisition sampling of k-space starts.

This method is an effective strategy to remove the nuisance effect due to field inhomogeneities associated to T_2', but do not reduce the effect of T_2 due to local fields and spin-spin interactions. The reason behind this is that static field inhomogeneities remain the same even after the π pulse, so they act the same way even after the pulse, while local fields which cause spin-spin interactions change and the rate at which they accumulate phase changes with them. In general no refocus strategy is possible to correct for this effect. Fortunately, this is not a critical issue for liquids since the time interval over which data are collected is usually smaller than the T_2 time constant.

2.3.2 Gradient Echo sequences

The sequence employed to acquire functional imaging is the Echo Planar Imaging (EPI) which is a particular sequence from the family of gradient echo (GE) sequences. Gradient echo sequences differ from spin echo because they allow a faster acquisition. They use a single excitation pulse with a flip angle less than 90 deg, this allows a minor time interval to make spins back to their longituginal component as they are not completely flipped on the transverse plane.

The transverse component just created decay and dephase according to T_2*. At this point in spin echo sequence, the π pulse would be applied; in its place, in GE sequences, the gradient along the x axis is reversed. This gradient inversion causes the spins to rephase but what is relevant is that this gradient inversion does not act neither on dephasing due to field inhomogeneities nor on dephasing due to chemical shifts, but just on dephasing due to the gradient field. This is the most important aspect that makes GE different from SE: the relaxation is dictated by T_2* and not just by T_2. In gradient echo sequence, the echo peak lies on T_2* decay curve while in a spin echo, it lies on T_2 decay curve. Thanks to this property, this image acquisition is susceptible to any chemical variation such as that due to hemoglobin shift.

2.3.3 Echo Planar Imaging

Echo Planar Imaging (EPI) refers to a technique to acquire the entire 2D k-space after a single RF pulse. This is achieved by applying intermittent small-phase encoding gradients called blips: triangular gradients that are switched on and off between gradient echoes in a multi-echo acquisition.

To illustrate this method can be useful to give a look at figure 2.2 where all the gradients in the game are illustrated, and to discuss line by line what they represent:

1. A single RF pulse with an angle of 90 degrees is applied to the sample;
2. Together with the RF pulse, a slice of the body is selected, corresponding to a position along the z axis, by applying the slice selection gradient;
3. Now we focus on the single 2D slice to begin the k-space data acquisition starting from the bottom line of the diagram in figure 2.3: both gradient of phase encoding and frequency encoding are switched on;
4. The frequency encoding (readout) gradient is reversed to create the first echo;
5. When the first echo occurs, the first line of k-space (along the k_x or $k_{readout}$ direction) is acquired. In figure 2.3 this lines corresponds to the horizontal bottom line;
6. Now we have to move to higher values of k_y or k_{phase}: to accomplish this, the phase encoding gradient performs a small blip: it is switched on for a brief time and then switched off. This dephase spins a little further so that we move up along the k_y direction in figure 2.3 and the system is ready to acquire the other line of k-space;
7. the frequency encoding gradient is reversed again and the second horizontal line along k_x is acquired.
8. This process is iterated: these last two steps are repeated until all the k-space is covered and the slice data is fully acquired.

This acquisition technique allows the acquisition of a single slice in a short time (around 50-100 ms). The whole k-space for a single slice is acquired sequentially, following a snake-like pattern and the number of k-space line acquired after each single RF pulse is named differently according to the scan brand factory: examples are *EPI factor* or *Echo train length*.

Since slice data are collected after GE inversions, this imaging technique is sensitive to the T_2^* time constant, which makes it suitable for functional imaging. For this reason this kind of imaging is often called T_2^*-weighted imaging. In fact, the period of the readout gradient is the echo time TE, and is modulated such that is sensible to T_2^* setting $TE \approx T_2^*$.

2.4 Functional MRI

Functional MRI (fMRI) is a non-invasive diagnostic tool which aims to measure the neural activity of different parts of the brain. The vast majority of scan acquisitions is carried out using EPI sequences (section 2.3.3) and consists of several 3D volumes acquired every 1.5-2 seconds (corresponding to the repetition time TR) for the entire duration of a scan which lasts from 5 to more than 8 minutes, depending on the scanner model.

The detected signal comes from the blood oxygenation level fluctuations following a neuronal activation, and for this reason this kind of analysis is referred as blood oxygenation level-dependent (BOLD) imaging.

Blood can be portrayed as a colloidal mixture where blood cells constitute around 40-45 % of volume [16].

From a physical point of view, the variation in oxygen content in blood affects the local magnetic susceptibility of the blood. Signal coming from blood cells is mainly due to the presence (in

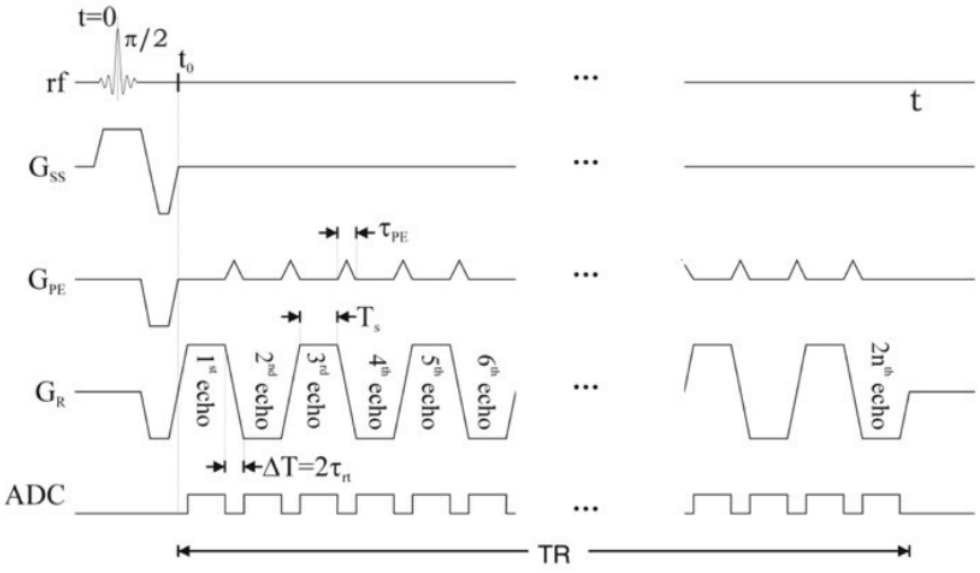


Figure 2.2: *Gradient activation sequences for a EPI acquisition where a complete image is acquired after a single RF impulse. G_{SS} is the slide selecting gradient, G_{PE} is the phase encoding gradient, with tiny blips to dephase spins along the y direction, G_R the readout gradient, or frequency encoding gradient, reversed each time for the acquisition of a line along the x axis. Finally, the ADC is switched on to record signals during the echo peak.*

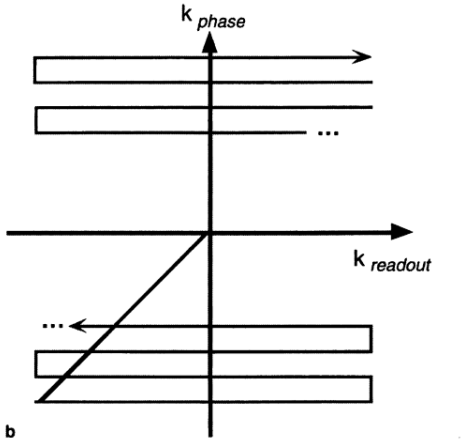


Figure 2.3: *Acquisition trajectory on the k-space for a single slice. Acquisition starts to negative values of k_x ($k_{readout}$) and k_y (k_{phase}), then, changing the direction of the readout gradient, the acquisition proceeds back and forth, gradually moving towards higher value of k_y at each blip of the phase encoding gradient.*

each blood cell) of several molecules of hemoglobin: a protein containing 4 heme groups each one including an iron atom which binds an oxygen molecule and carries it throughout veins and capillaries. Hemoglobin can be in two states: oxy-hemoglobin and deoxy-hemoglobin. These two molecules differ in the presence of the bounded oxygen molecule, which reflects in differences in their magnetic susceptibility as oxy-hemoglobin has diamagnetic properties while deoxy-hemoglobin is paramagnetic. This difference is due to unpaired iron electrons in the deoxy-hemoglobin which lead to an unshielded molecule against the external magnetic field. The presence of deoxy-hemoglobin causes local field inhomogeneities, leading to a reduction of the main signal coming from water molecules.

This signal weakening is due to spins going out of phase between each other more quickly, because of these additional field inhomogeneities, which result in reducing the total magnetization. This means that nuclei would lose their magnetization faster than the typical T_2^* decay constant. For this reason if the scanner is tuned to the T_2^* relaxation time, it would be possible to appreciate this chemical shift between oxygenated and deoxygenated areas. This gives the alternative name of BOLD-images as T_2^*-weighted images [17].

Since the presence of deoxy-hemoglobin causes a weakening of the signal, when a brain area is activated, it demands a greater amount of oxygen carried by oxy-hemoglobin molecules which results in a signal increase.

The process that, from a neuronal stimulus leads to a measurable blood signal is governed by the hemodynamic response function (HRF) shown in figure 2.4. This figure represent the blood response immediately after a neuronal stimulus (which occurs on a timescale of $\approx$ 10-100 μs). The BOLD signal takes $\approx$ 4-6s to reach a peak and a total of 20-30 s to return to its zero baseline.

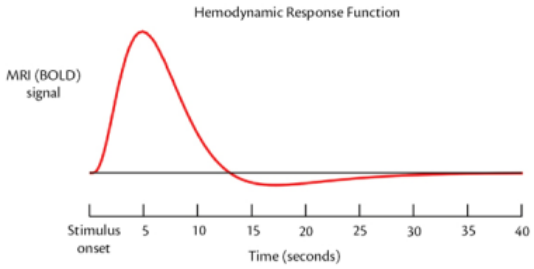


Figure 2.4: *Evolution of BOLD signal of an adult brain during time. At the instant t=0 the neuronal stimulus occurs; after that, it takes around six seconds to reach a maximum of signal, and a total of more than twenty-five seconds to return to its baseline value*

Since we are interested in these slow signal trends, the TR of 1.5-2 seconds is the minimum necessary to have an analysis sensitive to these signals. According to the Shannon-Nyquist sampling theorem, the maximum detectable frequency in a signal is equal to 1/2 of the sampling frequency. For this reason with a TR of 1.5 seconds for example, we are able to detect signals with a period $\geq$ 3 seconds.

During an MRI and fMRI scan session, data are usually acquired slice by slice, and the thinner is the slice, the more spatial resolution it is possible to accomplish. But there is a trade-off between spatial and temporal resolution: decreasing the slice thickness leads to a better space resolution but at the cost of increasing repetition time to maintain the same Signal to Noise Ratio (SNR). Thie main reason is that signal is proportional to the number of hydrogen nuclei, which is proportional to the slice volume. In order to obtain a strong BOLD signal, echo time plays a significant role as well: to obtain the maximum strenght signal echo time has to be set $TE \approx T_2^*$ that means TE around 30 ms. Images acquired using a shorter echo time have a weaker BOLD signal because of the weaker signal to detect [18].

Chapter 3

Machine learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that aims to learn from examples and apply this knowledge on new, unseen data. It is possible to make predictions or perform classification and, through the training process, gradually improve accuracy on a specific task.

In the following sections we discuss some of the main concepts related to machine learning. However, for a more exhaustive discussion we refer the reader to [19] [20] [21] [22].

According to the analysis we aim to perform, and the type of dataset we work on, machine learning algorithms can be divided into two macro areas: supervised and unsupervised algorithms. Given a dataset, consisting of a collection of samples, each one containing a certain number of features, machine learning algorithms can be categorized as follows.

- An **unsupervised** algorithm is able to learn the principal properties from the dataset structure and extract information from unlabeled samples of the dataset. Analysis with unsupervised algorithms include clustering or dimensionality reduction.

  Clustering is the most simple and intuitive example of an unsupervised learning: it is basically a classification process through which unlabeled data are reorganized and classified into subgroups according to some common properties or distance measures that arise from the analysis. After this process, data of each group, called cluster, share a certain degree of similarity in feature probability distribution.

  Dimensionality reduction is another important example of unsupervised learning process: it is performed with the task of finding a different representation of the data, with a lower dimensionality, and preserve as much information as possible from them. This can be accomplished either by compressing data into a lower-dimensional space, or by searching the main sources of variance across the data and create a representation in such a way that the dimensions of the new representation are statistically independent.

- A **supervised** algorithm, is able to learn from a dataset where each sample is associated to a label, specifying the class it belongs to. Thus, the algorithm is trained to learn features characteristics for each class and to predict the correct label associated to each input data.

  Essentially, given an input data x and an associated label y, a model tries to estimate the probability of obtaining y given x: $p(y|x)$. These algorithms are referred as supervised because a label is provided for each input data.

The general task of a supervised ML algorithm is to learn information from a train dataset and generalize it in order to perform well on an unseen dataset called test.

Performing well means to produce a low error on the test dataset. In a supervised classification problem, for example, a low error can be achieved when the model is able to predict the correct label of the samples that belongs to the test dataset.

The ability of a ML to acquire knowledge is often referred as *capacity*.

Capacity can be identified as the level of complexity that a model is able to learn and is related to the number of parameters available for the ML algorithm to create an inner representation of training data. The greater the capacity, the better a model is able to learn patterns from them. However, the task of a ML algorithm is not just to learn from the training dataset, but also to perform well on the test set. The themes of capacity and performances are strictly linked to two critical issues in machine learning: the concept of underfitting and overfitting.

- Underfitting occurs when the model is not able to learn the required amount of information during training. This usually happens for simple models when they have a small number of parameters in regard to the number needed to represent the input data. Small capacity results in poor performances, because the model is not able to learn the underlying structure of a complex dataset.

- Overfitting occurs when the model has too many parameters compared to those required to represent the input data. What happens then is that the model learns from all the data seen during train, but it is not able to generalize information. As a result, the model performs well on the train dataset, but it achieves low performances on the unseen test dataset.

To understand this concept it can be helpful to visualize an example in a two-dimensional space as reported in figure 3.1. The problem is a simple fit of a set of datapoints. Each datapoint is sampled from a quadratic curve and the objective is to find the right function to describe the distribution of datapoints. We use the training set, represented as blue dots, to fit the model and the test set (orange dots) to evaluate its performances. We can approach this problem using three different functions, a linear function (plot on the left), a quadratic function (in the centre) and a polynomial function with grade equal or greater to the number of data points (plot on the right). As shown in figure the linear model is not able to perform a proper fit of the training data since it has an insufficient number of parameters to describe that distribution.

On the other hand, the polynomial function has too many parameters, and is able to fit the train distribution, but it does not suit to describe data points belonging to the test. If the number of parameters (the grade of the polynomial in this example) is greater than the number of data points, it is possible to obtain an infinite number of different curves that are suitable to fit the train data, and finding the one that fits well the test dataset as well, is a hard task.

If, however, the number of parameters of the fitting model is the right one to describe the data distribution, it is possible to obtain a model able to fit both train and test datapoints.

Typical examples of traditional ML methods are Random Forest, Support Vector Machines (SVM) and K-nearest neighbor (KNN). A subset of Machine Learning is Deep Learning. The main difference between them lies on the structure of their algorithms. Deep learning methods include Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN) and more.

3.1 Random Forest

Random Forest classifier is a common machine learning algorithm that belongs to the category of *ensemble* classifiers. It combines multiple decision tree models to reduce overfitting of data and to create a model able to achieve good generalizability. In order to describe a Random Forest algorithm, we need to start discussing how a decision tree classifier works.

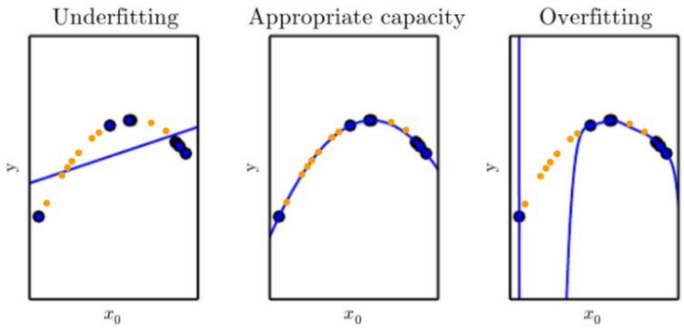


Figure 3.1: *Underfitting, appropriate fit and overfitting condition for a 2D dataset: three models with different capacities are trained on blue data, and tested on orange datapoints.*

Decision tree classifiers

A decision tree aims to learn distinctive traits from input data by asking yes/no questions. Focusing on one single input feature, the classifier splits the dataset based on the value of that feature according to an if/else statement like "if $feature\_i > a$ ". Thus, each branch of a decision tree consists of a question that causes the split of the dataset into two smaller sub-datasets. In a dataset containing multiple features, the process is recursively repeated until it reaches an end point, called *leaf*, corresponding to the ultimate partition, containing data points belonging to a single class.

The goal of a tree is to construct branches so that the partition are informative about the class labels. In a practical way, given a dataset X made of n different data samples, where each sample contains m features and belongs to one of k different classes, the algorithm constructs a tree following these simple steps: starting from the top node called *root*, it searches among the m features the one which allows the best split between classes and split the dataset into two subsets, each one constituting a node. Then, for each node, the process is repeated until it reaches a leaf.

The best split is performed according to a pre-defined objective function that we want to maximize throughout the construction of a tree. Usually these functions regard measures of the homogeneity of class labels in a node, usually referred as impurity measures. To lower the impurity of a node, the optimization algorithm seeks for the maximization of the information gain, defined as the difference between the parent node impurity and the weighted sum of the child nodes impurities.

A commonly used impurity measure is Gini Impurity [21]. It can be interpreted as the probebility to misclassify an observation. If a node contains a sub-dataset Q with a total number of data n, belonging to k different classes, the Gini Impurity measure H(Q) is obtained by the simple relation

$$H(Q) = \sum_k p_k(1 - p_k) = 1 - \sum_k p_k^2 \tag{3.1}$$

where p_k is the fraction of data in Q belonging to class k. Following this principle, the splitting process is iterated and the tree is grown.

The structure of a tree can be visualized in a plot that contains all the information necessary to understand it, such as the feature according to which each split is made, and the impurity measure of each node. This property is one of those that make decision trees so popular: they are easy to interpret since their structure can be visualized in a clear plot. Other positive aspects of decision trees are their easy-to-use implementation, which requires little data preparation and their rapidity since their complexity goes like $O(n_{features} n_{samples}^2 log(n_{samples}))$ [23].

An example of a decision tree dealing with flower classification is shown in figure 3.2. We chose this example because is easier to understand decision tree using this data where each feature is labeled with an intuitive name and contains a physical quantity easily comparable with everyday life such as petal lenght expressed in centimeters. This example is taken from the scikit-learn website [1].

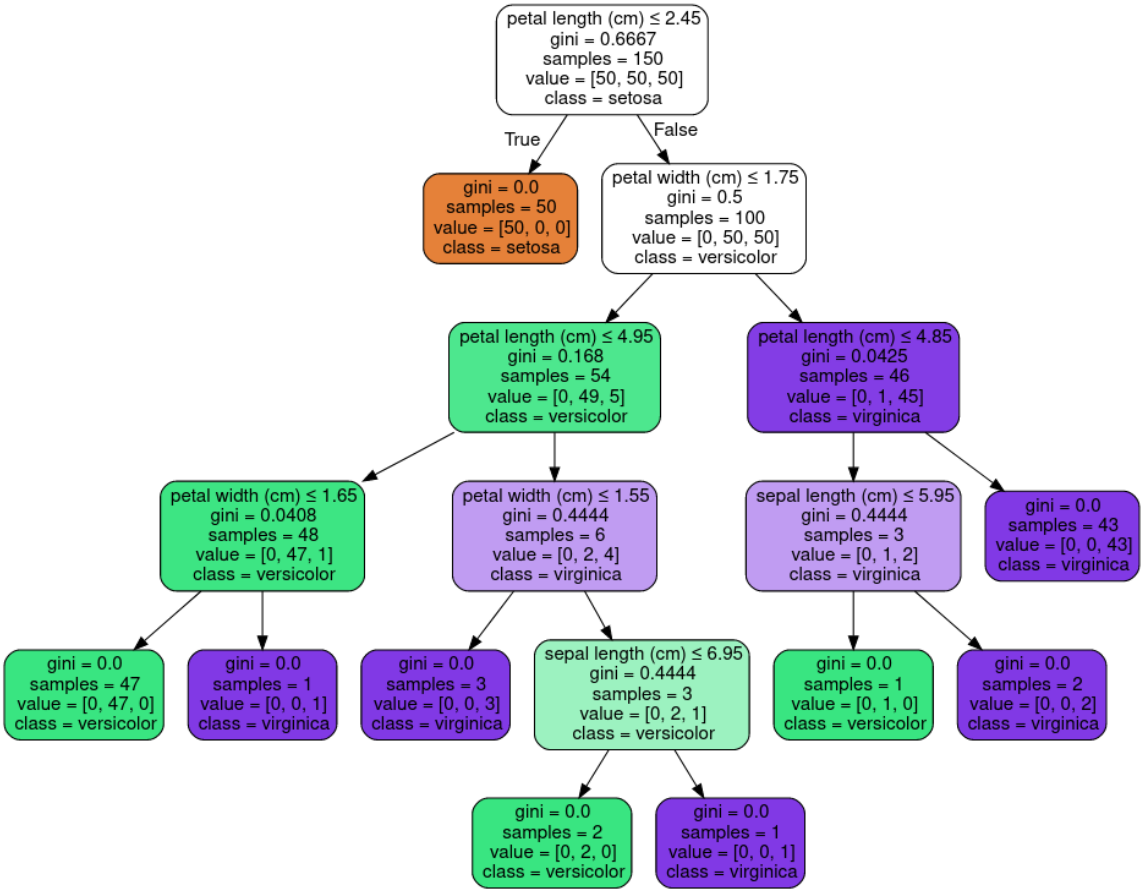


Figure 3.2: *A graphical representation of a decision tree for classification of the Iris dataset consisting on data belonging to three different classes of iris: Setosa, Versicolour, and Virginica. Each node of the tree shows the criterion used for its split, the Gini values, the total number of samples in the node and the main class that data belong to.*

Instead of looking at the whole tree, we may be interested in what feature contributed the most to the final output. To this end, a decision tree usually weights each feature with a number from 0 to 1, according to how much a feature contributed to the impurity decrease along the tree. The importance of a feature is calculated as the decrease in node impurity weighted by the probability of reaching that node. This probability can be computed as the number of samples that reach that node divided by the total number of samples [21] [22].

Unfortunately though, one of the main drowbacks of decision trees is that the iterating process towards the reaching of a leaf, can bring to the creation of a over-complex model that overfits the training data. That is, the model is not able to generalize the learnt information to a test dataset. To prevent overfitting, one possible strategy is to early stop the iteration towards the leaves, and

[1] https://scikit-learn.org/stable/modules/tree.html#

avoid the creation of too small partitions. An other strategy, which leads to the construction of a Random Forest classifier, is the creation of different trees and put information together, in order to obtain a stronger model, more prone to generalize information and to reduce overfitting.

Random Forest

A Random Forest is a collection of decision trees where each tree is different from the others. Each tree tends to overfit, but following different patterns. Taking advantage of this process, it is possible to reduce overfitting by averaging different results. Random Forest gets its name because of the insertion of randomness during the construction of different trees. Given an input dataset X with n different samples, there are two main steps where randomicity plays an important role: during the selection of the number of samples to grow each tree on, and during the selection of the number of features to consider, when looking for the best split of a node.

In a Random Forest, one of the main parameters is the number of tree to grow. Once selected this number, the process can start where each tree is constructed according to the following steps:

1. Randomly select $n' \leq n$ samples from the input dataset X. This operation is called bootstrap.
2. Grow a decision tree from this bootstrap sample. For each node, the algorithm randomly limits the number of features available and computes the best split on this remaining subset of features. This avoids correlations between trees and results in better performances.
3. Repeat steps 1) and 2) as many times as the number of trees to build.

Once all the trees are created, the forest is ready to make predictions. Each tree is used to make a prediction and all the results are collected. The final prediction of the forest is assessed by majoring vote: the predicted class will then be the one predicted by the majority of classifiers.

Just as we discussed for tree classifiers, it is possible to extract information about features for a Random Forest as well. Important features for a Random Forest are assessed by extracting them from each tree and by averaging these results.

3.2 Deep Learning and Artificial Neural Networks

Artificial neural networks (ANN) or Deep Neural Networks (DNN) are inspired by brain neuronal structure, and even if in a simplified way, they try to emulate the learning process of a brain.

These algorithms are called deep because of their structure: they are organized in layers, each one containing several fundamental units called neurons. Neurons between layers are connected to each other in a way that emulates synapsis transmission between neurons of a biological brain. Thanks to this similitude, they are also called artificial neural networks.

The foundamental unit which constitutes a neuronal network is the artificial neuron. The most relevant example of artificial neuron is the *perceptron* shown in figure 3.3.

The perceptron is the foundamental unit of a supervised deep learning algorithm: it takes as input a data, for example an n-dimensional array $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ and returns an output: a real number computed by applying a function to a linear combination of all the inputs $y = f(z) = f\left(\sum_i w_i x_i + b\right)$, where w_i are inner weight associated to each input and b is a bias term. w_i and b are randomly set when a perceptron is initialized. The function f that determines the output is called *activation function*, and can be either a simple step function, or a more complex function. We will briefly discuss some of the most important activation functions in section 3.2.1.

A deep neural network is a hierarchical organization of neurons into layers, connected to each other. Input data are passed to the first input layer. Each neuron acts like a perceptron and

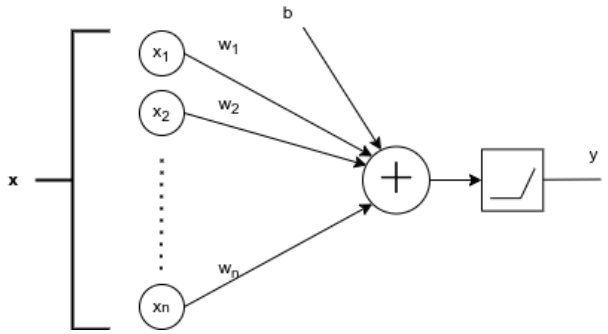


Figure 3.3: *A schematic representation of a perceptron: this perceptron receives an input vector* $\mathbf{x}$*, with n features* $x_1, ...x_n$*, each one weighted with a different weight* $w_1, ...w_n$ *and a offset b. It computes a linear combination of them and returns an output, activated by a step function: it returns 0 if the linear combination value does not reach a certain threshold, or returns the value itself if it does.*

produces an output using the activation function. All the neurons in a layer have the same activation function, but it can differ from the activation function of other layers. The outputs from the first layer neurons, are passed to the second layer, becoming the input to each neuron belonging to this layer. This process is reiterated through all the layers, up to the final output layer. As schematized in figure 3.4, an artificial neural network is mainly composed of three parts: an input layer, some middle layers, also called *hidden layers*, and a final output layer. Each neuron of a layer is linked to all the neurons of the previous and next layer and each connection is weighted. An input vector is passed to the first layer (input layer), then, each neuron of the first hidden layer, computes a linear combination of its input using the weighted connection and returns an output by applying the activation function. This process is repeated for each layer up to the output layer and the output vector is collected.

In figure 3.4 is shown an example where the output is a $m-$dimensional vector $(y_1, .., y_m)$.

The process through which given an input vector $\mathbf{x} \in \Re^n$ we obtain an output, or prediction $\mathbf{y} \in \Re^m$ is called *forward propagation*.

When a network is created, the weights of each connection are randomly set, but during the training procedure they are updated through a process called *back-propagation* in order to improve the network performances. To this end, the prediction is compared with the actual value of the label associated to the input data point, and an error is computed using a *loss function* as will be discussed in section 3.2.2. Then, during the back-propagation, the algorithm modifies its weights in order to minimize the difference between the actual value and the predicted one and reduce the value of the loss; this process will be discussed in section 3.2.3.

To fully define a neural network it is essential to define a set of parameters called hyperparameters, the principals of which are:

- Number of layers
- Number of neuron for each layer
- Activation function for each layer
- Number of epochs (or iterations)

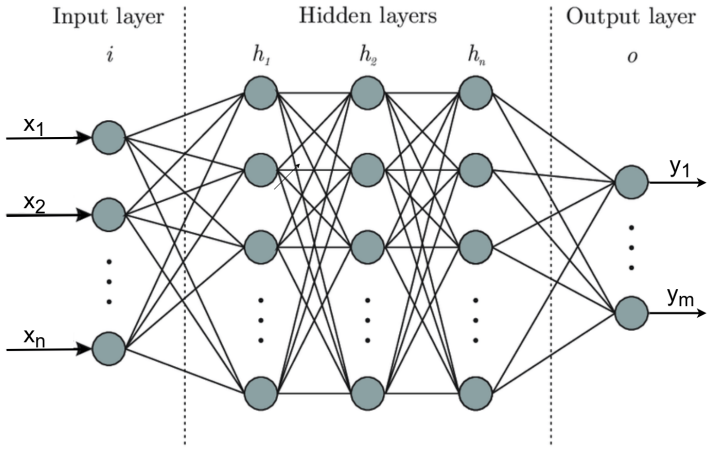


Figure 3.4: *Schematic representation of an artificial neural network containing an input layer, three hidden layers and an output layer: a single vector contained n features* $(x_1, x_2, ...x_n)$ *is given as input to the first input layer i. Each neuron of the first hidden layer computes a value by applying an activation function to a linear combination of its input. All the neurons in a hidden layer are linked to the neuron of the previous and the next layer. And at the end, a vector* $(y_1, ..., y_m)$ *is returned.*

- Learning rate

Number of layers and numbers of neurons are the most important parameters to determine the capacity of a deep learning model. As mentioned in previous sections, the capacity of a model determines its ability to learn complex patterns from data and to generalize them to a test dataset.

A common way to set the value of hyperparameters is while studying the evolution of the learning process. To monitor the progresses of a model during train it is a common practice to split the entire dataset into three subsets: a train, a validation and a test set. The model is trained on the train dataset, and the validation set is used to make constant checkups on how the model is learning. The validation dataset is used for the fine tuning of hyperparameters and it typically consists of 10-20% of the whole dataset. When the best combination of hyperparameters is found, it is possible to actually train the model using both the train and the validation sets, and evaluate its performances on the test dataset.

Regularization strategies

Some regularization procedures are often implemented to avoid overfitting. The strategy is to build a model with a capacity slightly higher than the necessary in order to perform well on the training dataset, and then, implement some regularization techniques to achieve good generalization performances.

Some of the most popular are Dropout and Batch-Normalization.

- Dropout is a regularization strategy that inserts randomness during the training of a model. It is usually applied to the neurons of hidden layers and it randomly drops a certain fraction of hidden neurons and their connections, during each training cycle. The percentage of neurons to drop, corresponding to the probability for a single neuron to be dropped, is set by the user. A visual representation of what occurs is illustrated in figure 3.5. When discarding some neurons in a layer, the remaining neurons needs to rescale their weights to account for the missing connection; doing so, every neuron cannot rely on the input of all the preceding

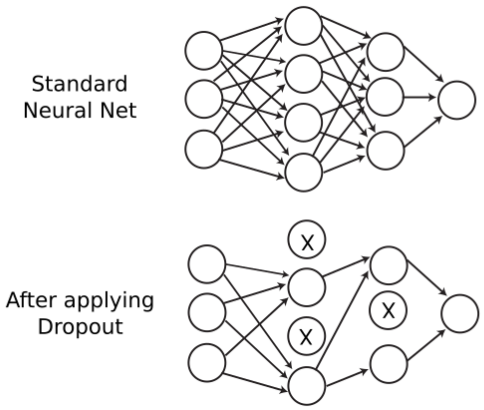


Figure 3.5: *Schematic representation of a 0.5 dropout procedure: in a standard neural network, each neuron is linked to all the others, but if we implement a 0.5 dropout, for each iteration, neurons on hidden layers have a 50% probability to be dropped and excluded from the computation of outputs*

neurons and the network is forced to learn more robust patterns from the data. With this strategy we achieve the same results as we would obtain by training different models with different structures and then average all the outputs.

- Batch normalization is a regularization scheme that has been commonly adopted since its introduction in 2015 [24]. It is based on the observation that a neural network works and performs better when its inputs are normalized, because this prevents the saturation of its neurons. A neuron can in fact saturate and settle to a certain value because of a high input. This causes the neuron outputs value to be always close to the asymptotic end of its activation function (see section 3.2.1), resulting in a biased and less accurate prediction. What is essentially done is then a simple scaling of each neuron input: for a layer l with d neurons its input $\mathbf{x} = \{x_1^l, x_2^l, ..., x_d^l\}$ is normalized by removing the mean value across all the input data, and by dividing for their variance $x_i^l \rightarrow \tilde{x}_i^l = \frac{x_i^l - \mathbb{E}[x_i^l]}{\sqrt{Var(x_i^l)}}$.

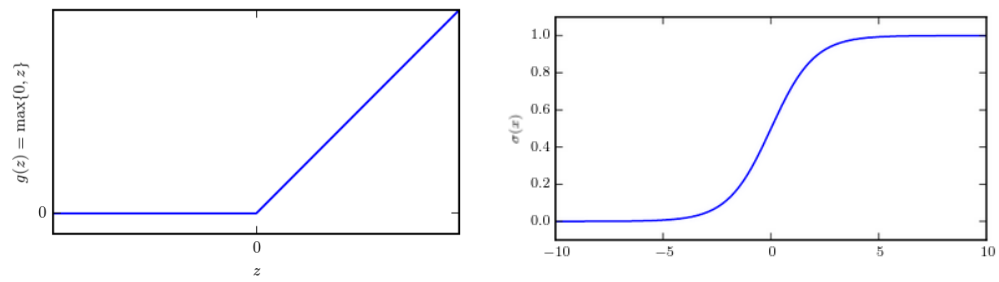
3.2.1 Activation functions

The activation function of a neuron, and consequently of a layer, defines the output of each neuron belonging to that layer. There are different activation functions, linear or non-linear.

In practice, however, it would not be very useful to introduce a linear activation function since the combination of linear functions is a linear function itself. In fact, in a neural model, we seek to introduce non-linearity in order to deal with more complex tasks. There are several non-linear functions that can be employed depending on the classification task we are performing. Denoting as z the scalar product between an input vector $\mathbf{x}$ and the weights vector $\mathbf{w}$ plus an eventual offset b, we show some of the most popular activation functions:

- The **ReLU** function, shown in figure 3.6a, is defined as

$$\phi(z) = max(0, z) = \begin{cases} 0 \ for \ z < 0 \\ z \ for \ z > 0 \end{cases} \tag{3.2}$$



(a) ***(b)***

Figure 3.6: *(a) Rectified linear unit (ReLU) function: it returns zero if its argument is negative and returns the argument itself if it is positive, according to equation 3.2; (b) Sigmoid function: returns a value between 0 and 1, according to equation 3.3*

it returns the maximum value between the input and zero, it essentially puts to zero all negative inputs and leaves the positives unchanged.

  - A modified version of the ReLU function called Leaky ReLU was introduced as $\phi_{leaky}(z) = \begin{cases} \alpha z \ for \ z < 0 \\ z \ for \ z > 0 \end{cases}$ where α is a coefficient usually smaller than 1, typically of the order of 10^{-2}

- The **sigmoid** function, or logistic function show in figure 3.6b is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

  and outputs a real number between 0 and 1. For this reason it is a common choice to model a probability. For example, in a binary classification task, when it is employed as activation function of the last layer, it can be interpreted as the probability of the input data to belong to one of the two classes.

- The softmax function, is a generalization of the logistic function, and is commonly used for multiclass classification. It is defined as

$$\phi(z) = p(y = i|z) = \frac{e^{z_i}}{\sum_{j=1}^{M} e^{z_j}} \tag{3.4}$$

  It is applied to the vector $\mathbf{z}$ and describes the probability of the input vector $\mathbf{x}$ of belonging to the class i over a total of M classes. Using this function, classes are regarded as mutually exclusives, so if the probability to belong to class i is p, the probability to belong to one of the other classes is $1 - p$. To hold this property, softmax can't be applied independently to each input z_i, since it depends on all elements of $\mathbf{z} = \{z_1, ..., z_M\}$

3.2.2 Loss functions

In order to train a network and improve its performances, we have to compare the predicted output with the actual value of the label associated to a data, and compute an error, or distance between these two values.

This error is computed by using a *loss function*. In common classification tasks, the goal of a network is to gradually reduce the error, looking for a minimum of these functions, according to a process called back-propagation, that we discuss in section 3.2.3.

The most common loss function for classification problems is the cross-entropy loss. It relies on the concept of cross-entropy between two distributions $\hat{y}$ *and* y defined as

$$H(y, \hat{y}) = -\sum_{m=0}^{M-1} y_i \cdot log(\hat{y}_i) \tag{3.5}$$

Where the sum is intended over all the M possible values a variable y can assume (all the M possible classes in a classification problem).

If classification only concerns two classes, it is called binary classification, and its associated label usually have values $y = \{0, 1\}$. If we explicit the sum of equation 3.5 for M = 2, cross entropy for a binary classification for one observation can be calculated as

$$\ell_i = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \tag{3.6}$$

If we have N data belonging to two classes, we can write the *binary cross-entropy* loss as

$$L = \frac{1}{N}\sum_{i}^{N} \ell_i = -\frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{3.7}$$

Binary crossentropy loss can be generalized to the case of multi-class classification. In this case, if data belong to M classes, labels y can assume values $y \in \{0, 1, ..., M - 1\}$. To define a loss, each label y_i must be one-hot encoded (see chapter 13) in order to create a binary vector of dimension M, with all but one entry equal to zero, and the position of the only entry equals to 1 specifies the class.

In this case the loss function is called *categorical cross-entropy*

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{m=0}^{M-1} y_{im} log(\hat{y}_{im}) + (1 - y_{im}) log(1 - \hat{y}_{im})) \tag{3.8}$$

In equations 3.7 and 3.8 $\hat{y}$ are probability of the input data i^{th} to belong to a class. For binary classification, this probability is usually modeled with a sigmoid function, while for multi-class classification, a softmax function is usually employed.

3.2.3 Gradient descent and back-propagation

During the training of a model, after the calculation of the loss function, the network seeks for the estimation of the best parameters through an algorithm called *back-propagation*. To this end, the weights of the network are updated in order to minimize the loss function. To perform the minimization of a loss function, even if it would be theoretically possible to find a minimum by means of an analytical calculus of its gradient with respect to all the weights of a network, in practice usually the number of weights is so huge that numerical methods must be employed. The most popular method to compute gradients and optimize parameters is the **gradient descent**.

We denote a generic loss function as $L(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_i, \hat{y}_i(\mathbf{w}))$ where $\mathbf{w}$ is a vector of weights, the minimum of this function corresponding to the vector $\mathbf{w}^0$ can be found by following the subsequent steps:

1. Choose a random initial guess for $\mathbf{w}^0$ and start iterating

2. At iteration $i+1$ we reach a weights vector $\mathbf{w}^{i+1}$ given by the formula

$$\mathbf{w}^{i+1} = \mathbf{w}^i - \eta \nabla_{\mathbf{w}} L(\mathbf{w}^i) \tag{3.9}$$

where the $\nabla_{\mathbf{w}}$ indicate the gradient of the cost function with respect to $\mathbf{w}$ components, and η is a parameters called *learning rate*. It specifies the dimension of each step during the descent toward the minimum of the cost function.

A drawback of this gradient descent algorithm is that it works by computing all the gradients for all the data points before updating the weights. So the weight is updated only after the whole dataset has been seen, resulting in a huge computational cost.

An optimized version of this algorithm is called **Stochastic Gradient Descent** (SGD). It is often used because it brings some advantages such as decreasing computational cost and, introducing stochasticity. The introduction of randomicity during the gradient computation results in a reduced chance for the algorithm to get stuck in local minima.

SGD introduces stochasticity by approximating the gradient calculated using all the input data, with a gradient computed using only a small subset of input data, called *mini-batch*.

With a dataset comprising N input data, we can create subsets containing m elements and obtain N/m minibatches. The gradient is computed on a mini-batch and weights are updated. This process is repeated and when the gradient is computed over all the mini-batches it is said that the training proces completed an *epoch*. The number of epochs is one of the hyperparameters to set when choosing a training strategy of a model.

Even though SGD performs quite well, it can be further improved by introducing the concept of momentum. Momentum is represented by a parameter $0 \leq \gamma \leq 1$ that takes track of the descending direction by running an average over all the preceding encountered gradients. This process helps the algorithm speeding up the descending process if in a certain direction the gradient is constant and it does not change slope.

The descending process can be further improved by taking track of the curvature of the loss function and adapt the learning rate according to it.

To accomplish this task algorithm has to be improved by adding the calculation of second order momenta also called *uncentered variance*. This procedure would ideally bring to the calculation of the Hessian matrix but with an increasing of computational costs. A recently introduced algorithm called **Adam**, can accomplish this task by approximating the calculus of the second-order momenta [25]. It accomplishes this computation by making use of two different optimization algorithms: AdaGrad and RMSProp. With this improvement, the algorithm keeps track of the curvature of the loss function in the space of parameters, and takes big leaps in steepest direction and small steps in flatter ones, allowing us to adaptively change the descending step size. This leads to better performances and quicker finding of the minima than simpler SGD with momentum algorithm.
The process of computing gradients is an important step of the back-propagation algorithm, which is the process through which the weights of a network are updated. A neural network with L layers, given an input data, produces an output through the feedforward propagation, and with this value, the loss is calculated. Denoting with z_j^l the weighted input to the $j-th$ neuron of the $l-th$ layer, the first step of back-propagation is to compute the error

$$\Delta_j^L = \partial L / \partial z_j^L$$

On the last layer. Then, using this quantity it is possible to calculate Δ_j^l for all the layers by

exploiting the chain rule of derivatives

$$\Delta_j^l = \sum_k \frac{\partial L}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

Once computed all the errors, it is possible, for each layer, to compute the loss with respect to the weights of the networks and modify them according to equation 3.9.

3.3 Dimensionality reduction: PCA

Principal Components Analysis is an unsupervised learning algorithm which aims to reduce the dimensionality of input data, preserving as much information as possible from it. It accomplish this task by learning a new representation of data with lower dimensionality than the initial one and whose elements have no linear correlation. It performs a projection of data on to this new space, created its axes lying on the directions along which data variance is larger.

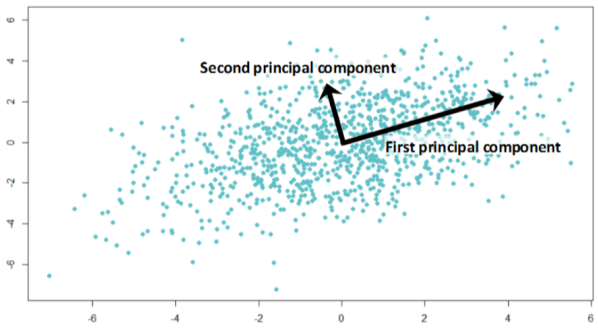


Figure 3.7: *Example of the first two principal components in a 2D dataset: the first principal component is placed along the direction where variance is greater, and the second component is in the orthogonal direction.*

PCA seeks to find an orthonormal basis so that the first base vector corresponds to the direction along which the variance of data is maximum. This vector is called first principal component (PC). The second principal component is defined as the vector orthogonal to the first, that explains the most variance left once the first component is removed. And so on, the i^{th} PC is the direction orthogonal to the first $(i-1)^{\text{th}}$ vectors that maximize the left variance. In figure 3.7 is reported a graphic example of two PCs extracted from a set of two-dimensional data.

Principal components are calculated as the eigenvectors of the covariance matrix, and the most popular way to implement this calculation is through Singular Value Decomposition (SVD) of the matrix of input data. SVD is preferred over the simpler calculation of the covariance matrix and its eigendecomposition because there are algorithms that can deal more quickly with SVD and avoid the explicit calculation of the covariance matrix.

Concretely, if we consider a set of n input data vectors lying in a space $\Re^m$, we can represent them as a matrix $X \in \Re^{n \times m}$ where n is the total number of input data and m is the dimensionality of each data (the number of features in each data vector).

We can suppose, without loss of generality, that feature distributions across data have zero mean, so that for each feature f_i with i = 1, ..., m, $\mu_i = \mathbb{E}[f_i] = 0$.

The covariance matrix $\Sigma \in \Re^{m \times m}$ of input data X is given by

$$\Sigma = \frac{1}{n-1} X^T \cdot X \tag{3.10}$$

because each entry can be written as $\sigma_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$, and, in the hypotesis of zero mean $\mu_j = \mu_k = 0$, we obtain $\sigma_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(x_j^{(i)} x_k^{(i)})$.
Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal of Σ we actually have the variances of each feature. And since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

We are interested in finding a new basis such that covariance matrix is diagonal in this basis and then perform a rotation of data using this basis. To find it, PCA makes use of Singular Values Decomposition of X.

SVD is basically a factorization of a $n \times m$ matrix X that (in the case X is a real matrix), allows to rewrite it as $X = USW^T$ where U and W are respectively $m \times m$ and $n \times n$ orthogonal matrices whose columns are called *left* and *right singular vectors* of X, and S is a $m \times n$ rectangular diagonal matrix. Diagonal values of S: s_i are uniquely determined by X and are called *singular values* of X.

Using this decomposition it is possible to write

$$\begin{aligned} X^T X &= (USW^T)^T (USW^T) \\ &= WS^T U^T USW^T \\ &= WS^2 W^T \end{aligned} \tag{3.11}$$

where we used the definition of orthogonal matrix for U: $U^T U = I$.

We therefore rewrite the covariance matrix

$$\Sigma = \frac{1}{n-1} WS^2 W^T. \tag{3.12}$$

which means that the singular values of X s_i are related to eigenvalues of the covariance matrix by the relation $\lambda_i = s_i^2/(n-1)$, while the right singular vectors of X represents the eigenvectors of the covariance matrix.

Using this result, we consider the transformation of data given by Z = XW, to show that with these rotated data, the covariance matrix is diagonal.

$$\begin{aligned} Var[Z] &= \frac{1}{n-1} Z^T Z \\ &= \frac{1}{n-1} W^T X^T XW \\ &= \frac{1}{n-1} W^T WS^2 W^T W \\ &= \frac{1}{n-1} S^2 \end{aligned} \tag{3.13}$$

This shows that if we use right-singular vector of X to perform the transformation, the covariance matrix of the transformed data is in a diagonal form.

To actual reduce the dimensionality of our data, we need to extract the first $\tilde{m} \leq m$ eigenvalues from the covariance matrix, order them in a descending order of magnitude, and collect the corresponding eigenvectors from the matrix W.

With them we can construct the projection matrix $\tilde{W}_{\tilde{m}} \in \Re^{m \times \tilde{m}}$ of the first $\tilde{m}$ eigenvectors, and use it to project data in order to obtain a new dataset $Y = X\tilde{W}_{\tilde{m}}$ made of n new data of dimensionality $\tilde{m}$.
An important parameters to quantify the amount of variance that PCA is able to explain, is the *variance explained ratio*, given by the ratio of an eigenvalue of the covariance matrix, and the sum of all the eigenvalues.

$$\frac{\lambda_i}{\sum_{k=0}^{m} \lambda_k} \tag{3.14}$$

The maximum number of principal components is limited by the number of data we can use to compute them. If we have a dataset of $n\_samples$ samples, each with $n\_features$ features, the maximum number of PC is given by $N\_pc = min\{n\_samples, n\_features\}$. This is because in a $n\_feature$-dimensional space, our points lie in a $n\_samples$-dimensional hyperplane and all the variance can be explained inside this subspace. For a better understanding of this concept, we can imagine a dataset made of only two data, each containing 3 features. If we represent these data, we construct a 3D space where each orthogonal axes corresponds to a feature, and we insert these two points, each corresponding to a data point. If we want to explain the maximum variance we need to find a new axis, the first principal component, but through two points passes just one straight line so we can at most limit our analysis to the plane passing through that line.

3.4 ML model performances assessment

After the model has been trained, it can be evaluated on a test set to determine its classification performances on a new dataset. A common metric for evaluating model performances is the *accuracy*: it is simply defined as the ratio between the total number of correct prediction and the total number of predictions (total number of data in the test dataset). For a binary classification, indicating as T and F, true and false, and P and N, positive and negative, it is possible to give the following definitions:

- TP: data classified as P, belonging to class P
- TN: data classified as N, belonging to class N
- FP: data classified as P, actually belonging to class N
- FN: data classified as N, actually belonging to class P

Using these definitions, accuracy is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.15}$$

Even though accuracy may seem a good parameter, it is not the most accurate one, when we're dealing with an unbalanced test dataset. As an example, in a binary classification concerning classes A and B, we can be in a situation where a model is not able to make distinctions between the two classes, and classifies all the data as belonging to A. If we test this model using a dataset consisting on 10 samples, with just 2 of them belonging to B and the rest to A, we would get an accuracy of 80 % , which is not a thruthful result.

For this reason other ways to evaluate performances of a network are often used, which overcome this problem. A possible alternative is the use of two quantities: precision and recall. Precision is the ratio between true positive and total positive classified cases (TP + FP), which is a measure of how many positive predicted cases are actually positive. Recall measures the percentage of actual positives that were correctly classified, or in other words it represents the true positive rate.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{3.16}$$

Unfortunately precision and recall are linked in a sense that often enhancing one leads to the decrease of the other and vice-versa. It can be set a trade-off between recall and precision according to what quantity is more crucial for the classification task we are performing. For example we could be willing to take the risk to have more false positive, in order to achieve a greater number of true positive.

To reach a threshold of positive missing $< x\%$ means to set the recall to (100-x)%. This operation of establishing a threshold is usually referred to as "setting the operating point". This threshold though is not always set at the beginning, since the best operating point is not always clear a priori. For this reason, what is done is to study the model under all the possible thresholds, and plot results creating a curve called precision-recall curve, an example of which is reported in figure 3.8. The closer a curve lies on the upper right corner (high precision and high recall), the more correctly the model is working. One way to summarize the information of a precision-recall curve can provide us, is to compute the area under the curve, known as *average precision*

Similarly to the precision-recall curve, a other curve is usually employed to study the effect of different thresholds. It is called the *Receiver Operating Characteristics curve*, usually referred as ROC curve. The ROC curve is constructed using the true positive rate (TPR) corresponding to the recall and the false positive rate (FPR) defined as $TPR = \frac{FP}{FP+TN}$ The ROC curve shows the evolution of TPR vs FPR for different thresholds. The ideal curve would be close to the top left (high TPR and low FPR) and the less accurate is the model, the more this curve tend to lay down to the bisector line. To summarize the model performances with a single number using ROC curve information, we compute the Area Under the Curve (AUC). The reference value for an AUC is 0.5 which is obtained with a model that is just randomly predicting and it corresponds to a curve lying on the bisector line.

The AUC can be regarded as the probability that a randomly picked point from the positive class, will have a higher score (according to the model) than a randomly picked point from the negative class. In other words, the percentage of the AUC value is an estimate of the probability that the model is able to distinguish between the two classes. An example of ROC curve is shown in figure 3.9.

3.5 Cross validation procedures

When dealing with small datasets, dividing them into train and test can be a non trivial issue because of the shortage of data.

A possible solution can be the implementation of a procedure called k-fold cross validation. It consists on the creation of k different partitions of the main whole dataset. From these partitions, k-1 are used as train, and the last is used as test. Every subset of partitions is used so that each different partition is used as test, while the others are used as train. Thus, for each fold there are two subsets (a train and a test) of the original dataset. Following this procedure, at each fold, the train dataset is different, and the model is tested on a different dataset each time. This process is described in 3.10.

In practice what is usually done, is not a sequential partitioning the dataset as shown in figure 3.10 but a shuffled partition of data. This avoids the creation of folds containing all the same label in case the dataset was grouped according to the label.

This process is similar to the creation of k different models, each one trained on a different partition (k-1 folds) and tested on the remaining k-th fold. We evaluate each one of these models, and we take as a result the average score across all the outputs.

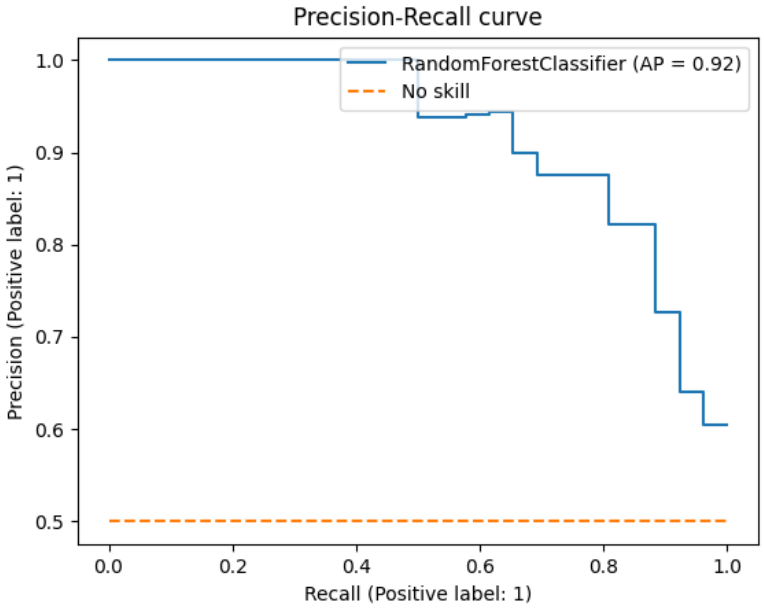


Figure 3.8: *Precision-recall curve (blue), representing the values of precision and recall for different thresholds values. Classification is accomplished with a Random Forest classifier, and the the Average Prediction value is also displayed in legend. Dashed orange the curve representing a model unable to distinguish between classes.*

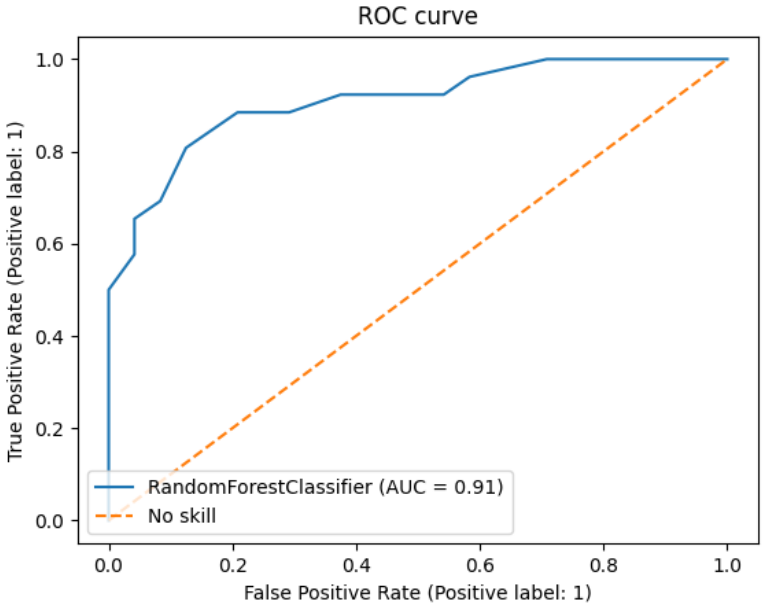


Figure 3.9: *ROC curve (blue) representing values of true positive and false positive rates for different thresholds values. Classification is accomplished with a Random Forest classifier, and the value of the Area Under the Curve is displayed in legend. Dashed orange the curve representing a model unable to distinguish between classes.*

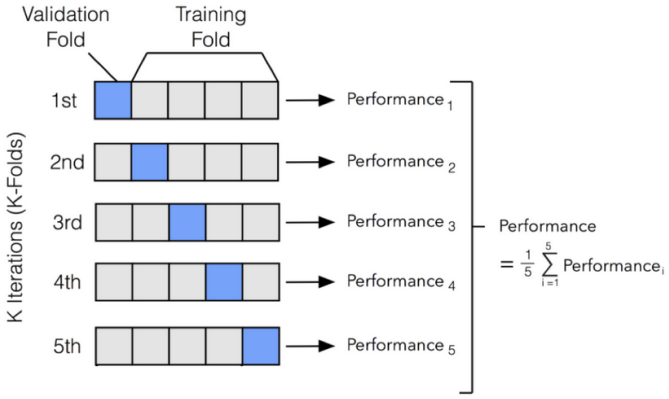


Figure 3.10: *Schematic representation of a k-fold cross validation procedure with $k = 5$: Dataset is partitioned into 5 subsets, four of which are used for training and the remaining one for the testing dataset. during each iteration a model is trained over 4 different folds and tested on the remaining one, until each fold has been used at least once as part of training and once as test.*

Chapter 4

Explainable AI: a game theory approach

As methods to learn pattern from data becomes more complex, they become harder to interpret. Deep learning represents an example of a technique to search for nonlinear relations between data, but introducing nonlinearity, it makes results difficult to be explained. Because of this, it is not uncommon to consider a machine learning model as black box where results are generated without any knowledge of the inner processes that produced them. However, in order to obtain more reliable results, it is a crucial issue to get rid of the black box idea and provide an explanation of the main features that characterized the output of a model.

One state-of-art explainatory algorithm is called SHAP, and it is built making use of an important result from game theory. For this reason, in the following section, we briefly discuss some key concepts related to this field, that were subsequently readapted, for the implementation of this machine learning explainatory model.

4.1 Shapley values

Game theory is a branch of mathematics related to the study of mathematical models to conceive social situations among competitive players [26].

It had a great development in the XX century especially during and after the Second World War thanks to the contribution of mathematicians like John Von Neumann, John Nash and Lloyd Shapley. Game theory mainly focuses on two major research areas: non-cooperative and cooperative games.

- Non-cooperative games concern competition between individual players who do not cooperate with each other and cannot form coalitions. The main task in non-cooperative games can be summarised as the search for a good strategy for each player. The objective of each player is in fact the maximization of his own utility function. A big contribution to the development of this theory came from von Neumann, or John Nash with the concept of Nash equilibrium [27].

- Cooperative games (or coalition games) concern competition between groups of player forming coalitions. Each coalition plays as a single participant and earns a payoff. One of the the main tasks related to cooperative games is to find a way to divide the total utility among the member of a coalition in an equally way proportionally of how much they contributed to the final score.

In 1951, Lloyd Shapley introduced a way to compute the exact amount of payoff for each player, making use of what was named after him: Shapley values [28][29]. Shapley introduced those values in the field of coalition games, therefore to properly understand his work, we need to briefly illustrate what a coalition game consists in.

A coalition game involves N players and different subsets, called coalitions, created from these N players. Each subset of players S gains a payoff at the end of the game. A function ν maps every subset to its payoff $\nu(S) = \text{payoff(S)} \in \Re$. On the basis of which player had more influence in this final score, we ask how to split this payoff in a fair way among the players of the subset, where "fair" is to be intended as, proportional to one's own contribution. A solution for this problem comes from **Shapley values** $\phi_i(\nu)$, specific for each player $i \in N$ in a coalition $S \subseteq N$. The idea behind them is the marginal contribution of that player to the final score, where marginal contribution is defined as the difference on the score of the coalition when player i joins the coalition. In other words they are the difference between the coalition score with player i and the coalition's score without him (*marginal contribution*$= \nu(S \cup \{i\}) - \nu(S)$) [30].

Shapley defined these coefficients (Shapley values) for a player i as a weighted average of marginal contribution values, over all the possible subsets that include player i.

The mathematical formulation that Shapley introduced for $\phi_i(\nu)$ is[1]

$$\begin{aligned} \phi_i(\nu) &= \frac{1}{N} \sum_{S \subseteq N \setminus \{i\}} \binom{N-1}{|S|}^{-1} [\nu(S \cup \{i\}) - \nu(S)] \\ &= \sum_{S \subseteq N \setminus \{i\}} \frac{S!(N-1-S)!}{N!} [\nu(S \cup \{i\}) - \nu(S)] \end{aligned} \tag{4.1}$$

which exactly represent the amount of reward for each player i.

[1]We denote as $S \cup \{i\}$ a subset of players including i and as $N \setminus \{i\}$ the set of all the players excluding i.

MATERIALS & METHODS

Chapter 5

Dataset: ABIDE I & ABIDE II

The data we are going to work on in this Thesis belong to the ABIDE dataset (Autism Brain Images Data Exchange), which is a project founded with the aim of investigating ASD using structural magnetic resonance images and resting state fMRI scans.

The whole ABIDE dataset was published in two releases: ABIDE I released in August 2012 and containing 1112 subjects scans, and ABIDE II released in June 2016 containing 1114 scans. For each site ASD was diagnosed either by gold standard diagnostic tests, clinical judgment or a combination of clinical gold standard procedures.

ABIDE I includes scans collected from 17 different sites, and the 1112 subjects consist on 539 patients with ASD and 573 typical control patients. ABIDE II includes scans collected from 19 different sites, and the 1114 subjects consist on 593 patients with ASD and 521 typical control patients. Not every site belonging to ABIDE II is different from those of ABIDE I, but, even though some medical centers are the same, the acquisition pipeline and parameters may have been changed during the time interval between the two releases, so in all our analysis, they are regarded as different acquisition sites.

Furthermore, even within a single release, such as ABIDE II, there are sites that released two samples of data. For this reason, some of these samples are labeled with a subscript number (e.g. _1 or _2). For a better understanding of the different sites and samples, and the main acquisition parameters employed, a list is reported in table 5.1. In this table, the acronym of each site and sample, which is employed to label them in the following figures, is associated to the actual site name.

In addition to scan images, ABIDE provides information related to each subject, such as age, sex, Full Intelligence Quotient (FIQ), eye status during the scan (open or closed), and every additional clinical information provided by patients.

The vast majority of patients are males as shown in figure 5.1a, for a total amount of 1804 males and 422 females. The number of males is greater than the number of females as a consequence of the greater probability for male to be affected by ASD. The ratio between males and females has been estimated to be approximately 4:1 [31]. The number of control subjects and ASD patients is more or less balanced for every site, with the exception of KKI_1 that provided two times more controls than ASD patients, and KUL_3 and NYU_2 that only provided ASD cases. For a visual comparison, the number of controls/ASDs for each site is displayed on the histogram in figure 5.1b.

Patients in ABIDE dataset have ages ranging from 4 to > 50 years old, but as shown in the distribution in figure 5.2a the vast majority of participant are younger than 40, precisely > 97 % of participant are under 40 y.o., also, the majority of sites provide only young patients in a restricted age range, but as can be seen from figure 5.2b there are some sites that acquired patients with a wide age range (MAX_MUN or BNI_1 for example).

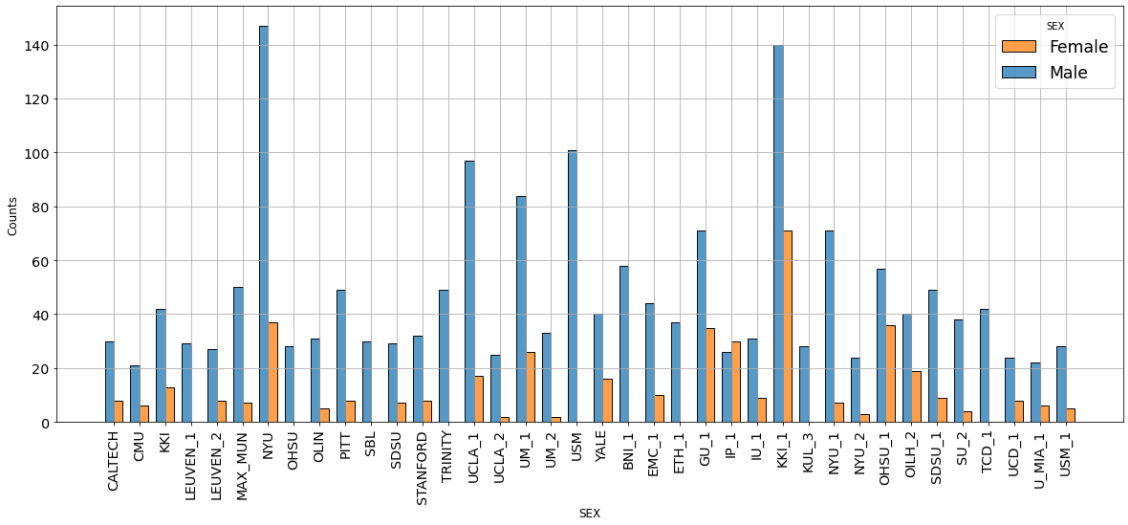
Table 5.1: *Table of the sites included in ABIDE dataset, with the corresponding acronym labelling the site and the sample For each site, the full name of the center, the belonging dataset and the number of subjects provided (total, ASD and controls) are shown. In addition the scan type, the echo time TE, the repetition time TR and the static magnetic field B_0 are provided.*

Acronym	Name	ABIDE	Total-ASD-TC	Scanner name	TE [ms]	TR [s]	B_0 [T]	
CALTECH	California Institute of Technology	1	38-19-19	Siemens TrioTim	30	2	3	1
CMU	Carnegie Mellon University	1	27-14-13	Siemens Verio	30	1.5	3	2
KKI	Kennedy Krieger Institute	1	55-22-33	Philips Achieva	10	2.34	3	3
LEUVEN_1	University of Leuven	1	29-14-15	Philips Intera	33	1.68	3	4
LEUVEN_2	University of Leuven	1	35-15-20	Philips Intera	33	1.65	3	5
MAX_MUN	Ludwig Maximilians University Munich	1	57-24-33	Siemens Allegra	30	3	3	6
NYU	NYU Langone Medical Center	1	184-79-105	Siemens Allegra	15	2	3	7
OHSU	Oregon Healt and Science University	1	28-13-15	Siemens TrioTim	30	25	3	8
OLIN	Olin Neuropsychiatry Research Center	1	36-20-16	Siemens Allegra	27	1.5	3	9
PITT	University of Pittscurgh School of Medicine	1	57-30-27	Siemens Allegra	25	1.5	3	10
SBL	Social Brain Lab	1	30-15-15	Philips Intera	30	3.2	3	11
SDSU	San Diego State University	1	36-14-22	GE MR750	30	2	3	12
STANFORD	Stanford University	1	40-20-20	GE Signa	30	2	3	13
TRINITY	Trinity Centre for Health Sciences	1	49-24-25	Philips Achieva	28	2	3	14
UCLA_1	University of California Los Angeles	1	82-49-33	Siemens TrioTim	28	3	3	15
UCLA_2	University of California Los Angeles	1	27-13-14	Siemens TrioTim	30	2	3	16
UM_1	University of Michigan	1	110-55-55	GE Signa	30	2	3	17
UM_2	University of Michigan	1	35-13-22	GE Signa	30	2	3	18
USM	University of Utah School of Medicine	1	101-58-43	Siemens TrioTim	28	2	3	19
YALE	Yale Child Study Center	1	56-28-28	Siemens TrioTim	25	2	3	20
BNL_1	Barrow Neurological Institute	2	58-29-29	philips ingenia	25	3	3	21
EMC_1	Erasmus University Medical Center Rotterdam	2	54-27-27	GE MR 750	30	2	3	22
ETH_1	ETH Zurich	2	37-13-24	Philips Achieva	25	2	3	23
GU_1	Georgetown University	2	106-51-55	Siemens TrioTim	30	2	3	24
IP_1	Institute Pasteur & Robert Debrè Hospital	2	56-22-34	Philips Achieva	45	2.7	1.5	25
IU_1	Indiana University	2	40-20-20	Siemens TrioTim	28	0.813	3	26
KKI_1	Kennedy Krieger Institute	2	211-56-155	Philips Achieva Ds	30	2.5	3	27
KUL_3	Katholieke Universiteit Leuven	2	28-28	Philips Achieva Ds	30	2.5	3	28
NYU_1	NYU Langone Medical Center	2	78-48-30	Siemens Allegra	78	5.2	3	29
NYU_2	NYU Langone Medical Center	2	27-27	Siemens Allegra	78	5.2	3	30
OHSU_1	Oregon Healt and Science University	2	93-37-56	Siemens TrioTim	30	2.5	3	31
OILH_2	Olin Institute of Living Hartford	2	59-24-35	Siemens Allegra	27	1.5	3	32
SDSU_1	San Diego State University	2	58-33-25	GE MR750	30	2	3	33
SU_2	Stanford University	2	42-21-16	GE Signa	30	2	3	34
TCD_1	Trinity Centre for Health Sciences	2	42-21-21	Philips Achieva	27	2	3	35
UCD_1	University of California Davis	2	32-18-14	Siemens TrioTim	24	2	3	36
U_MIA_1	University of Miami	2	28-13-15	GE Healtcare	30	2	3	37
USM_1	University of Utah School of Medicine	2	33-17-16	Siemens TrioTim	28	2	3	38

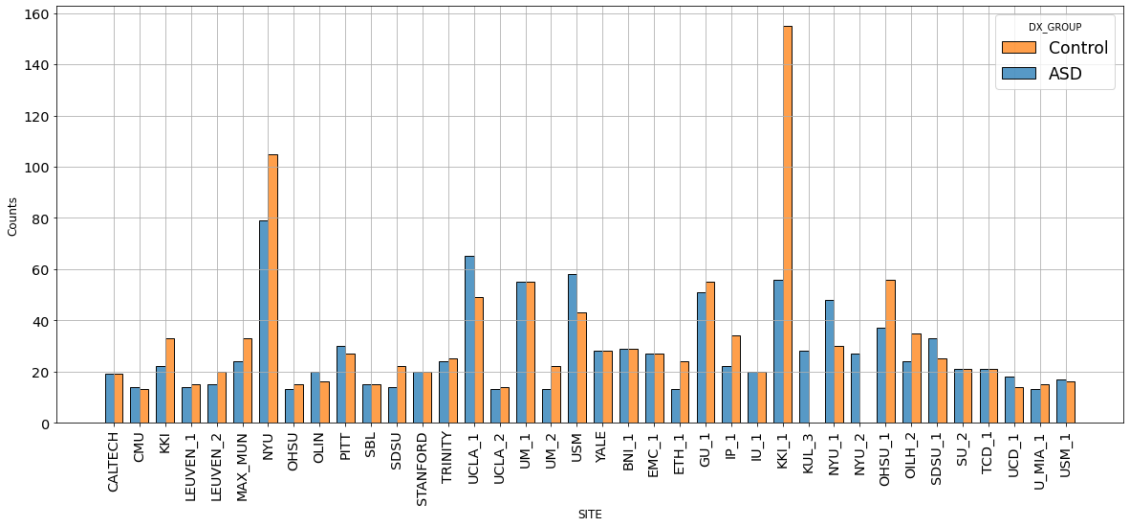
Information about full intelligence quotient, whose distribution is shown in figure 5.3a, are not provided for every participant, in fact 171 patients out of 2226, coming from sites UM1 and EMC1 (figure 5.3b) lack this information. In our further analysis, before proceeding these values were replaced by the average value of all the other values.

The lack of a common acquisition protocol is also evident from the eye status at scan. As shown in figure 5.4a each site acquired scans either with open eyes or with closed eyes, without a common procedure, and sometimes this information is not even specified. Overall, the whole dataset consists of more than 70% of patients acquired with open eyes, as shown in figure 5.4b.

Considering all the information above, we can limit our further analysis in order to work on a more homogeneous dataset. In this way, we try to remove some source of variability due to unavoidable differences due to sex or age. We have then carried out our analysis on a dataset consisting on only male subjects with an age within 5 and 40 years. Some further analysis were performed with further constraints on eye status at scan, selecting only patients with open eyes.

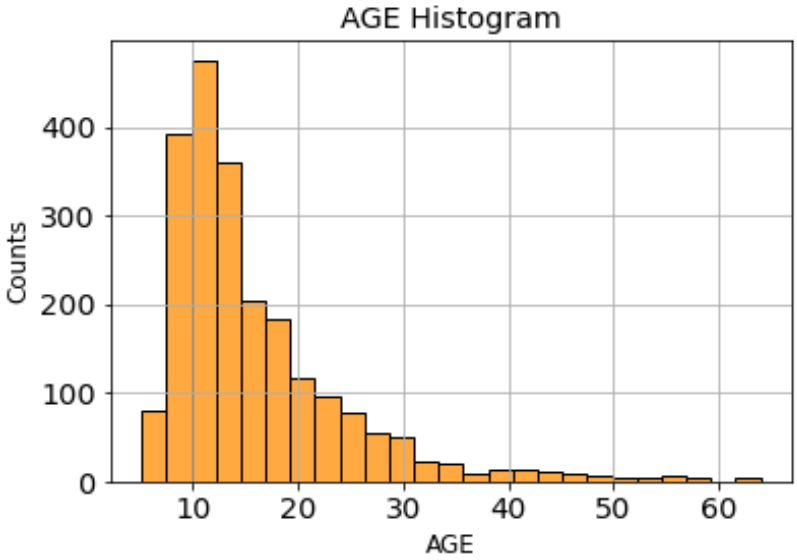


(a)

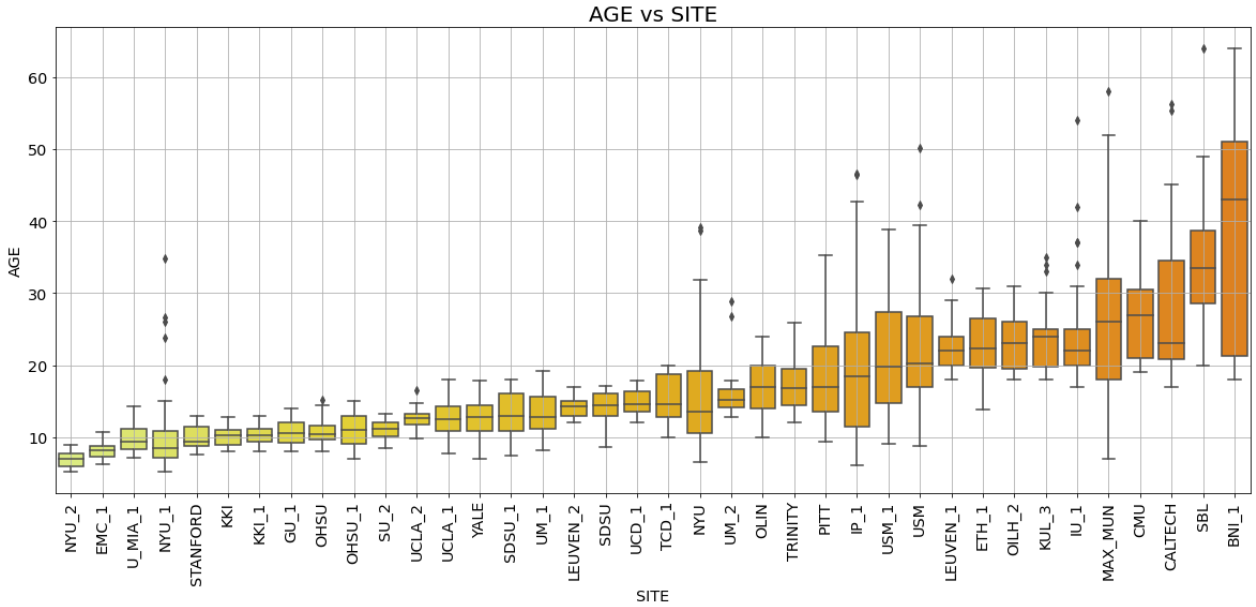


(b)

Figure 5.1: *Distributions of male and female subjects per site (a) and control and ASD subjects per site (b), belonging to ABIDE I and ABIDE II datasets.*

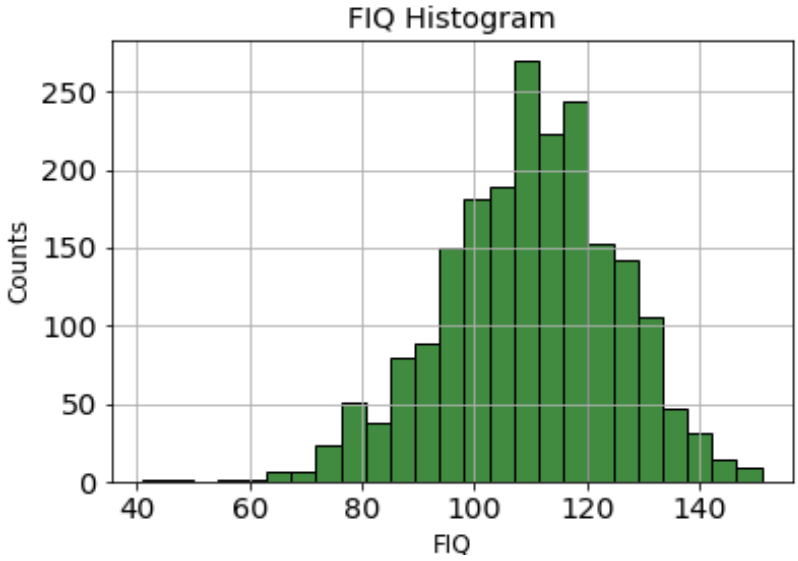


(a) *Ages distribution across the whole ABIDE dataset.*

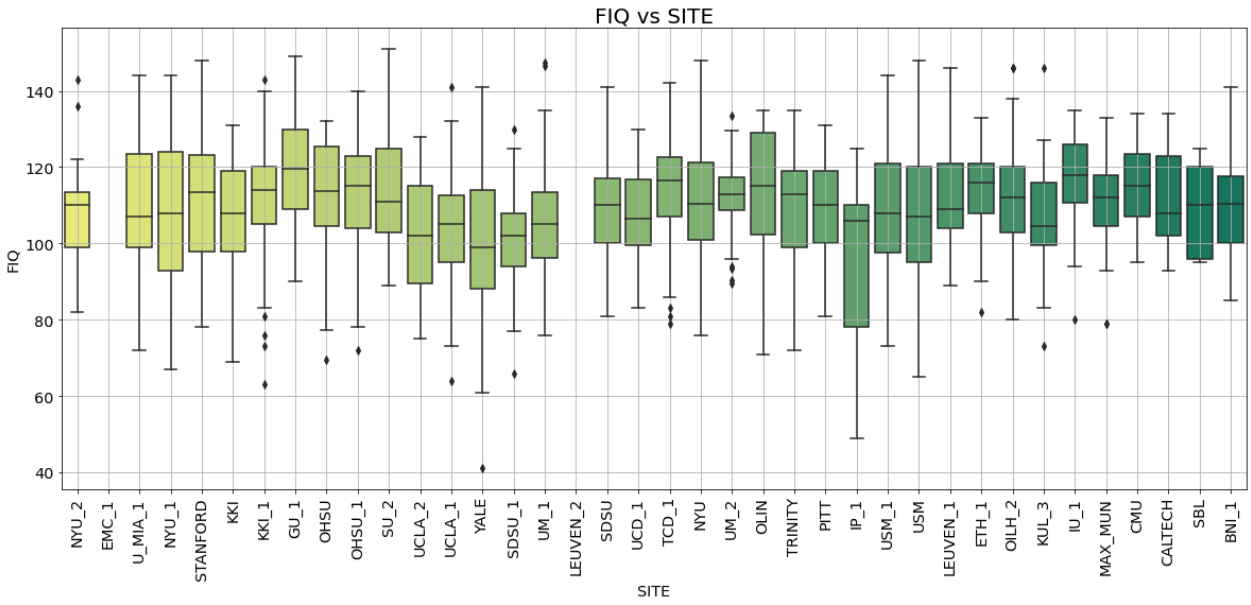


(b) *Boxplot of ages per sites, sorted by increasing median age.*

Figure 5.2: *Histogram and boxplot distribution of the ages of subjects belonging to the ABIDE dataset (jointly controls and ASD).*

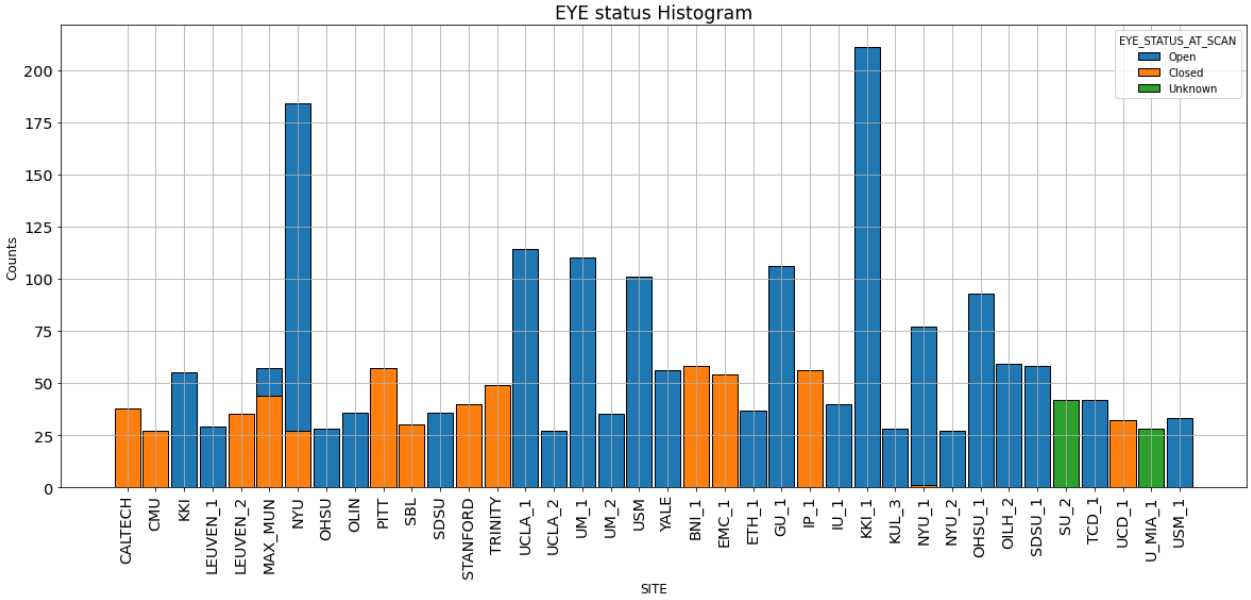


(a) FIQ distribution across the whole ABIDE dataset

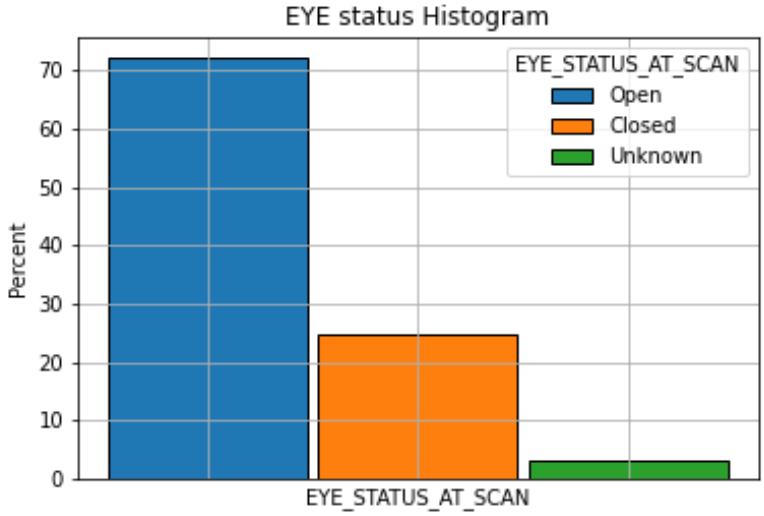


(b) Boxplot of patient's FIQ per site

Figure 5.3: *Histogram and boxplot distribution of FIQ of subjects belonging to the ABIDE dataset (jointly for controls and ASD)*



(a) *Eye status distribution per site.*



(b) *Overall eye status distribution on the entire dataset*

Figure 5.4: *Histograms showing distributions of open (orange), closed (green) and unknown (blue) eye status at scan of subjects belonging to the whole ABIDE dataset.*

Chapter 6

Image preprocessing

6.1 Common preprocessing steps

Data acquired after a MRI and fMRI session are not ready to be analyzed to perform functional connectivity measures or meaningful statistical analyses. In fact data are usually affected by different sources of noise and artifacts. It is possible to distinguish between two main sources of noise: physiological noise due to normal biological functions of patients such as heartbeat and breathing, and movement of subject inside the scanner. Additionally it is possible to identify artifacts due to inner flaws of the scanning systems such as inhomogeneities of the static magnetic field or drift of the baseline of the BOLD signal [32].

In this section we briefly review some of the most important artifacts and the common procedures to reduce their effect on the acquired images. We also show other common preprocessing steps applied to EPI data to prepare them for subsequent analyses such as functional connectivity measures.

- **Motion correction** steps are usually applied at the beginning of image preprocessing. They aim to reduce head motion effects due to physical movements of the patient inside the scanner. These movements usually result in a misalignment from one acquired volume to the next. Motion correction works by spatially applying transformations such as volume by volume rotation or translation. These transformation are aimed to overlap every acquired slice to a chosen reference volume, like the first or the middle one.

- **Slice timing correction** is usually performed as well. It aims to correct artifacts deriving from the sequentiality of acquisition for each slice of the brain, due to which each slice is acquired at different time. In EPI images, the entire time elapsed to acquire all the slices is called repetition time, and it usually ranges from 1 to 3 seconds. During this interval signal may lose its initial strength. Slice timing correction uses interpolation in time to shift the BOLD timeseries of each voxel, in order to align them to a reference starting time. A downside of the use of interpolation, though, is the signal smoothing it performs that can lead to a slight loss of high frequencies information.

- **Distortion correction** is usually applied to reduce distortions in EPI images caused by inhomogeneities of the magnetic field. Inhomogeneities usually arise from flaws of the external magnetic field, or are caused by different tissues within the brain whose contiguity causes localized distortions. They are usually corrected during the image acquisition procedure by means of electromagnets called *shims* coils used to generate a magnetic field that opposes and cancels these inhomogeneities. Yet, this procedure, often referred as *shimming*, is not

always able to cancel all the inhomogeneities, and the remaining ones are corrected during preprocessing. The correction is possible by using a *fieldmap*: a map containing magnitude information of the magnetic field across the space. Fieldmaps sould be acquired after each scan, but in absence of them, they can be generated using tools such as SPM [1]

- A further common step is **spatial smoothing** of both structural and functional data. It operates calculating a weighted average of each pixel over neighbor voxels. To this end, a Gaussian kernel with a chosen FWHM is applied to create a smoothing window. Spatial smoothing is useful to avoid abrupt changes of signal between two neighbouring voxels.

- **Band-pass temporal filtering** is commonly applied to BOLD data, aiming to reduce artifacts and noise. By applying a high-pass filter it is possible to correct for an effect called *drift*. It consists on the slow change of the BOLD baseline signal over time. High-pass filtering is usually referred as removal of linear trends, since are removed from a signal only components with a frequency lower than the biological low-frequency fluctuations of BOLD signal. At the same time, a low-pass filter is applied to remove all the frequencies above a cut off frequency. It is commonly applied in processing resting state fMRI data because the physiological signal is driven by low frequencies oscillations while high frequencies are associated to noise.

All the preprocessing steps cited above are performed for each patient, and their main common objective is to enhance the signal to noise ratio of each image. However, if our task is to perform a group analysis, and compare different patient images, a further, essential step called **registration** has to be carried out.

During registration structural (T1-weighted) data are aligned over a standard coordinates system space. This allows to universally describe the location of the different brain parts, to make sure that the same voxel coordinate corresponds to the same brain area for all the subjects. Registration occurs for functional images as well: they are firstly aligned and adapted to the structural image and then registered to the same standard template. The most common template, provided by packages like FSL is Monreal Neurological Institute's MNI152. It was adopted by the International Consortium of Brain Mapping (ICBM) as the international standard, replacing the previous Talairach and Tournoux template. Because of the adoption as international standard it is also called ICBM152, however the most common name remains MNI152.

To understand how MNI152 differs from the Talairach and Tournoux we have to spend a few words about the latter [33].

[1] https://www.fil.ion.ucl.ac.uk/spm/toolbox/fieldmap/

From Talairach-Tournox to MNI152 template
The Talairach and Tournoux space was published in 1988 and introduced some innovative aspects to tackle the problem of the great variability in brain anatomy between people. Until then this issue had limited the accuracy of statistical analyses. They introduced a common coordinate system to identify different brain locations, based on some anatomical landmarks, and presented spatial transformations to make different brains match each other. They chose two anatomical areas as reference areas: the anterior and the posterior commissure, which are relatively invariant between different brains and conjuncted them with an axis. This axis constitutes the horizontal y axis, and the origin is placed in correspondence of the anterior commissure. From it, a perpendicular axis passing thgough the interemispheric fessure is chosen as the z axis and consequently the x axis is chosen perpendicular to y and z. This way they created a 3D coordinate system called the Talairach coordinate system. Afterwards, to match different brains they described a set of spatial transformations, one for each different brain quadrant, to transform a brain in order to match the principal anatomical structures of each other.
From this template, a first MNI template was created, called MNI305, created by manual scaling 241 brains to the Talairach template and averaging them to obtain new template, and then transforming 305 additional scans to this new template and averaging them to create a second average and final template (MNI305).
From this template, MNI152 was finally created and published in 2001, by registering 152 T1 scans to the MNI305 template and averaging them [33].

This template is used both for the structural and the functional registration of a MRI and fMRI scan.

Looking at figure 6.1 we can see an example of functional and structural image registration to a MNI152 template: firstly the functional image is registered to the structural and next, they are both registered to MNI template.

Once the structural and functional images have been registered to a standard space, is possible to extract brain region information using an **atlas**.

Atlases are in the same space as the template image (MNI or Talairach space for example) and consist of a 3D standard brain template where different brain areas are associated to a label. This division into areas and the subsequent labeling, is called *parcellation*.

There are different atlases, each one including a different parcellation of the brain, which is obtained by dividing it into N labeled ROIs (Regions Of Interest) to focus on anatomical and/or functional regions, according to the study to carry out. Some of the most popular atlases are:

- The H-O atlas is an atlas firstly employed in 2006 [34] and consists of two parts: a subcortical and a cortical atlas, with a total of 117 ROIs of which 21 subcortical and 96 cortical (48 for each emishpere). A colored representation of the two (cortical and subcortical) parcellations is shown in figure 6.2.

- The Automated Anatomical Labeled is an atlas created in 2002 [35] that consisted of 90 total ROIs, 45 each emisphere. Since then, different modified and improved version were released, the last of which AAL3 consisting of 166 ROIs.

- CC200 and CC400 are more recent atlases created by C. Craddock in 2012 [36] for functional parcellation, consisting of 200 and 392 ROIs respectively [2].

- The Desikan atlas [37] originally created in 2006 and consisting on 68 regions also underwent different transformation towards a finer parcellation of the brain.

[2] http://ccraddock.github.io/cluster_roi/atlases.html

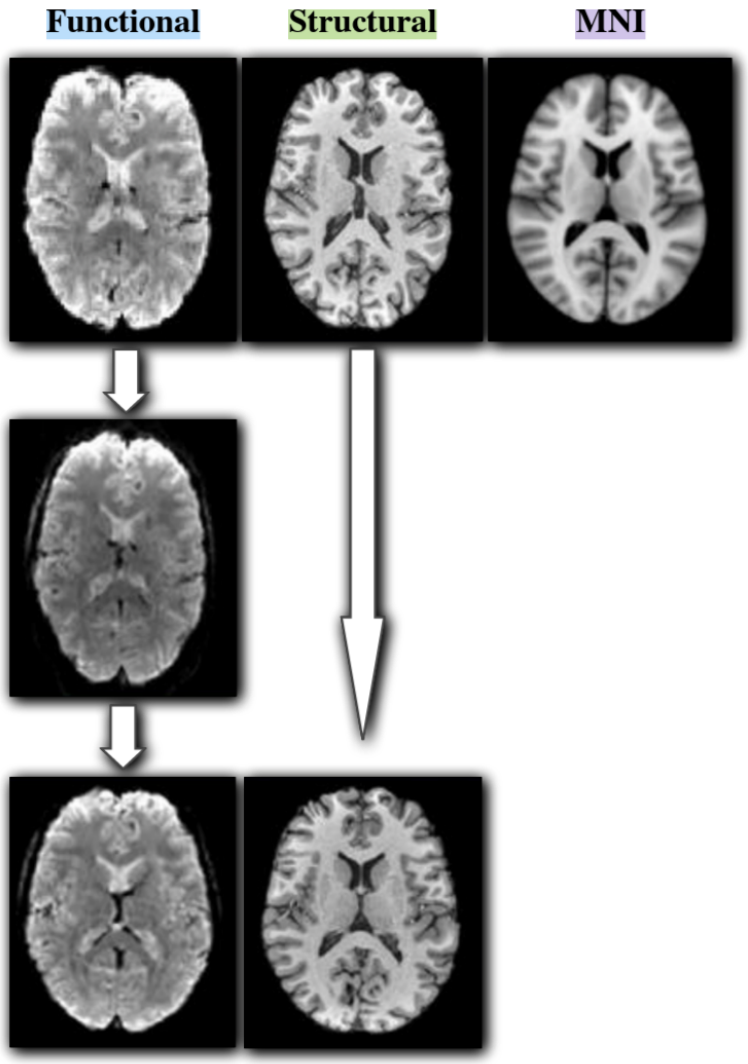


Figure 6.1: *Registration steps: starting from the first row there are the acquired functional and structural images and the MNI template respectively. The second row shows the functional image registered on the structural space. The third row shows the final images both registered to the MNI152 template.*

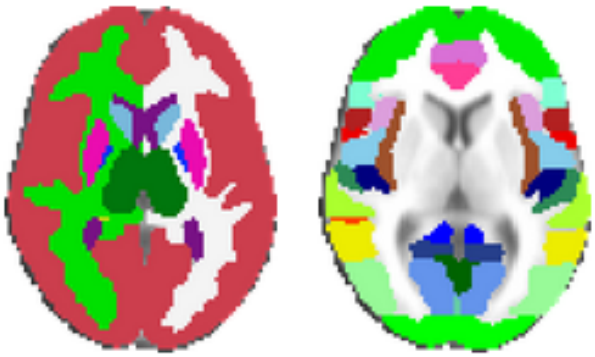


Figure 6.2: *Horizontal section of Harvard-Oxford subcortical (sx) and cortical (dx) atlased*

6.2 Preprocessing of ABIDE I & ABIDE II dataset

ABIDE-preprocessed initiative

In 2013 Neuro Bureau Preprocessing Initiative took care of data preprocessing of ABIDE I dataset and shared its results making them publicly available [38]. Thanks to their work, ABIDE I data are now available both as raw data and as preprocessed data. Four preprocessing approaches were employed each from a different team and each one publicly available [3] for download. The four pipelines for image preoprocessing employed for ABIDE I are:

- Connectome Computation System (CCS)
- Configurable Pipeline for the Analysis of Connectomes (C-PAC)
- Data Preprocess Assistant for Resting-State fMRI (DPARSF)
- Neuroimaging Analysis Kit (NIAK)

The preprocessing steps implemented by different pipelines are similar. However they differ on the programming language on which they rely (Python, MATLAB..) the algorithm implementation, the order of preprossessing steps and their parameters.

C-PAC the Configurable Pipeline for the Analysis of Connectomes

Preprocessed data are only available for the ABIDE I dataset. In order to perform analyses on a bigger dataset comprising ABIDE II as well, we need to run a preprocessing pipeline to create a unified dataset with data belonging to ABIDE I and ABIDE II, obtained following the same preprocessing steps.

In our work, we choose to process data from ABIDE I and ABIDE II using C-PAC: a configurable, open source pipeline, based on Nipype platform. We selected this software after a brief review of the most used pipeline among the four employed for ABIDE I. C-PAC and DPARFS resulted to be the most popular choices, however, we choose C-PAC since previous studies compared the different pipelines and found out that images preprocessed with C-PAC lead to a better classification control/ASD on ABIDE I dataset [39].

C-PAC was run on a Docker container installed on a personal computer with a hardware and software setup consisting on:

[3] http://preprocessed-connectomes-project.org/abide/C-PAC.html

- 16-core Intel i7-5960X processor and 64 Gb RAM
- Ubuntu 20.04 operating System
- C-PAC 1.8.1 installed on Docker v. 20.10.11

We were allowed to run 3 patients in parallel, reserving 4 cores for each partecipant and up to 12 Gb memory to each patient, necessary to save intermediate-steps outputs.

C-PAC employs tools like AFNI[4], FSL[5] and ANTS[6] to perform image correction of structural MRI and rs-fMRI.

Preprocessing of ABIDE I and ABIDE II data was carried out along the lines of what was accomplished on ABIDE I by the C-PAC team. This pipeline includes both anatomical and functional preprocessing. Anatomical pipeline steps consist on:

- Skull removal using AFNI's 3dSkullStrip;
- Tissue segmentation using FSL-FAST, to separate gray matter, white matter and CSF, using a thresholding probability map;
- Registration to a standard template using ANTS, with a spatial resolution of 2mm;

Functional pipeline consists on the following steps

- Slice timing correction using AFNI-3dTshift;
- Motion estimate and correction using AFNI-3dvolreg;
- Distortion correction using PhaseDiff and AFNI 3dQWarp;
- Create a brain-only mask of the functional data using AFNI 3DAUTOMASK;
- Band-pass filtering of (0.01 - 0.1) Hz;
- Time Series extraction using different atlases;

At the end of functional preprocessing steps, timeseries are extracted from each patient, making use of different atlases implemented in C-PAC, provided by different softwares as FSL for Harvard-Oxford (H-O), AAL Toolbox for the Automated Anatomical Labeled (AAL) atlas, pyClusterROI for CC200, CC400 etc.

T

According to a previous study [12] the atlas that gives best classification performances is Harvard-Oxford, so this is the atlas we chose to employ for our work.

The H-O atlas employed by C-PAC is provided by FSL library [7], and it includes both subcortical and cortical atlases. However, it only consists on 111 ROIs, namely 14 subcortical and 97 cortical. The lacking ROI in the subcortical areas are L/R cerebral white matter, L/R cerebral cortex, L/R lateral ventricle and the brain stem [8]. Removing these regions from the 117 ROIs we should obtain 110 ROIs, while the atlas employed in C-PAC has 111 regions. The extra cortical ROI in this H-O

[4]https://afni.nimh.nih.gov/
[5]https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslOverview
[6]http://stnava.github.io/ANTs/
[7]https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/
[8]Subcortical ROIs https://neurovault.org/images/1700/ Cortical ROIs https://neurovault.org/images/1705/

atlas is present in the 83th position and is named 3455; however there are no information abut this ROI neither on Harvard-Oxford documentation nor on C-PAC's/FSL's documentation, and because it is only made of 2 voxels it was excluded from our further analyses. In summary we are able to obtain 110 ROIs with Harvard-Oxford atlas, and for each ROI we are able to extract a timeseries using C-PAC.

As is possible to notice from figure 6.3, timeseries extracted after our preprocessing pipeline do not exactly match those from ABIDE preprocessed. Even if they follow a similar trend, there are some local differences between the two plots. This slight difference is most likely due to the differences in software version. Abide preproccessed data were obtained using one of the first version of C-PAC; nowadays, after more than 7 years, C-PAC and the libraries it relies on were upgraded several times and this may have slightly affected the final outcome of the process. We set our pipeline along the lines of the old pipeline configuration file employed on ABIDE preprocessed [9], but since it belongs to a previous version of C-PAC, it lack information about lots of sub-parameters and sub-settings added in these years. For this reason, in our pipeline configuration file, parameters in common between our file and ABIDE preprocessed' were set the same as ABIDE, and for all the others we choose to left the C-PAC's default values.

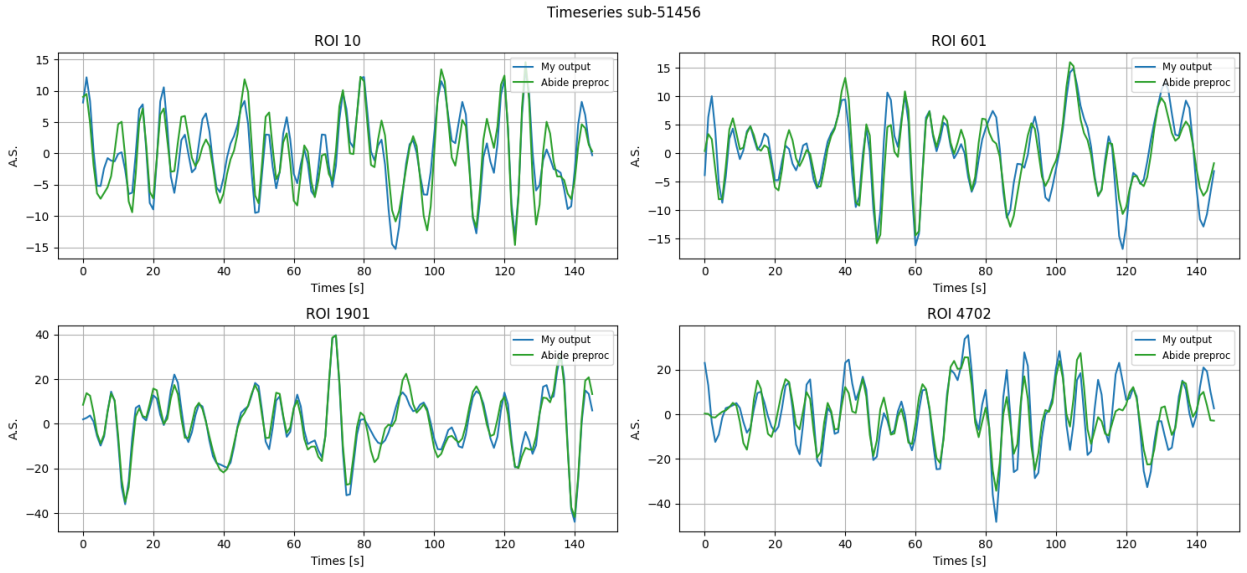


Figure 6.3: *Comparison of 4 timeseries between ABIDE-preprocessed and the ones obtained by our implementation of C-PAC. The four plots show four randomly chosen ROIs belonging to patient 51456 from ABIDE I. They corresponding to ROI: 10, 601, 1901, 4702 which are the corrisponding lables on the Harvard-Oxford atlas legend of regions: left thalamus, right inferior frontal gyrus, right supramarginal gyrus, left supracalcarine cortex.*

[9] https://github.com/preprocessed-connectomes-project/abide

Chapter 7

Connectivity measures

As described in chapter 6.1 choosing an atlas to employ on fMRI data, we set a brain parcellation, and we are able to extract timeseries of different brain areas. If we use Harvard-Oxford atlas, we are able to extract 110 timeseries for each patient. With these data, we want to construct a correlation matrix for each patient, and, by pairwise comparing timeseries, extract a coefficient representative of the functional correlation between two areas. We are only interested in correlation between timeseries belonging to different brain areas since correlation between a timeseries with itself would not carry any information. According to this, the total number of combination achievable with n timeseries is given by

$$N_{comb} = \frac{n \cdot (n-1)}{2} \tag{7.1}$$

Therefore, employing H-O atlas with 110 ROIs we obtain 5995 combinations each one expressed by a correlation coefficient.

In the following section we discuss two different approaches to accomplish a measure of connectivity by comparing two timeseries: the first makes use of Pearson correlation analysis, a useful tool to quantify the linear correlation between two timeseries, and the second approach, based on wavelet analysis, performs a comparison between two signals in time-frequency domain. We start discussing Pearson correlation coefficients in the following section and in the next section we describe the main traits of wavelet analysis and how we extracted a correlation coefficient from a time-frequency analysis of two timeseries.

7.1 Pearson correlation and Z-Fisher transform

Pearson correlation often simply called correlation coefficient is the measure of a linear relation lying between two sets of data x and y, is defined as the covariance of (x, y) over the product of the standard deviation of the two sets.

$$Corr(x, y) = r_{xy} = \frac{Cov\,(x, y)}{\sigma_x \sigma_y} = \frac{E\left[(x - \mu_x)\,(y - \mu_y)\right]}{\sigma_x \sigma_y} \tag{7.2}$$

Pearson correlation coefficients have the important property to be scale and magnitude invariant, since each timepoint is shifted by the average value of the timeseries and divided by its standard deviation. They can assume values between -1 and +1 where the extremes correspond to exact anti-correlation or correlation respectively. If a linear relation lies between the two sets of data, a high absolute value of Pearson coefficient indicates that the two series are highly (positively or negatively) correlated. [40]

Given two series $x = \{x_1, ..., x_n\}$ and $y = \{y_1, ..., y_n\}$ of length n, correlation can be easily computed by

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{n} (x_i - \mu_x)^2} \sqrt{\sum_{i=1}^{n} (y_i - \mu_y)^2}} \tag{7.3}$$

For each patient, Pearson correlation coefficient was computed for all the timeseries pairs.

Before further analyses, a common way to proceed is to transform each coefficient with Fisher z-transformation [12].

The Fisher transformation is defined as:

$$z = \frac{1}{2} ln \left(\frac{1+r}{1-r} \right) = arctanh(r) \tag{7.4}$$

The reason behind this common operation appears clear when we are dealing with highly correlated variables. When Pearson correlation distribution results in a highly skewed distribution, Fisher's transform sought to transform it into a normal distribution whose standard error is approximately constant, equal to $\sigma = \sqrt{n-3}$ where n is the total number of points, and it does not depends on the values of correlation [41].

Thus, this transformation allows to obtain a variable which is normally distribuited even when Pearson correlation coefficients follow a bivariate normal distribution In our data, this difference between the two distributions is not pronounced, one example where this difference is more evident is shown in figure 7.1, but since we are not working with highly correlated or uncorrelated variables, there are many other examples where pearson correlations already follow a Gaussian distribution and there is not much difference with z-score values distribution.

At the end of this analysis, we obtain, for each subject, a correlation matrix like the one shown in fig 7.2, referred to patient 20243 from the ABIDE I dataset.

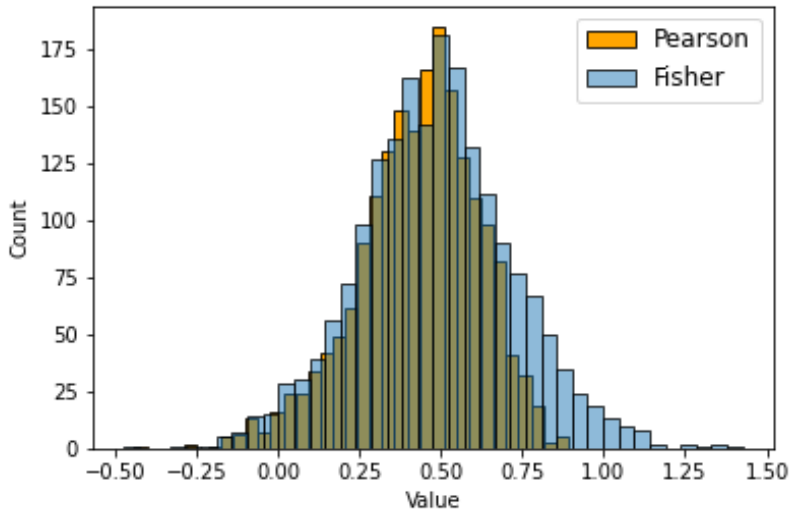
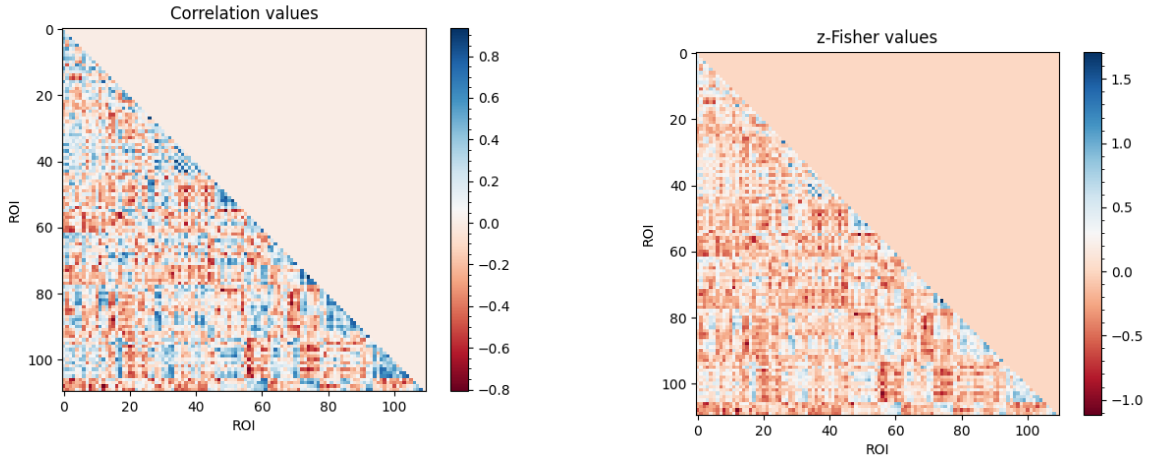


Figure 7.1: *Distribution of feature 3 (correlation between left caudate and left putamen): Pearson correlation values in orange and Fisher transformed values in light blue.*



(a)

Figure 7.2: *Correlation and z-values matrix computed from timeseries extracted using Harvard-Oxford atlas, from patient 20243 belonging to ABIDE I dataset.* ***Note:*** *The two figures are not on the same values scale because Pearson correlation coefficients have range [-1, 1], and z-scored coefficients have range* $(-\infty, \infty)$

7.2 Wavelet analysis

A different approach to measure a correlation between two timeseries is by making use of wavelet analysis [42] [43]. Wavelet transformation is a mathematical tool for analyzing time series or images, and provides a comprehensive way for investigating the bivariate relationship of a timeseries both in time (or space) and frequency domain.

To understand wavelet transformation it could be helpful to compare it with Fourier analysis. Fourier analysis allows to expand a periodic function x(t) into a series, an ideally infinite sum of weighted sines and cosines with different frequencies:

$$x(t) = a_0 + \sum_{k=1}^{\infty}(a_k cos(2\pi kt/T) + b_k sin(2\pi kt/T)) \tag{7.5}$$

In equation 7.5 terms corresponding to the k-th frequency are called *harmonics* and they are multiples of the fundamental frequency corresponding to k = 1.

The Fourier Transform (FT) is a natural extension of the Fourier series to aperiodic functions defined over the real axis. Aperiodic functions do not allow a discrete superposition of sines and cosines terms, and for this reason they need to be represented by a continue superposition. Fourier transform takes a function from space or time domain and turns it into spatial or temporal frequency domain. It is a complex function since sines and cosines terms can be represented by a complex exponential as shown below in equation 7.6.

$$\tilde{F}(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-i\omega t}dt \tag{7.6}$$

To deal with a discrete signal, for example a sequence x_n obtained from discrete samplings of a continuous signal x(t), is possible to make use of the Discrete Time Fourier Transform, which allows to write the $\tilde{F}(\omega)$ as in equation 7.7 which represent the discrete version of 7.6.

$$\tilde{F}(k) = \tilde{x}_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i k n}{N}} \tag{7.7}$$

The Discrete Fourier Transform, is useful to analyze discrete periodic signals. It acts on a function defined over a finite domain and returns a sequence of samples of the discrete Fourier transform that we can denote as $\tilde{x_k}$

In equation 7.7 N is the total number of points of the timeseries x_n. The frequencies at which samplings are computed are $\omega_k = 2\pi k/N$. The series we obtain from equation 7.7: $\tilde{x}_k$ is periodic, with a period equal to N.

One of the main drawbacks of FT though, is the lacking of spatial information, for example for a non stationary signal, is possible that the signal contains a variable component of frequency in different space or time locations.

A time-frequency analysis provides for this lacking by computing both frequency and spatial information from a signal, allowing to analyze non stationary signals and to obtain information both on time (or space) and frequency domain. Since we are working on signals defined over a time domain, from now on we would refer only to time and frequency domain, but we keep in mind that all the relation are true for spatial and spatial frequency domain as well.

The process through which we extract information about frequencies and time location is a convolution of the signal $x(t)$ with a function $w(t)$ as shown in equation 7.8.

$$\tilde{x}(a,b) = \int_{-\infty}^{\infty} x(t) w(t-b) e^{-2\pi i a t} dt \tag{7.8}$$

The function $\omega(t)$ is called *windowing function* and is usually centered at zero with a symmetric trend that rapidly decays towards zero. In a convolution, parameters describing frequency and time location are often simply called a and b,

The convolution term $w(t-b)e^{-2\pi i a t}$ changes according to the type of analysis we are performing. While the time parameter b acts the the same way through all the types of analysis, by simply shifting the windowing function throughout the entire timeseries $x(t)$, it is a that differentiates the most the different analyses.

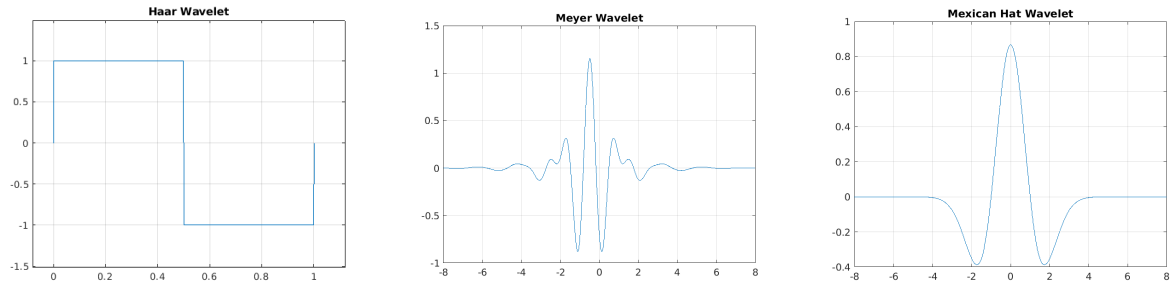
It is the frequency parameter and the way it is introduced in a convolution determines the developing of the transform.

We distinguish between two main families of time-frequency analyses, according to the convolution term employed and the effect of the parameter a:

1. Windowed Fourier Transform (WFT) employs a convolution term $\psi_{ab}(t) = e^{it/a}\phi(t-b)$ where $\phi(t)$ is a window function of constant width and the parameter a acts as frequency modulation (freq $\sim$ 1/a): the lower is a, the greater the number of oscillation inside the window $\phi(t)$.

2. Wavelet Transform (WT) employs a convolution term $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$ where ψ is a function called *mother wavelet*; a is a positive real number and defines the dimension of ψ: with a>1 we obtain a dilation and a<1 corresponds to a contraction. b is any real number (positives and negatives) and defines the location of the wavelet in time.

In Wavelet Transform, the equivalent to the window function in WFT is a wavelet function $\psi(t)$.



(a) ***(b)*** ***(c)***

Figure 7.3: *Visual comparison between three different mother wavelets. Haar wavelet (fig 7.3a) is a simple square wave function, mexican hat (fig 7.3c) is a wavelet belonging to the family of Gaussian functions and represents the negative normalized second derivative of a Gaussian function; Meyer wavelet (fig 7.3b) is a wavelet with applications in different fields like adaptive filters*

Continuous Wavelet Transform

A *wavelet*, literally "small wave" is a wave function limited both in space and period: it begins at zero, grows and decreases in a limited time period and returns to zero. With these properties, it is a function local in the temporal domain as well as in the frequency domain.

To be defined as so, a wavelet is required to satisfy two properties: to have zero mean (it must be oscillating) and to have unitary squared norm [44].

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \qquad \int_{-\infty}^{\infty} |\psi(t)|^2 dt = 1 \tag{7.9}$$

It is defined over the space L^2 of Lebesgue measurable functions that are both absolutely integrable and square integrable. In this space the two properties can be satisfied.

There are different wavelets that satisfy properties 7.9, such as Haar (fig 7.3a), Meyer (fig 7.3b) [45], Mexican hat (fig 7.3c) or Morlet (fig 7.4) [1].

What we are going to use in our analysis is Morlet wavelet, defined by equation 7.10 and shown in figure 7.4.

$$\psi\left(t\right) = \pi^{-1/4} e^{iw_0 t} e^{-t^2/2} \tag{7.10}$$

For each wavelet, there is a trade-off between time and frequency resolution: compressing the wavelet in a shorten time domain improves time resolution but at the cost of frequency resolution and vice-versa. With Morlet wavelet the trade-off between time and frequency resolution can be controlled by the choice of ω_0 [46]. We choose to run our analysis with a value of $\omega_0 = 6$, since it provides a good balance between the two resolutions and besides, this value gives the best ratio between Fourier Period and the scale parameter a during a Transform, equal to $\lambda = 1.03$ [47]. In this way the results in frequency domain are more interpretable since the scale parameters a used for the transform is almost equal to the Fourier period.

Two main types of analysis can be performed using wavelets: Continuous Wavelet Transform (CWT) and Discrete Wavelet Transform (DWT). The CWT is what we employed for our analysis.

Continuous wavelet transform decomposes a time series in time-frequency domain by successively convolving the timeseries with several scaled and translated version of the mother wavelet

[1] https://it.mathworks.com/help/wavelet/gs/introduction-to-the-wavelet-families.html

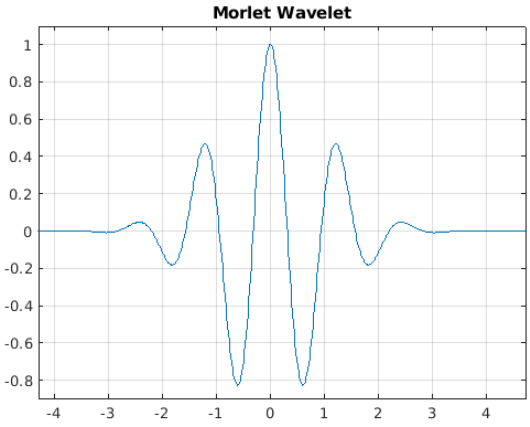


Figure 7.4: *Morlet wavelet*

ψ: $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$. If the timeseries is described by a function f, assumed to be real in the equation below, the convolution of f and $\psi_{a,b}(t)$ is

$$W_{a,b}^{f} = \int_{-\infty}^{+\infty} f(t) \cdot \psi_{a,b}^{*}(t)dt \tag{7.11}$$

Where the $*$ indicates the complex conjugate. This integral is performed for different values of a and b. It can be useful to visualize this integral in a dynamic way: set a value for a, a wavelet centered in b, is slided across the signal by changing the value of b and for each of these values, a coefficient is extracted by integrating the product between the wavelet and the signal; in this way, coefficients are function of frequency (or scale) and time. This operation is repeated for different values of a and b, and allows us to assemble a matrix called *scalogram*, which is basically a plot of these coefficients in time-frequency domain. A wavelet transform, can then be described as a mapping from $L^2(\Re) \rightarrow L^2(\Re^2)$.

To computationally implement CWT, a discretized version was implemented on MATLAB tools Wavelet toolbox. However, it should not be confused with the Discrete Wavelet Transform [2], a similar type of analysis but with some important differences.

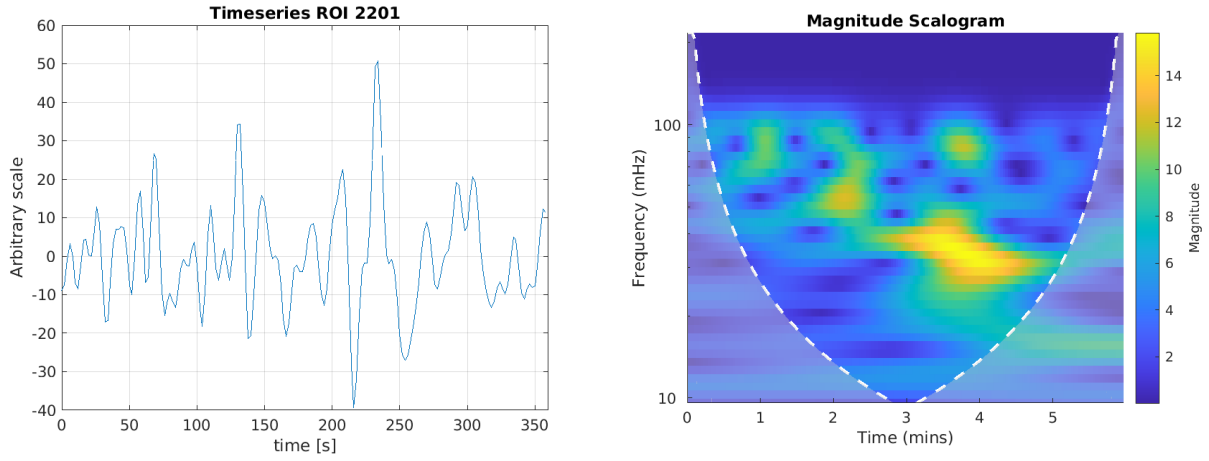
The difference between the continuous wavelet transform implemented on MATLAB and discrete wavelet transform lies in how finely stretching and shifting parameters are sampled: the CWT discretizes scale more finely than the discrete wavelet transform.

In the CWT, parameters are discretized based on a fractional power of two, by setting a = $2^{j/\nu}$ [48] [45] where ν, j are integers[3]. The parameter ν is often referred to as the number of *voices per octave* because increasing the scale by an octave (namely to double the frequency) requires ν intermediate steps. For example from $f = f_0 2^{\nu/\nu}$ *to* $f = f_0 2^{2\nu/\nu}$, ν steps are required. The larger the value of ν, the finer the discretization of the scale parameter. In the DWT, the scale parameter is always discretized to integer powers of 2, 2^j, j=1,2,3,..., so that the number of voices per octave is always 1. Since it employs a more rough discretization, DTW is usually used for denoising and compression of signals and images.

In our analysis we are going to use DTW implemented in MATLAB with the default parameters of 12 voices per octave.

[2] https://it.mathworks.com/help/wavelet/gs/continuous-and-discrete-wavelet-transforms.html

[3] The discretized wavelet can be written as $\frac{1}{2^{j/\nu}}\psi\left(\frac{t-b}{2^{j/\nu}}\right)$



(a) *(b)*

Figure 7.5: *Timeseries (fig a) of ROI 2201: right angular gyrus of patient 51056 from ABIDE I, and its corresponding Continuous Wavelet Transform (fig b). x axis corresponds to time points and y axis the frequencies derived from the scale parameter a of the wavelet convolving function. The COI is shown as a dotted white line and values outside the COI, are shown with a lighter faded. The color represents the magnitude of each wavelet coefficient.*

One problem that arises from working on a timeseries, which is basically a signal with a finite support, is that when a wavelet is located at the beginning or at the end of the signal, the wavelet extends itself outside the boundary of the signal, and the convolution would require nonexistent values beyond the boundary.

To overcome this problem it would be possible to accept this data information loss and truncate the values beyond boundaries; whereas an other approach could be to artificially extend data over the boundaries using different methods such as *zero padding* which assumes that the signal is zero outside its original support; *symmetrization* which extends the signal symmetrically outside the boundaries or *smooth padding* which recovers a signal by extrapolating values from the first derivative values or from the signal itself. Symmetrization method is the one employed for the subsequent analysis. The confidence value obtained on the boundaries, though, is lower than other obtained for central values of time location. Areas of the scalogram affected by these edge effects are indicated as outside the *Cone of influence* (COI) .

Figure 7.5 shows the timeseries extracted from the right angular gyrus of patient 51056 from ABIDE I, and its corresponding Continuous Wavelet Transform. In CWT scalogram, the COI is marked with a dotted line.

Wavelet coherence

Even though what we discussed so far only concerns the analysis of a single timeseries, with wavelet analysis is possible to compare time-frequency information of two signals by performing an analysis called *wavelet coherence*. From two timeseries it is possible to compute a transformation called Cross Wavelet Transform (XWT) which allows to examine the cross-wavelet power and relative phases. Using XWT is then possible to compute the wavelet coherence. From wavelet coherence we are finally able to extract a correlation coefficient, indicative of how much two signals

are correlated or anti-correlated.

Denoting as $W^X(a,b)$ the continuous wavelet transform of a signal X, the *wavelet power spectrum* of a single signal X(n) is defined as:

$$W^{XX}(a,b) = W^X(a,b)\left[W^X(a,b)\right]^* \tag{7.12}$$

where the * represents the conjugate transpose. Similarly, the **Cross Wavelet Transform (XWT)**, sometimes also called *cross wavelet spectrum*, of two time series X and Y is defined as:

$$W^{XY}(a,b) = W^X(a,b)[W^Y(a,b)]^* \tag{7.13}$$

whose module $|W^{XY}(a,b)|$ represents the amount of joint power between the two time series, and it is called *cross wavelet power*. From the cross wavelet spectrum (eq 7.13), is also possible to compute the the complex argument

$$\Delta\phi(a,b) = arg(W^{XY}(a,b)) = arctan\left(\frac{Im\left[W^{XY}(a,b)\right]}{Re\left[W^{XY}(a,b)\right]}\right) \tag{7.14}$$

which represents the relative phase between X and Y for each given value of the parameters a and b, and is defined over the interval $[-\pi,\pi]$.

The **wavelet coherence** $R^2(a,b)_{XY}$ for two timeseries X and Y, is defined as:

$$R^2(a,b)_{XY} = \frac{|S(W^{XY}(a,b))|^2}{S(|W^X(a,b)|^2)S(|W^Y(a,b)|^2)} \tag{7.15}$$

This coefficient ranges in the interval (0, 1) and represents the localized correlation coefficient between X and Y in the time-frequency domain.

In equation 7.15 S is a smoothing operator, both in frequency and time, defined as [49] [47]

$$S = S_{scale}S_{time}(W) \quad S_{time}(W) = W \cdot c_1^{\frac{-t^2}{2a^2}} \quad S_{scale}(W) = W \cdot c_2\Pi(0.6)$$

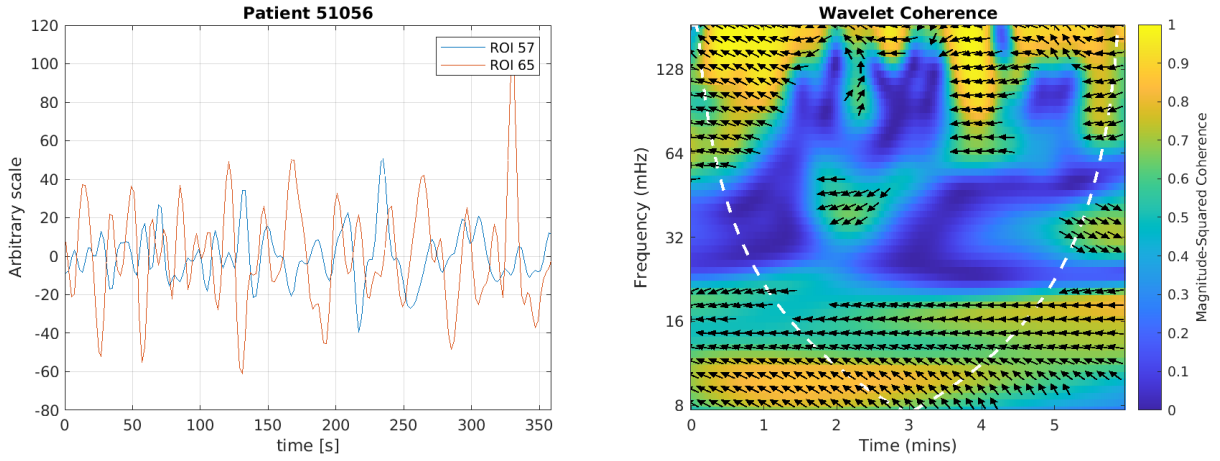
where c_1, c_2 are normalization constants, Π is a boxcar (rectangle) function and 0.6 is an empirically determined factor for Morlet Wavelet [50].

An example of wavelet coherence scalogram is shown in figure 7.6. It shows two timeseries (fig 7.6a) extracted from patient 51056, and the respective wavelet coherence scalogram (fig 7.6b). The color represents the magnitude of the cross-power, and phase information is represented as oriented arrows, pointing towards an imaginary 360 degrees circle, where the zero phase shift is represented by an arrow pointing right and a 180 degrees phase, by an arrow pointing left. Starting from 0 degrees arrows rotate counterclockwise.

Our goal is to extract from this wavelet coherence matrix, a single value, interpretable as a correlation coefficient, just like we had with Pearson coefficients. We can accomplish this, by extracting two types of information from this scalogram: the magnitude of the correlation and relative phase between the two signals.

We can extract the magnitude information of each entry of the scalogram, but before doing so, we need to assess the level of significance of this coherence matrix over the noise, this way we can estimate the statistical significance level of our values, and only collect the significant ones.

The theoretical procedure [47] [51] is to generate, using Monte Carlo methods, a large (1000) samples of wavelet coherence matrices using red noise timeseries. These red noise samples should be generated, for each time series, with the same 1-lag autoregressive coefficients (AR1 coefficient)



(a) *(b)*

Figure 7.6: *Plot of two timeseries from patient 51056 in figure a and their relative wavelet coherence scalogram (figure b). Timeseries are extracted from ROI 57 corresponding to right angular gyrus and ROI 65 right lateral occipital cortex. On the x axis the timepoints and on the y axis the frequencies of this time-frequency decomposition are reported respectively.*

of the two timeseries under examination. Then, for each pair of red noise timeseries, the wavelet coherence matrix should be computed.

However, since AR1 coefficients have little impact on the significance level [47], we choose to generate for each subject, a single sample of 1000 pairs of red noise. For each pair we obtained the wavelet coherence matrix using equation 7.15. All the matrices created in this way are stacked to obtain a $A \times B \times 1000$ (3D) noise matrix used to extract the distribution of the entries.

For each entry of a matrix $A \times B$ we obtain a distribution of 1000 coefficients given by the stack of all the noise matrices. This null distribution of wavelet coherence coefficient is used to estimate the threshold to 5% significance level for the subsequent analysis. We refer to the value corresponding to this $95-th$ percentile as a_{95}. Collecting the value of the $95-th$ percentile for the distribution of each entry, we obtain the noise matrix visualized in figure 7.7 that we can call $A95$ matrix.

Once assessed the threshold for the significance level, the second step is to extract the relative phase information of each entry of the wavelet coherence matrix of the two timeseries under analysis. At last, combining together information on significance level and on relative phase, we can estimate the time of in-phase (or out-of-phase) coherence, which can be seen as the percentage of time synchronicity (or anti-synchronicity) between the two timeseries [51]. This time of in-phase coefficient is defined as:

$$c_{XY} = \frac{100}{N} \sum_{a,b \in COI}^{N} I\left\{R^2_{XY}(a,b) > A_{95}(a,b)\right\} \cdot I\left\{-\frac{\pi}{4} < arg(W^{XY}(a,b)) < \frac{\pi}{4}\right\} \tag{7.16}$$

where the indices X and Y refer to the two timeseries X and Y; N is the total number of points inside the cone of influence (COI). $I\{...\}$ is either 0 or 1 depending on whether the condition inside is satisfied; a_{95} is the threshold value above which the computed wavelet coherence coefficient is regarded as significative.

The time of counter-phase coefficients have a similar definition. They can be obtained as:

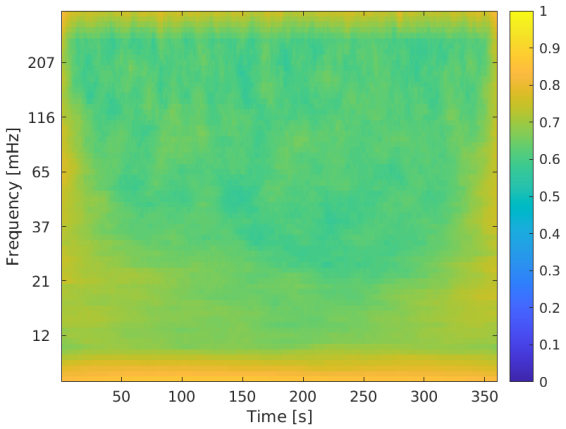


Figure 7.7: *Noise matrix obtained by collecting the 95th percentile of the distribution of each entry of the 1000 noise matrices stack. It determines the threshold to assess significance of the coefficients of the wavelet coherence matrix computed for two timeseries*

$$c_{XY} = \frac{100}{N} \sum_{a,b \in COI}^{N} I\left\{R^2_{XY}(a,b) > A_{95}(a,b)\right\} \cdot I\{arg(W^{XY}(a,b)) < -\frac{3\pi}{4} \ \vee \ arg(W^{XY}(a,b)) > \frac{3\pi}{4}\} \tag{7.17}$$

which is basically the same relation as 7.16 with a modification of the phase condition

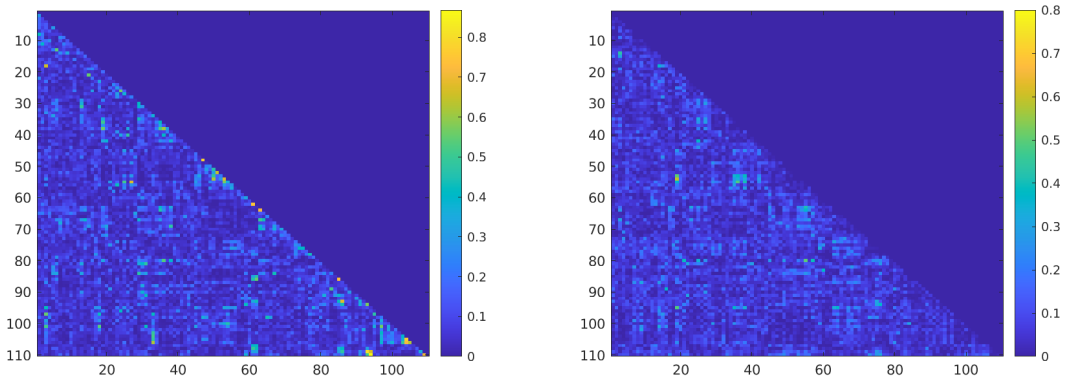
$$I\left\{-\frac{\pi}{4} < arg(W^{XY}(a,b)) < \frac{\pi}{4}\right\} \ \longrightarrow \ I\{arg(W^{XY}(a,b)) < -\frac{3\pi}{4} \ \vee \ arg(W^{XY}(a,b)) > \frac{3\pi}{4}\}$$

In short with this analysis, we are extracting coefficients just considering values from the wavelet coherence scalogram with a high significance level (above 95%) and with a small $\in [-\pi/4, \pi/4]$, or big ($\in [-\pi, -3\pi/4] \vee \in [3\pi/4, \pi]$) phase shift

Wavelet coherence maps were calculated using MATLAB's wavelet Toolbox, which employs the Morlet wavelet, and decomposes the frequency range using 12 subscales per octave and 9 octave.

From equations 7.16 and 7.17 both in-phase and out-of-phase coefficients matrices were computed.

An example of correlation matrix is created with in-phase and out-of-phase coefficients from data of patient 51056 is shown in figure 7.8.



(a) *Coefficients of In-phase percentage* ***(b)*** *Coefficients of out-of-phase percentage*

Figure 7.8: *Correlation matrix created with wavelet coefficients of in-phase and out-of-phase percentage extracted from patient 51056, ABIDE I*

Chapter 8

Multicenter data harmonization

Since now, a big percentage of neuroimaging studies have been carried out with limited data acquired from a single center, to minimize varibility in data due to the different instrumentations employed. In recent years, however, there is the tendency to create shared datasets, by pooling together data acquired from different centers. A positive side of creating a large dataset by putting together data from multiple centers is the possibility to work on dataset of bigger dimension, resulting in statistically more accurate analysis.

Yet, this process has some downsides since it introduces variability between data. Data acquired from different sources are affected by the so called *batch effects* or *scanner effects*, related to differences in the acquisition system such as the scanner model or the acquisition parameters. When analyzing data either with statistical analyses or with machine learning methods, a level out procedure becomes necessary. This procedure is generally referred as *harmonization*. When working with machine learning models, for example, harmonization can be helpful to avoid the model to learn patterns related to the acquisition site, and improve its ability to learn pattern related to the classification task it is conducting.

Harmonization was tested on different datasets and has proven to be an effective way to reduce scanner effects in different types of datasets such as genes microarray data, diffusion tensor imaging or structural MRI [52] [53] [54].

8.1 ComBat harmonization and NeuroHarmonize

The state-of-art procedure used in genomic is called ComBat (named after Combating Batch effects), used for structural MRI but applicable to any kind of imaging data, is used to mitigate scanner effects. It is based on a previous method initially proposed in 2007 [52], for gene expression studies to compute batch effect corrections, later implemented by Fortin et al [55] for the harmonization of cortical MRI volumes, and finally in its current improved version, developed by Pomponio et al. [56] in 2019 and available as a free Python package called Neuroharmonize [1]. In this paragraph we describe how ComBat technique works and after that, we discuss how it was modified and improved in the implementation made by Pomponio et al.

ComBat technique belongs to the family of Location and Scale adjustment methods, but it represents an improved version of them since it models site-specific scaling factors using empirical Bayes estimate to improve the estimation of the site-related parameters for small-sized datasets [55]. ComBat aims to reduce inter-site variance while preserving biological variability such as differences due to sex, age, FIQ (Full intellective quotient), ICV (Intra Cranical Volume) and so on. The model

[1] https://github.com/rpomponio/neuroHarmonize

assumes that a feature can be modeled as a linear combination of the biological variables plus the site effect, and the errors introduced with site effect can be modeled as both a multiplicative and an additive term. This error can be standardized by adjusting the mean and the variance across the batches.

To clarify how ComBat harmonization works, we can suppose to work with a dataset comprising different data, where each data consists on F features. This dataset is made of data acquired with K different scanners, where each scanner is used to obtain data from more than one subjects. We denote with y_{ijf} be the numeric value of the feature f for the patient i acquired with the scan (or equivalently, from the site) j so that $i = 1, 2, ..., K$ indexes the scanner, $j = 1, 2, ..., N_i$ indexes the subject acquired with scanner i, for a total of N_i subjects acquired for that scanner. f ranges from 1 to F being F the total number of features. As a premise, ComBat assumes that the value of each feature: y, can be written as depending on different parameters

$$y_{ijf} = \alpha_f + x_{ij}^T \beta_f + \gamma_{if} + \delta_{if}\epsilon_{ijf} \tag{8.1}$$

where:

- α_f is the mean value for the feature f,
- x_{ij} is the vector of the matrix X created with the covariates of interest such as age, sex,
- β_f is the vector of regression coefficients corresponding to X for the feature f,
- γ_{if} and δ_{if} represent the additive and multiplicative terms for site-i effects related to feature f respectively,
- ϵ_{ijf} is a residual term which is assumed to follow a normal distribution with zero mean and variance σ_f^2.

The final location-and-scale-adjusted data y_{ijf}^* are given by

$$y_{ijf}^* = \frac{y_{ijf} - \hat{\alpha}_f - x_{ij} \cdot \hat{\beta}_f - \hat{\gamma}_{if}}{\hat{\delta}_{if}} + \hat{\alpha}_f + x_{ij} \cdot \hat{\beta}_f \tag{8.2}$$

where $\hat{\alpha}_f, \hat{\beta}_f, \hat{\gamma}_{if}, \hat{\delta}_{if}$ are estimators of the corresponding parameters, based on the model.

The process that ComBat employs to estimate feature-dependent parameters, and to adjust data for batch effect can be summarized in 3 steps, at the end of which we obtain the results shown in equation 8.2

1. **Standardization**: Data are standardized feature-wise, so that every feature has similar overall mean and variance. least square regression, is then performed to determine parameters $\hat{\alpha}_f, \hat{\beta}_f, \hat{\gamma}_{jf}$ and subsequently, $\hat{\sigma}_f^2 = \frac{1}{n} \sum_{ij} \left(y_{ijf} - \hat{\alpha}_f - x_{ij}\hat{\beta}_f - \hat{\gamma}_{if} \right)^2$ being n the total number of patients.

   The standardized data are then calculated by equation 8.3

$$Z_{ijf} = \frac{y_{ijf} - \hat{\alpha}_f - x_{ij}\hat{\beta}_f}{\hat{\sigma}_f} \tag{8.3}$$

2. **Empirical Batch parameters estimate**: We assume that standardized data follow a normal distribution $Z_{ijf} \sim N(\gamma_{if}, \delta_{if}^2)$ and we seek for a proper estimation of parameters $\gamma_{if}, \delta_{if}^2$.

   One of the disadvantages of using a simple location and scale batch adjustment is that it requires a large batch size for the implementation because is not robust to outliers in small sample sizes. ComBat uses empirical Bayes estimates to provide a more robust adjustment for the parameters $\hat{\gamma}_{if}$ $\hat{\delta}_{if}^2$, making the model able to deal with small-sized dataset as well.

   It is also assumed that these site-effect parameters follow a normal distribution and an inverse gamma distribution respectively:

$$\gamma_{if} \sim N(\eta_i, \tau_i^2) \qquad \delta_{if}^2 \sim \text{Inverse Gamma}(\lambda_i, \theta_i) = \frac{\theta_i^{\lambda_i}}{\Gamma(\lambda_i)}(1/x)^{\lambda_i+1} \cdot e^{-\theta_i/x}$$

   And these hyperparameters $\eta_i, \tau_i^2, \lambda_i, \theta_i$ are empirically estimated from standardized data Z_{ijf} by using the method of moments. We then obtain improved estimation of parameters γ_{if}^* and δ_{if}^* and we use them for the third and last step, where we adjust our data using them.

3. **Adjust the data**: After the site-effect parameters have been calculated, we are finally able to adjust our initial data using the relation

$$y_{ijf}^{ComBat} = \frac{\hat{\sigma}_f}{\delta_{if}^*}(Z_{ijf} - \gamma_{if}^*) + \hat{\alpha}_f + x_{ij} \cdot \hat{\beta}_f \tag{8.4}$$

   which is just an equivalent way to write equation 8.2, using parameters estimated in the previous passage.

This is the main idea behind ComBat technique, but, as we said before, the state-of-art implementation of this technique by Pomponio et al. [56] is an improved version of it.

It differs in the modeling of the biological covariates, which in the formulas above are expressed by the terms $\hat{\alpha}_f + x_{ij} \cdot \hat{\beta}_f$ which represents a simple linear model. Pomponio substituted this linear model with a Generalized Additive Model (GAM). In this model covariates such as sex, age, FIQ, are represented by terms $x_{ij},\ z_{ij},\ w_{ij}$ which allow a better parametric modeling to deal with non-linear trends such as the trend of cortical thickness in relation to age. This way, the terms $\hat{\alpha}_f + X\hat{\beta}_f$ in equation 8.2 are substituted by a linear combination of function F of these covariates

$$\hat{\alpha}_f + x_{ij} \cdot \hat{\beta}_f \ \longrightarrow\ F_f(x_{ij}, z_{ij}, w_{if}...) = a_f + g_f(x_{ij}) + h_f(z_{ij}) + p_f(w_{ij}) + ... \tag{8.5}$$

Where functions g_f, h_f, p_f can be either linear or non-linear functions of our covariates, according to how we want to model these covariates. In the implementation of Neuroharmonize, when a covariate is set to be treated as non-linear, its function will be handled using thin plate regression splines to find the right basis expansion. This procedure of including non-linear terms though, brings with it the downside of a huge increasing in computational costs.

Chapter 9

Domain-adversarial Neural Networks

In this section we discuss a different and innovative approach to deal with multi-center datasets. So far we presented one important harmonization procedure that aims to remove site-related features in an analytical way; an other possible approach is to make use of a deep neural network specially designed to perform a classification unbiased by site-related features. The construction of this network gets its main idea from the network proposed by Ganin, Ustinova et al [57].

Their work addresses the issue of working with two different datasets. They call them *source* and *target* dataset, where each dataset contains data following different distributions. Data are in a total amount of N, of which n samples belong to the source dataset and $N - n$ samples to the target. Both source and target data belong to different classes and the task of their work is perform a classification over these classes. However, all the data belonging to the source dataset are associated to a label specifying the respective class, while data on the target dataset are unlabeled. Their goal is to train a network using both the source and the target dataset. The network must be able to learn distinctive patterns between classes, from data belonging to the source dataset. To this end they can use only data from the source dataset because it is the only one labeled. At the same time, the network is trained to learn distinctive characteristics between data from the source and the target domain. Combining these two information (of classes learned from the source domain and of source/domain differences), they manage to create a network able to learn important features for classification, as well as to generalize this information form one domain to another.

We illustrate the main aspects of this problem using a shallow neural network with a single hidden layer consisting on D nodes. We suppose that the network takes as input an m-dimensional features vector. The hidden layer can be represented as a function $G_f : \Re^m \to \Re^D$ with a weight parameters matrix $\mathbf{W}$ and bias vector b. For brevity's sake we can denote the parameters $\mathbf{W}$ and b together as θ_f. Given an input vector $x \in \Re^m$ the hidden layer acts like

$$G_f(x; \mathbf{W}, \mathbf{b}) = G_f(x; \theta_f) = f(\mathbf{W} \cdot x + b) \tag{9.1}$$

Where f is some activation function that we can identify as a sigmoid function. Similarly the output will be a function $G_y : R^D \to \Re^Y$ where Y is the total number of classes of our data. This layer can be written as

$$G_y(G_f(x; \theta_f); V, c) = f'(V \cdot G_f(x) + c) \tag{9.2}$$

Here, too we can denote parameters V and c with a single notation θ_y.

An usual train carried on only on the (labeled) source domain, will therefore bring to the minimization of the loss function associated with the output, dependent on parameters θ_f, θ_y

$$\min_{\theta_f, \theta_y} [\frac{1}{n} \sum_{i=1}^{n} L_y^i(Gy(G_f(x_i; \theta_f); \theta_y), y_i)] \tag{9.3}$$

Training procedure and minimization of this loss function can be performed only on the source dataset consisting on n samples, since target data are unlabeled. At this point, to tackle the problem of domain independence the idea introduced by Ganin et al. is to consider the hidden layer as an internal representation of data, and use its information to create a domain regressor.

A domain regressor can be implemented as a layer G_d, completely similar to the output layer G_y, depending on parameters U, d which we will denote as θ_d. It takes as input the output of the hidden layer G_f, and after being activated by a sigmoid function, returns an output. We indicate the loss function associated to this output as L_d, and the loss previously introducted for label classification as L_y.

The complete optimization function can now be written as

$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y^i - \lambda(\frac{1}{n}\sum_{i=1}^{n} L_d^i + \frac{1}{N-n}\sum_{i=n+1}^{N} L_d^i) \tag{9.4}$$

Following this strategy, the optimization of the function $E(\theta_f, \theta_y, \theta_d)$ involves a minimization with respect to parameters θ_f and θ_y, and a maximization with respect to θ_d. More precisely they seek for a saddle point given by

$$\begin{aligned} \hat{\theta}_f, \hat{\theta}_y &= \underset{\theta_f, \theta_y}{argmin} E(\theta_f, \theta_y, \theta_d) \\ \hat{\theta}_d &= \underset{\theta_d}{argmax} E(\theta_f, \theta_y, \theta_d) \end{aligned} \tag{9.5}$$

This task is accomplished by embedding the domain regressor G_d into the neural network consisting on G_f and G_y. The resulting neural network includes two branches: one for label classification and the other consisting of the domain regressor. They link the domain regressor to the layer G_f using a *gradient reversal layer* and exploit a classic stochastic gradient descent procedure to update weights. A gradient reversal layer allows to train the network in an adversarial way. It is placed at the top of the domain regressor branch, as we can see from figure 9.1 directly taken from their paper. Let us point out that so far, we discussed the structure of this network by using a shallow neural network consisting just on a single hidden layer, but this architecture can be generalised by adding additional layers for each branch of the network, as shown in figure 9.1.

During the training, of this network, the forward propagation is not influenced by this gradient reversal layer, however, during back-propagation, this layer acts by multiplying the gradient by -1 before passing it to the preceding layer. In this way, the partial derivatives of the domain loss L_d with respect to the parameters θ_f get a minus sign. The whole updating procedure for parameters $\theta_f, \theta_y, \theta_d$ can then be described by equations 9.6.

$$\begin{aligned} \theta_f &\to \theta_f - \frac{\partial L_y}{\partial \theta_f} + \lambda \frac{\partial L_d}{\partial \theta_f} \\ \theta_y &\to \theta_y - \frac{\partial L_y}{\partial \theta_y} \\ \theta_d &\to \theta_d - \lambda \frac{\partial L_d}{\partial \theta_d} \end{aligned} \tag{9.6}$$

During training, classification branch and domain regressor compete against each other in an adversarial way for the optimization of the parameters. For this reason this network is called Domain-Adversarial Neural Network (DANN).

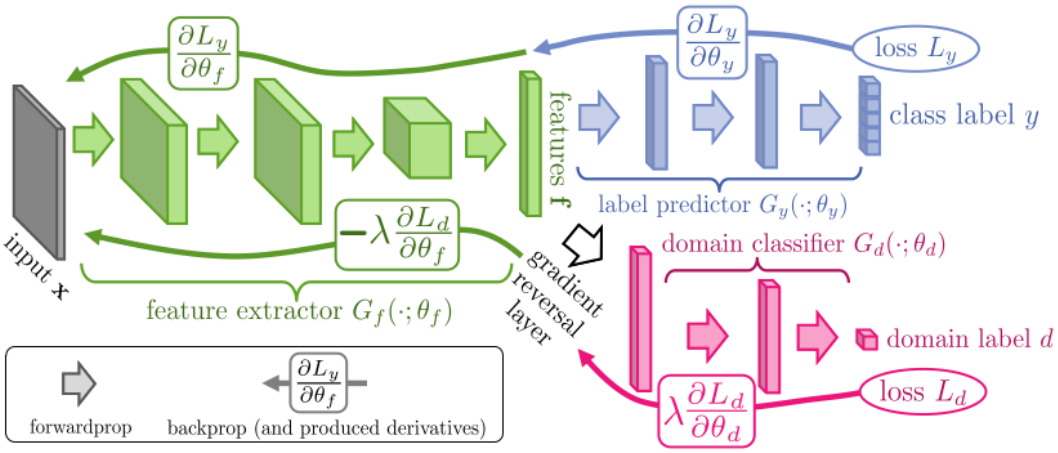


Figure 9.1: *A schematic representation of a Domain Adversarial Neural Network, from the presentation article [57]. The network mainly consists of three parts: a feature extractor, which takes inputs and creates an internal representation of data. From it the network is splitted into two branches: a label classificator G_y with associated loss L_y and a domain regressor/classifier denoted as G_d with associated loss L_d. This latter branch is linked to the feature extractor branch by making use of the gradient reversal layer.*

Chapter 10

Machine learning model explanation with SHAP

SHAP (SHapley Additive exPlanation) is a technique based on game theory that provides a method for machine learning model explanability. It is a powerful tool to get rid of the "black box" idea of a machine learning model and to get a clearer idea of the reasons why a model returns a certain output, after receiving an input sample.

As shown in fig. 10.1, which reports just a visual example took from the SHAP documentation [1], a common machine learning model acts like a black box that returns an output value after some non-linear unknown calculations over some input values. With an explanatory model, it is possible to quantify the contribution of each feature belonging to an input data and assess what are the most important features and how much they contributed to the final outcome.

Figure 10.1: *Representation of how a deep model works: it typically acts as a black box, taking some input data, such as age, sex, blood pressure, and useing them to compute an output whose value differs form a reference value (base rate). With SHAP it is possible to explain the contribution of each input and determine which one influenced the output the most. Credits: https://shap.readthedocs.io*

The idea behind SHAP is to use Shapley values (see chapter 4) to explain every feature contribution in a machine learning model, treating the learning process as a cooperative game. To apply the concept of Shapley values to a machine learning model we just adapt some terms we used for game theory: the payout of a coalition becomes the model prediction and the players are the features in an input data. The Shapley value of a single feature is defined as the average marginal

[1] https://shap.readthedocs.io/en/latest/

contribution of that feature across all possible coalitions.

Explainatory methods can be divided into two classes as local and global methods. In general, the objective of a local explainatory model can be expressed as the attempt to explain an output f(x) after a single instance x that can be identified as a vector of features. Local methods differ from global methods, because the latter provide a global explanation of the model, across all the instances. They are able to attribute an importance to each feature to determine which one contributes the most to the output of the model.

One of the key point of SHAP is its flexibility, being both a local and global explainatory model, this way we are able to explain a single instance as well as an entire set of instances and extract from it the most important features.

As a local model, SHAP shares with other algorithms some common traits. These algorithms typically employ a simplified function g to explain a model f. Given a single input vector $x \in \Re^F$ they introduce a simplified vector $x' \in \Re^{F'} = \{0,1\}^{F'}$. The simplified space F' is not necessarily equal to the input space F. A map function h_x, unique for each vector x, maps the simplified vector x' to the original vector x. x' is a binary vector consisting only on 0 and 1, if an entry is zero, it means that the corresponding features of x will be withheld from the subsequent analysis, while 1 means that it is included. Local methods seek to find the best approximation of g in the locality of x'. In order to do this, they create a perturbed dataset consisting on different vectors z' obtained by random sampling from the nonzero elements of x' [58]. Local methods try to obtain a model g that faithfully approximate f, and that $g(x') \approx f(x)$ if $x = h_x(x')$. This simplified model g' should also be a linear combination of the binary entries of x' [59]:

$$g(x') = \phi_0 + \sum_{i=1}^{F'} \phi_i x'_i \tag{10.1}$$

Several methods before SHAP were created that satisfy this condition such as LIME [58] or DeepLIFT [60] but they lacked additional properties that SHAP has:

1. **Local accuracy**:

$$g(x') = \phi_0 + \sum_{i=1}^{F'} \phi_i x'_i = f(x) \quad \text{when} \quad x = h_x(x') \tag{10.2}$$

   $g(x')$ is equal to $f(x)$, and not just an approximation $g(x') \approx f(x)$. Coefficients ϕ_i represent the impact of the associated feature to the model output, and ϕ_0 represents the coefficient corresponding to a vector x' with all entries set to zeros.

2. **Missingness**:

$$\text{if } x'_i = 0 \implies \text{ then } \phi_i = 0 \tag{10.3}$$

   If an entry of the simplified vector x' is set to zero, the corresponding feature of the x vector has no attributed impact on the model outcome.

3. **Consistency**: if we have two different models g'_1 and g'_2. Denoting with z'/i the setting where $z'_i = 0$

$$\text{if } g'_1(z') - g'_1(z'/i) \geq g'_2(z') - g'_2(z'/i) \forall z' \in \{0,1\}^F \implies \text{ then } \phi_i(g'_1, x) \geq \phi_i(g'_2, x) \tag{10.4}$$

   where $x = h_x(z')$. This property states that if the contribution of a feature z'_i increases, regardless all the other features in a model, its associated importance value increases or at least remain the same, but does not decrease.

In 1975 it was demonstrated [61] that the only coefficients satisfying all of the above properties are Shapley values, and, as a consequence, methods which employ different coefficients violate one of these properties, usually local accuracy and/or consistency. For this reason SHAP employs Shapley values and takes advantage of pre-existing algorithms (some of which were cited before: LIME and DeepLIFT) that used different coefficients. It readapts and enhances them providing an unified approach to assess feature importance for different models without any violation of the axioms above.

To compute the Shapley value for a feature we have to evaluate the model over all the possible subsets S that we can create with our data S$\subseteq F \setminus \{i\}$. This approach, in a problem with a high number of features would result in a huge computational work. To bypass this problem, we can choose not to calculate the exact Shapley value but just an approximation of it. Depending on the ML model we want to explain, different SHAP algorithms exist to optimize the preocess of computing Shapley value taking advantage of the knowledge of the ML model. Algorithms like TreeSHAP[2] are able to compute the exact Shapley value from tree and ensambles of trees. Others like KernelSHAP[3] and DeepSHAP[4] compute just an approximation of the Shapley values. Furthermore, while KernelSHAP is a model-agnostic explainatory algorithm, DeepSHAP is optimized to perform rapidly on deep neural networks.

SHAP is implemented as a free Python package with MIT license, developed and maintained by Scott Lundberg [5].

10.1 DeepLIFT and DeepSHAP

DeepSHAP is a model-specific explainatory algorithm specifically designed to perform on deep neural networks making use of previous knowledge of their structure. It can be considered as an enhanced version of the DeepLIFT algorithm, thanks to the introduction of the Shapley values, rather than the coefficients employed in DeepLIFT called *DeepLIFT multipliers* [60].

The strategy implemented by DeepLIFT and DeepSHAP to explain features is based on the calculation of the difference between the output computed with the true sample we want to explain and the output computed with some reference input data. For DeepLIFT the choice of the reference inputs depends on the type of the data, it can differ for example between images and genomic data. For DeepSHAP this reference input is computed by averaging over a subset of the train dataset, which is named *background* dataset.

To evaluate the importance of a feature, DeepLIFT employs coefficients called DeepLIFT multipliers and compute them during the back-propagation procedure assigning a multiplier to each neuron of the network. The process can be summarized as follow:

Denoting as t the output of an inner neuron and as $x_1, x_2, ..., x_n$ the output of some preceding neurons necessary to compute t, if we label as t_0 the reference output, we can compute the value $\Delta t = t - t_0$. Denoting as Δx_i the difference between the value of the feature and the reference value for that feature, DeepLIFT computes a contribution score $C_{\Delta x_i \Delta t}$ associated to the feature x_i. A contribution score can be defined as the amount of the difference from reference in t (Δt) that is attributed to the difference from reference of x_i (Δx_i). Using contribution scores, a DeepLIFT multiplier is simply defined as:

[2]https://shap.readthedocs.io/en/latest/generated/shap.explainers.Tree.html
[3]https://shap-lrjball.readthedocs.io/en/latest/generated/shap.KernelExplainer.html
[4]https://shap-lrjball.readthedocs.io/en/latest/generated/shap.DeepExplainer.html
[5]https://github.com/slundberg/shap

$$m_{\Delta x_i \Delta t} = \frac{C_{\Delta x_i \Delta t}}{\Delta x_i} \ .$$

Once computed the multipliers of the last neuron of the network, it is possible to compute the multiplier of any preceding neurons during the back-propagation. For a proper discussion of how contribution scores and multipliers are computed in DeepLIFT, we refer to [60].

DeepSHAP exploits the same principles as DeepLIFT, and uses Shapley values for smaller components of a network to compute values for the whole network. It accomplish this by recursively passing an equivalent value of the DeepLIFT multipliers, now defined in terms of Shapley values, during the back-propagation. To simplify this discussion, we explain the main idea of how this process works by making use of a simplified neural network.

We suppose to have a single input vector x consisting on two features only: $x = (x_1, x_2)$ and a single background vector $b = (b_{x_1}, b_{x_2})$.

We denote as f_{x_i} the actual value of the feature x_i and as b_{x_i} the value of the entry from the backgound vector corresponding to x_i. We also indicate as h_i the output of a neuron after the input x, and with b_{h_i} the output of the neuron when the input is b. Giving a look at figure 10.2 we can write formulas for forward propagation.

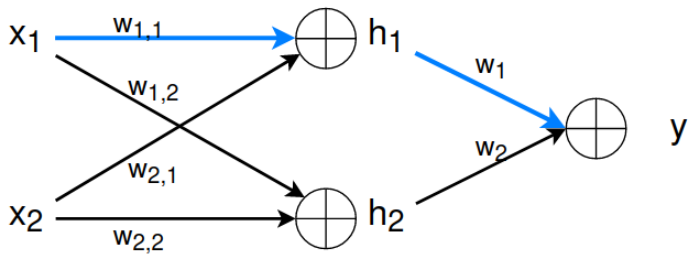


Figure 10.2: *Representation of a linear network and important quantities involved during forward propagation.*

$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 \qquad y = w_1 h_1 + w_2 h_2$$

The Shapley values for each node are computed by summing all the contributions from each path, where the path contribution to a Shapley value for x_i is defined as the product of the weights along the path and the difference between f_{x_i} and the background value b_{x_i}. In practice, if we aim to compute the contribution to the Shapley value of h_1 (see figure 10.2) coming from the blue path, we have:

$$\phi(h_1)^{\text{blue path}} = w_1(f_{h_1} - b_{h_1})$$

with this information, we can compute the contribution to the Shapley value of x_1 coming from the blue path

$$\phi(x_1)^{\text{blue path}} = (f_{x_1} - b_{x_1})w_{1,1}w_1 = (f_{x_1} - b_{x_1})w_{1,1}\frac{\phi(h_1)}{f_{h_1} - b_{h_1}} \ . \tag{10.5}$$

Adding all the contributions from all the possible paths linked to x_1, we are able to determine the Shapley value associated to this feature [62].

In figure 10.2 only two paths for x_1 are represented, however, a real neural network consists of much more input features and consequently many more patterns will contribute. Considering all

the possible patterns, it is possible to compute the final Shapley value for a feature x_i in terms of the attributions of all the intermediary nodes of the network.

10.2 Shapley values as feature importance

To express the absolute significance of a feature, we can rely on its Shapley values since features with high absolute Shapley can be considered as more influent during the computation of the output for a ML model.

The absolute importance value for a feature f is calculated as the mean of the magnitude of all the Shapley value for that feature, so the mean is performed across all the n samples we aim to explain. Thus, the absolute importance value can be expressed as:

$$I_f = \frac{1}{n} \sum_{i=1}^{n} \|\phi_i(f)\| \ . \tag{10.6}$$

IMPLEMENTATION & RESULTS

Chapter 11

Analysis workflow

Up to this point, we have been discussing what type of data we used in our work and how we preprocessed them. In this chapter we specify what kind of analysis we carried out using these data.

We mainly performed classification of control subjects vs ASD patients using a deep neural network. We compared different harmonization pipelines implemented during the classification process.

To this end, before running any classification procedure, in chapter 12 we show the evaluation of the effect of the harmonization process explained in chapter 8. We report these results to allow a visual feedback and understand how harmonization modify data and how effective is this procedure in eliminating site-related information from our data, which is a confounding information for a ML classifier.

In chapter 13 we present the architectures of the deep neural networks employed for classification and how we reached these architectures.

In chapter 14 we present the controls vs ASD classification results using deep neural networks. We followed different classification and harmonization pipelines and we compared the performances between a DNN and a Random Forest classifier. We seek to find the best strategy to obtain the best separability between data from controls and ASD.

Finally, in chapter 15 we applied techniques of explainable AI to extract information about what features are relevant in the discrimination between controls and ASD.

We start now examining in details what connectivity measures we employed and the classification and harmonization pipelines we carried out.

11.1 Choice of the functional connectivity measure

All the different classification and harmonization pipelines explained in this chapter are carried out using different measures of brain FC, created following the analysis explained in chapter 7, or obtained through a combination of them. In details, we used the following measures as input to DNN and ML classifiers.

- **Pearson** correlation coefficients, Fisher transformed according to equation 7.4;
- Four different coefficients computed using wavelet analysis:
  - **w_in**: in-phase wavelet coefficients, computed using equation 7.16;

- **w_out**: counter-phase wavelet coefficients, computed using equation 7.17;
- **w_stack** = w_in $\oplus$ w_out: in-phase and counter-phase coefficient stacked together in a single array of dimension 11990, namely twice as long as the array of w_in or w_out singularly taken (with length equal to 5995);
- **w_diff** = w_in - w_out: coefficients resulting from the elementwise subtraction of w_in and w_out. This is an attempt to create a single coefficient with similar properties as Pearson correlation coefficients. w_diff have range [-1, +1] where w_diff = -1 is obtained when the two signals are completely anti-correlated which means w_in = 0 and w_out = +1 and w_diff = +1 is obtained with perfect in-phase correlation namely w_in = 1 and w_out = 0.

11.2 Classification and harmonization pipelines

To assess the performances of a machine learning model we implemented a k-fold cross validation scheme in order to train and test the model using each time a different train and test dataset. We evaluated and reported model performances as the mean of the results and the standard deviation of the AUC scores for each fold. Since we are not working with a huge amount of data (< 1600 samples), we performed a 5-fold CV. With a 10 k-fold CV, the reduced number of data in each test set would have lead to a higher variability of the results. For this reason we preferred to hold a greater number of data in test, at the cost of reducing the number of data available for train. This choice reduces the variability among them and consequently reduces the error of the model score. K-fold CV was implemented using the stratifiedKFold class provided by scikit-learn library for Python[1].

A flowchart of the different harmonization pipelines used to perform classification with a machine learning or deep learning model is shown in figure 11.1. Each block indicates a process or a result.

1. Selection of FC measure: select whether to use Pearson-based correlation coefficients, or wavelet-based measures among the ones listed in section 11.1;

2. Dataset selection: select the desired attributes of patients in order to reduce the dataset to a more homogeneous cohort. For example we could choose to use the whole ABIDE I + II dataset or just ABIDE I, to limit analysis to a certain age range, sex, or to limit the eye status at scan to open or closed eyes. In this regard, our choices are explained in section 11.3;

3. The classification of controls vs ASD can be fulfilled by either a deep neural network, a Random Forest classifier, or the domain-adversarial neural network;

4. Based on how we implemented the harmonization during the classification procedure, we can distinguish four pipelines: 1) harmonization of data implemented inside the k-fold CV, 2) harmonization implemented before the k-fold CV, 3) without implementing harmonization 4) harmonization included within the DNN structure and learning process by using the domain-adversarial neural network.

5. For each pipeline, the final classification score is reported as the mean and standard deviation of the AUC across all the partitions created by the k-fold CV.

In details, the four harmonization and classification pipelines we followed are:

[1] https://scikit-learn.org/stable/

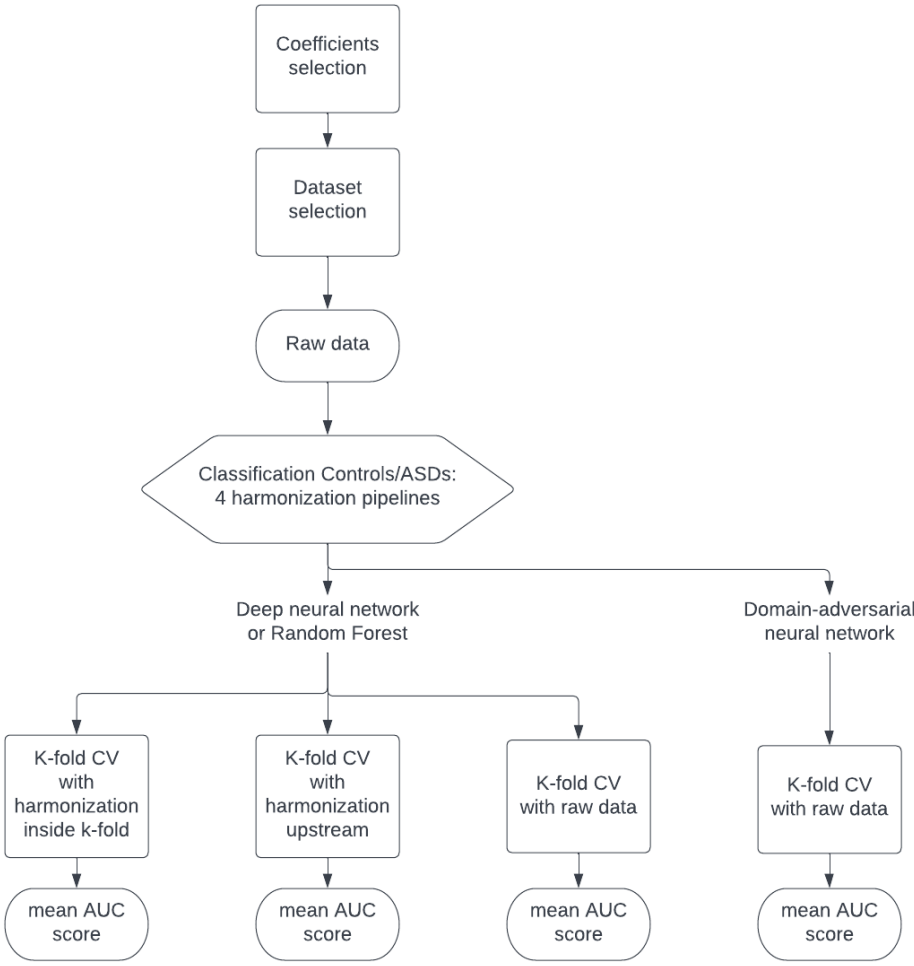


Figure 11.1: *Flowchart of the harmonization and classification pipelines carried out in this work.*

1. Classification of control subjects vs ASD patients with harmonization of data implemented inside the k-fold CV procedure, following the flowchart shown in figure 11.2. For each fold, data are splitted into a train and a test subset. The harmonization model is created on control data of the train dataset, and applied to the ASD data of train and to the entire test dataset. Harmonized data of control and ASD belonging to train, are merged again to obtain a harmonized train dataset. Classification is performed on the new harmonized dataset created in this way. The training of the model is accomplished on the harmonized train dataset and then the model is evaluated on the harmonized test dataset. This entire process is repeated for each fold with different train/test partitions;

2. Classification controls vs ASD using dataset harmonized before the k-fold CV procedure, created following the pipeline shown in figure 11.3. We refer to this procedure as upstream harmonization. From the entire dataset controls data are collected, and the harmonization model is created using their information. The harmonization model is then applied to the all the controls and the ASD data, and the harmonized dataset is created. Using this dataset, a k-fold CV is performed: the dataset is split into train and test subsets and on them, classification is carried through;

3. Classification controls vs ASD using raw (non-harmonized) data;

4. Classification controls vs ASD using the domain adversarial neural network, giving as input raw data. Using this network, an embedded harmonization is implemented through an adversarial learning process.

We believe that implementing the harmonization procedure inside a k-fold CV process, is the best way to keep train and test data apart and avoid data leakage between the two subsets. As a general definition, we have data leakage when information outside the training set is used to create and train models. This additional information allows the model to know something more about the data that otherwise it would have not known. This leads to an improvement of the results and an overestimation of its performances. When we harmonize the entire dataset upstream, namely before the splitting into a train and test, the harmonization model executes a fit using all the data, so each data is modified according to information obtained from every other data. In this way, once the harmonization is done, within the data belonging to the train set there is already information about the data belonging to the test. This would likely lead to a misleading increase of model performances.

Furthermore, some published papers where harmonization of data is performed, evaluated model performances using a leave-one-site-out Cross Validation [12] [63]. In our case this procedure is not feasible, if we want to perform harmonization inside the CV procedure. To compute the harmonization model, we need information about control data form each site. If a site is only on the test dataset, we would not have information to harmonize its data since we are using a model created with only information obtained from the train dataset.

11.3 Selection of sample subsets

With reference to the "Dataset selection" in figure 11.1, we discuss now what selection criteria were adopted for the cohorts of subjects.

First of all, we excluded from the analysis data from sites NYU_2 and KUL_3 because as noticed in chapter 5, they did not provide control subjects but only ASD patients. Without control data, we are not able to perform the harmonization procedure on these sites.

We choose to run analysis considering only male subjects, aged between 5 and 40 years. This allows to reduce variability due to sex. The limitation in the age range has been comprised as very few subjects over 40 years are included in the dataset.

The first collection of data we used to run classification on, is the whole dataset consisting of ABIDE I and II. With the cuts on covariates cited above, we obtained a dataset consisting on 1470 subjects. This dataset can be divided to analyze classification performances separately on ABIDE I and ABIDE II, maintaining the same constraints on ages and sex.

This choice was made because some works dealing with controls vs ASD classification on the ABIDE dataset, are carried out only on ABIDE I, or, more precisely, on ABIDE I preprocessed (see section 6.2) dataset [12]. Thus, for an immediate comparison we chose to run classification on the two disjoint datasets as well.

To seek for a more homogeneous dataset, we can choose to set constrains on the eye status at scan. The objective is to select only patients who kept open eyes throughout the entire scan session. This, as mentioned in chapter 1.3 helps removing data potentially altered by sleep. This constraint was applied on the dataset consisting on ABIDE I + II and separately on ABIDE I and ABIDE II. In short, considering these constraints, we obtain 6 different datasets, with different number of patients each, summarized in table 11.1.

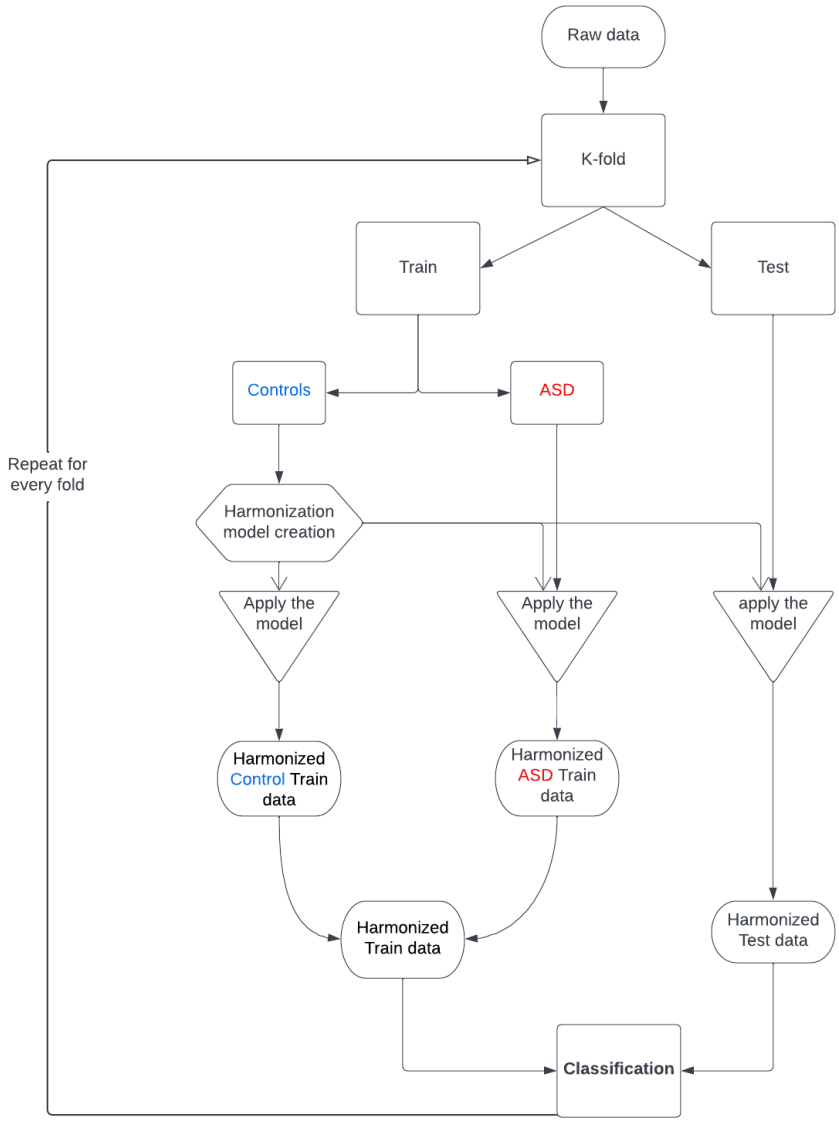


Figure 11.2: *Flowchart for the implementation of harmonization inside a k-fold CV:*

Dataset	Constraints	Tot	Controls	ASD
AB I+II	eye = all, sex = M, age = (5, 40)	1470	737	733
AB I	eye = all, sex = M, age = (5, 40)	841	426	415
AB II	eye = all, sex = M, age = (5, 40)	629	311	318
AB I + II	eye = open, sex = M, age = (5, 40)	1026	514	512
AB I	eye = open, sex = M, age = (5, 40)	568	281	287
AB II	eye = open, sex = M, age = (5, 40)	458	233	225

Table 11.1: *Total number of data and control/ASD amount for each subset defined according to the specified constraints.*

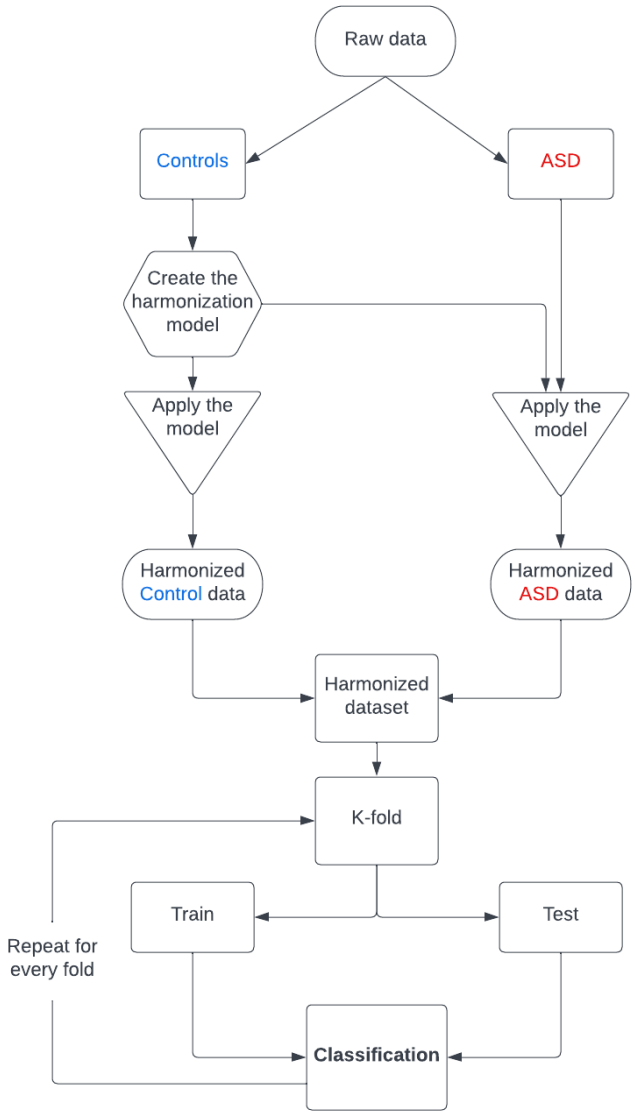


Figure 11.3: *Flowchart of the upstream implementation of harmonization, before the k-fold split*

11.4 Data dimensionality reduction

An important issue related to data we are working with concerns their dimensionality. We are dealing with data lying in a high dimensional space but we lack an apropriate number of data to make acccurate predictions on them: this problem is usually referred as the "curse of dimensionality". In our work, dimensionality is a relevant issue since we are dealing with 5995 features and less than 1500 patients and, for this reason, we run our analysis in a condition of permantent overfitting. One important method to tackle this aspect is to perform a Principal Components Analysis (PCA), as explained in section 3.3, hence reducing the dimensionality of the problem.

When applying PCA to a dataset we seek to explain as much variance as possible, without losing relevant information. For this reason we start our PCA analysis extracting a number of principal components that explain more than 90% of variance, then we gradually reduce this number by

halving it.

As mentioned in section 3.3, the maximum number of principal components that is possible to extract is limited by the number of samples in the dataset and by the number of features of each sample. In our case the number of samples $n\_samples$ is always lower than the number of features $n\_features$, since we have less than 1500 samples and 5995 features for each sample. Specifically, the number of principal components will be limited by the number of data in the train dataset. We use the train dataset only to calculate principal components because this is the correct way to proceed to avoid data leakage between train and test dataset. The search for principal components is then accomplished only on the train dataset and subsequently the same transformation is applied to the test dataset. In accordance to what we made for harmonization, we calculated the principal components just on the training set, thus keeping train and test set apart. For this reason we had to implement it inside the k-fold CV procedure.

To avoid repeating all the analyses mentioned earlier with the additional implementation of PCA, we chose to implement it only with in the pipeline that gave the best classification results and using just two dataset: ABIDE I + II and ABIDE I with patients with open eyes. From the initial input data consisting of 5995 features, we performed classification using different values of principal components. The results are shown in section 14.3.

Chapter 12

Effect of the Harmonization on data

As described in chapter 8, multicenter dataset need to be harmonized in order to attenuate the confounding site effect. ComBat-based harmonization procedure simultaneously models and estimates biological and non biological terms, from data and algebraically removes the estimated additive and multiplicative site-effect terms.

The harmonization procedure was tested on the dataset created by using patients from ABIDE I + ABIDE II once the restrictions on sex and age discussed in section 11 were applied. From these patients we tested the effect of the harmonization on Pearson-based FC measures and on wavelet-based ones (in particular, on the in phase time percentage w_in). We present a visual representation of how features are modified after being processed with a harmonization pipeline. Then, we run a more ML-based analysis to have a more quantitative evaluation of the harmonization effect.

To harmonize data we used the NeuroHarmonize package[1] for Python, developed by Pomponio et al. [56] which implemented and added some features to the previous NeuroComBat Python package[2].

To use NeuroHarmonize package we need to input the feature data to harmonize and the information about sites as well as information about all the covariates we want to preserve the effect of. We could also specify whether or not to include non-linear terms for some covariates.

For our data, biological information were provided by a file made available on the ABIDE website. In this file, for each patient, medical and biological data are collected in order to provide as much information as possible for each patient.

We tested the harmonization procedure on 1470 male subjects coming from ABIDE I and ABIDE II of which 737 are controls and 733 patients, choosing as covariates to be preserved the age, and the FIQ (Full Intelligence Quotient) to maintain important possible biolgical trends in the data and avoid overcorrections.

The central idea is to create the model as explained on chapter 8 on control subjects only, to operate without the influence of information related to ASD patients whose feature may follow a different distribution compared to controls. Once created the model on control subjects, it is applied to both control and ASD subjects. Following this procedure site-related information are eliminated from our data.

[1] https://github.com/rpomponio/neuroHarmonize

[2] https://github.com/Jfortin1/ComBatHarmonization

12.1 Visual assessment of the Harmonization effect on data

To have a visual representation of how much the FC measures are modified by the harmonization protocol across different sites, in fig. 12.1 two features among the 5995 are shown as a function of the sites: *feature 324* and *feature 2800*, randomly drawn from the 5995 possible correlation coefficients. Feature 324 represents the FC between right and left inferior frontal gyrus, and feature 2800 the FC between left precuneous cortex and the left inferior frontal gyrus. This figure shows features from Pearson-based FC while in figure 12.2 are shown the same two features obtained with the wavelet-based FC measure (w_in). The FC values are plotted as a box along the y axis and sites are indicated along the x axis. It appears clear how the median value of each feature for each site is shifted and features are stretched or shrinked to make them follow a more smooth distribution.

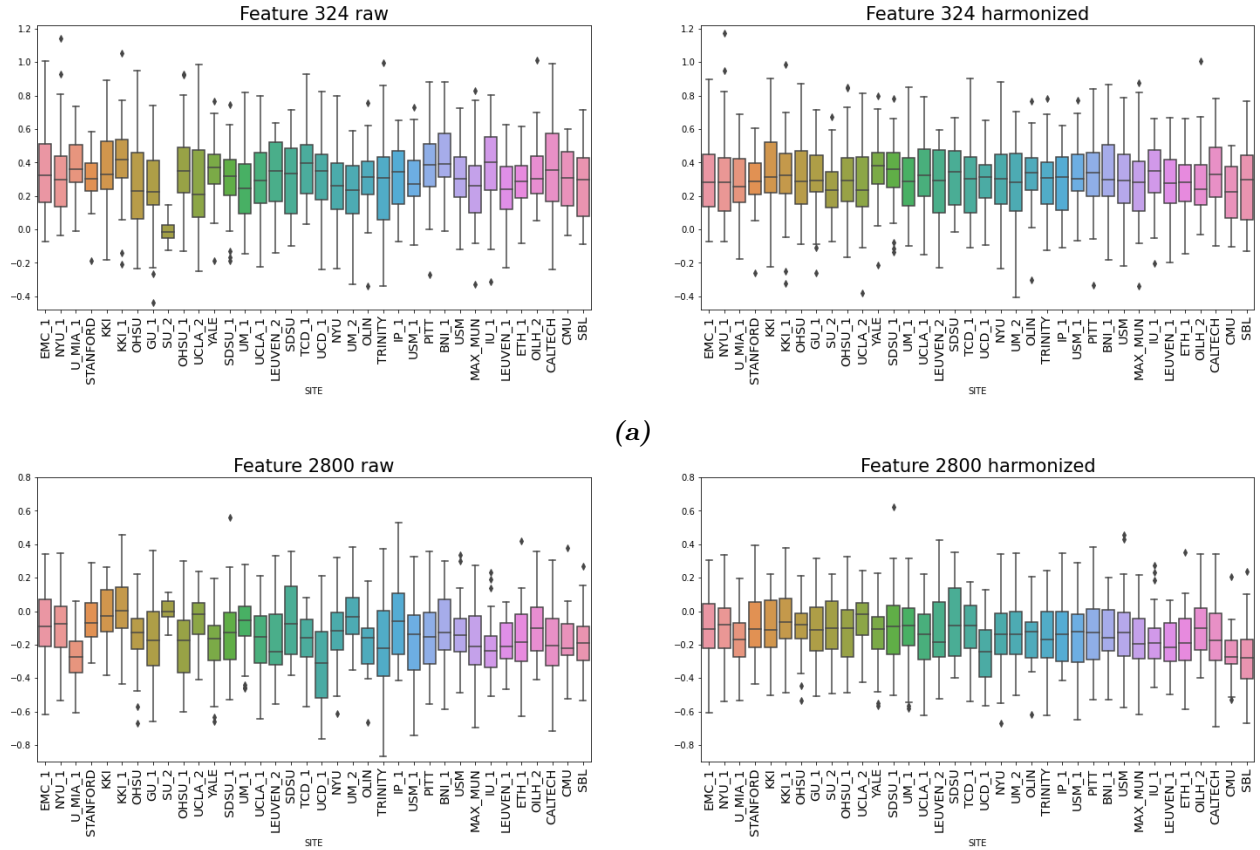
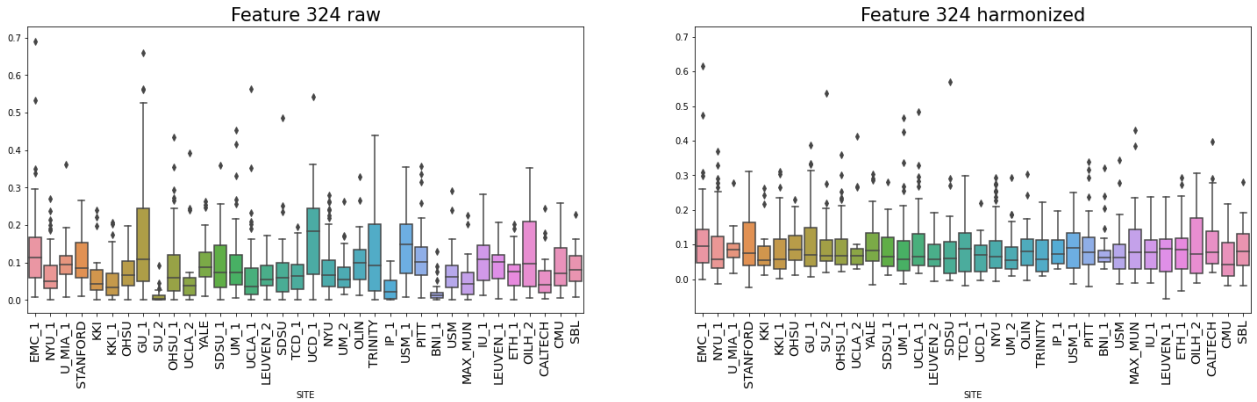
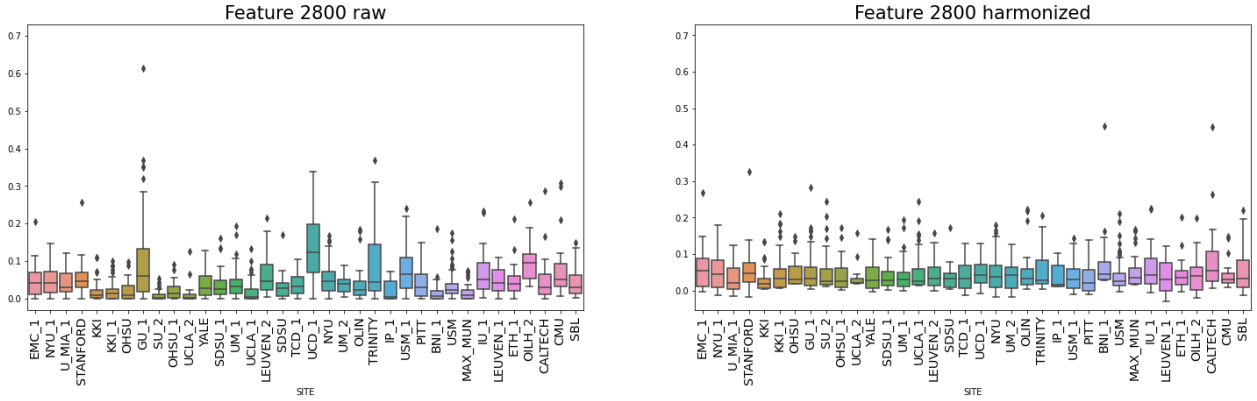


Figure 12.1: *Pearson-based connectivity measures per site for features 324 (fig 12.1a) and 2800 (fig 12.1b) before and after harmonization*

If we split these plots into control and ASD subjects to see the effect of harmonization separately on these subsets, we obtain fig. 12.3 which shows feature 324 computed using Pearson coefficients and figure 12.3b with wavelet-based FC.



(a)



(b)

Figure 12.2: *Wavelet-based connectivity measures (w_in), for features 324 (fig 12.2a) and 2800 (fig 12.2b) before and after harmonization*

12.2 Quantification of the effect of data harmonization

To test the effectiveness of this harmonization procedure, we performed a ML-based binary classification of data: site vs. site, for each pair of sites. The classification is carried out using a Random Forest classifier and we tested its performances for site classification before and after data harmonization. In this analysis, we splitted the model into a train and a test subsets: using only control subjects from the train dataset, we created the harmonization model as explained before.

To actually harmonize data, the model is applied to all the data: controls and ASD of the train dataset and controls and ASD of the test set. Thus, two Random Forest classifiers have been trained: one using raw data and the second one using harmonized data.

As mentioned in the previous paragraph, two sites (`NYU_2` and `KUL_3`) provided only ASD patients and, since we can not apply the NeuroHarmonize harmonization procedure on them, we excluded these sites from our analysis.

Figures 12.4 and 12.5 show the results of the site vs site classification using two different measures of functional connectivity: the Pearson correlations and wavelet of in-phase percentage coefficients. Sites along the x and y axes are ordered by increasing average age of patients and the performance are reported in terms of AUC scores.

Specifically, fig. 12.4a and 12.5a show the AUC scores of the model trained on the raw, non-harmonized dataset, while fig. 12.4b and 12.5b the results obtained on harmonized data.

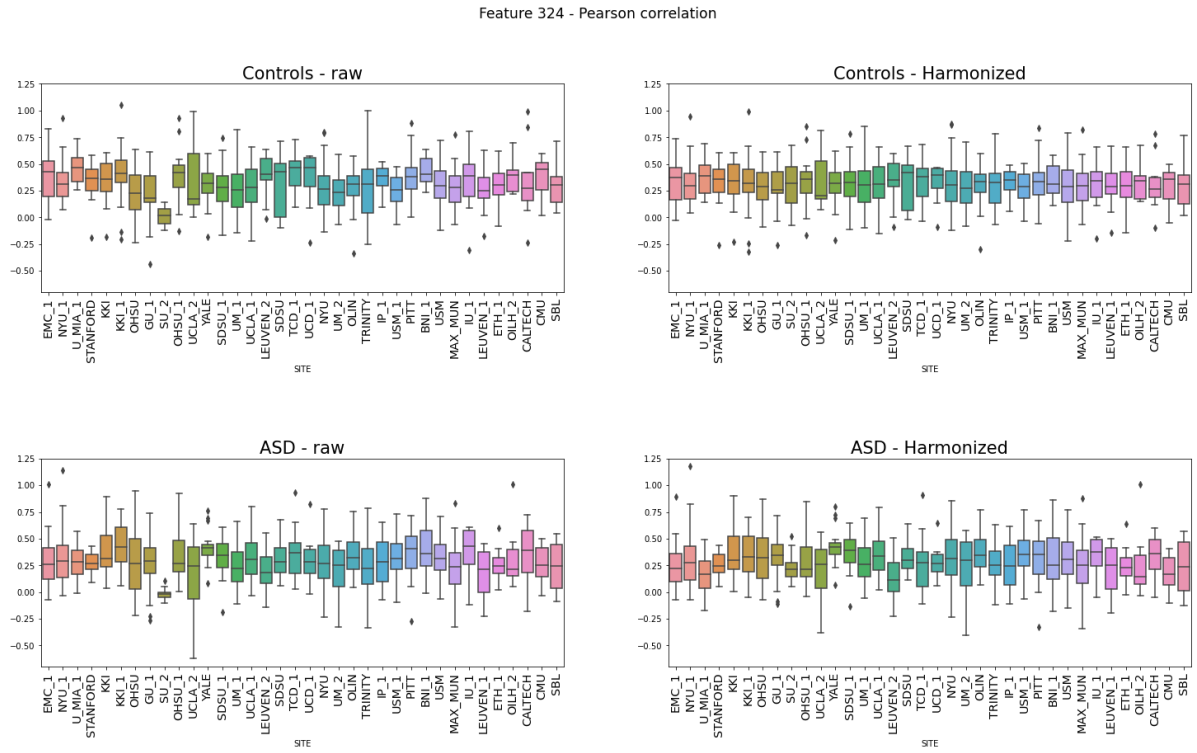
12.3 Discussion on harmonization

Figure 12.4 and 12.5 show a binary classification performed with a Random Forest, for site classification of Pearson-based and wavelet-based correlation coefficients respectively. Before the harmonization of data we notice (see fig. 12.4a and 12.5a) that the AUC values are mainly close to 1, which stands for exact ability to distinguish site_a from site_b. With Pearson-based FC this value decreases when the classification is performed using harmonized data (see fig. 12.4b). This stands as a confirmation that the harmonization procedure removes (or, at least, reduces) site-related effects on features making them less recognizable among data. This same effectiveness, though, is not visible with wavelet-based FC measures. It appears clear from fig. 12.5b) that sites still remain distinguishable even after harmonization. Actually, they even becomes more distinguishable. To understand why this happens we can take a look at fig. 12.2 and compare it with fig. 12.1 we can notice the presence of a high number of outliers in data. It is likely that those values are decisive and for the discrimination between two sites. The discriminability of sites becomes higher because after harmonization, for each site, a feature is scaled to follow a more regular distribution centered at its mean value, however, the outliers still remain outliers even after the harmonization, as it is possible to see from the plots. This leads to a greater effect of these outliers to the outcome of the classification. In addition, an other factor that facilitates the discrimination of different sites after the harmonization, is the effect that harmonization has on ASD data. As it is possible to notice from fig. 12.3a and 12.3b that both for Pearson and for wavelet FC measures, ASD data keep a site-related trend. The effect of both this trend and the presence of outliers is what makes sites recognizable even after the harmonization.

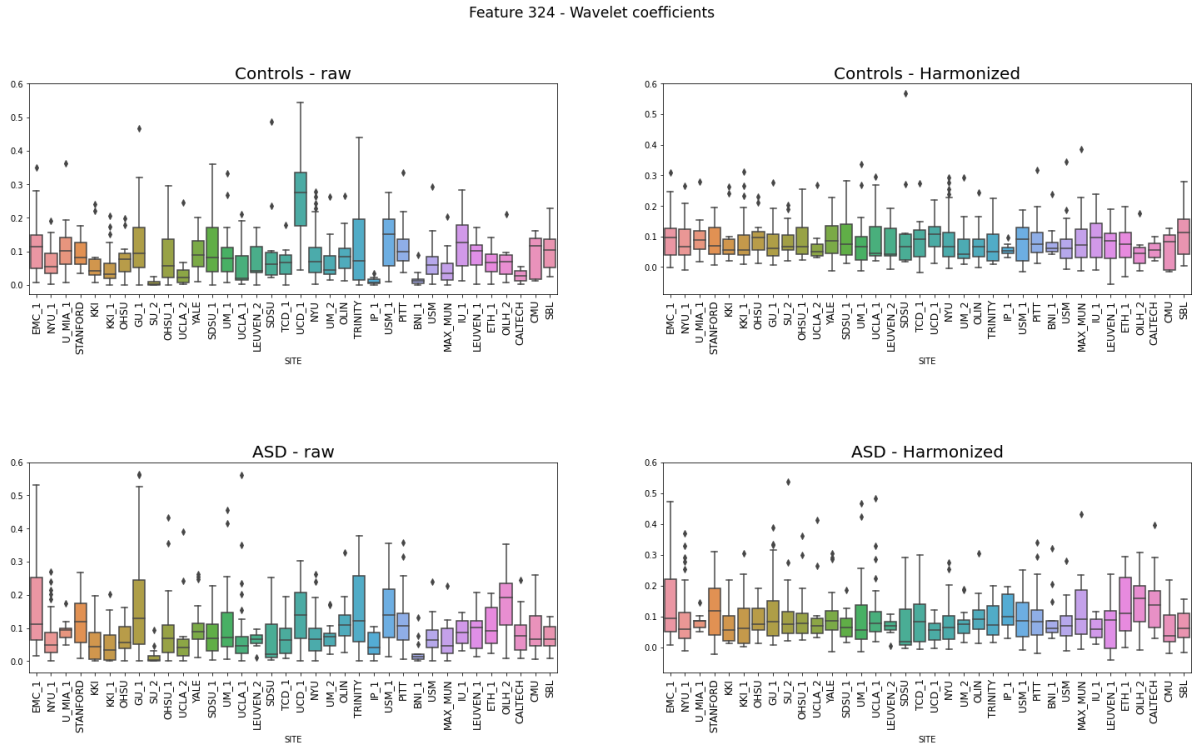
Both for Pearson-based FC in fig. 12.1 and for wavelet-based FC in fig. 12.2 we can notice a site-related trend of features of raw data (plots on the left side). This trend is mitigated by the effect of harmonization, on the right side of fig. 12.1 and 12.2. However, harmonization acts differently on controls and ASD data.

In figure 12.3a and 12.3b it is possible to notice how harmonization affects controls and ASD data separately. It appears that it is more effective in removing site-effects from controls than from ASD. This result is a consequence of the different distribution followed by controls and ASD, so that, parameters for harmonization extracted from control distribution may differ from the necessary ones to properly harmonize ASD data. The main reason is likely due to the low statistics in both cases. In a dataset of larger dimensions, the theoretical trend of harmonized ASD data should be as smoothed as the one of harmonized control data.

We can therefore conclude that in our data there is a bias linked to the acquisition site. After harmonization procedure both Pearson-based FC measures and wavelet-based ones assume a more regular trend. This is a visual confirmation that harmonization effectively removes, or at least reduces, site-related effects.

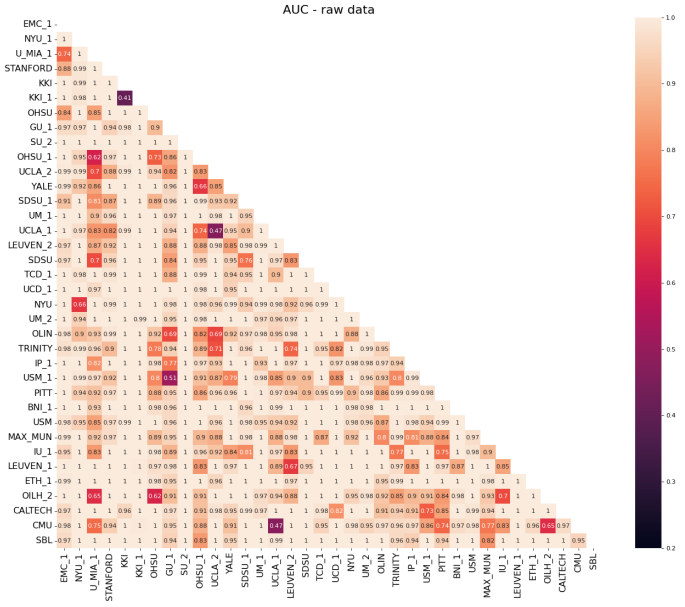


(a)

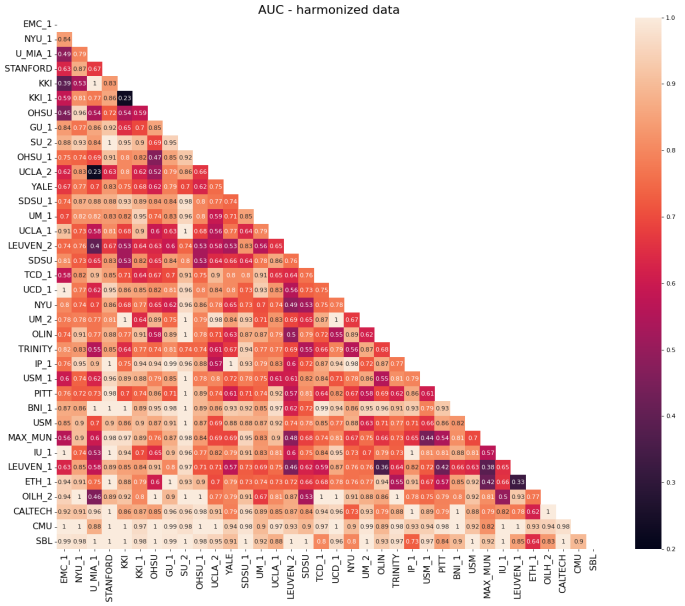


(b)

Figure 12.3: *Pearson-based (a) and wavelet-based (b) FC measures for feature 324 before and after harmonization with a separated plot for controls and subjects with ASD.*

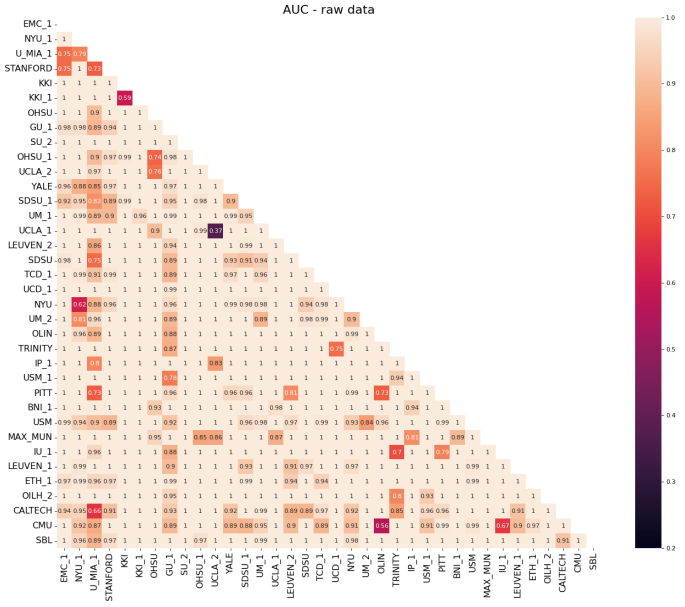


(a) *Heatmap with AUC score of a binary classification site vs site with non-harmonized data*

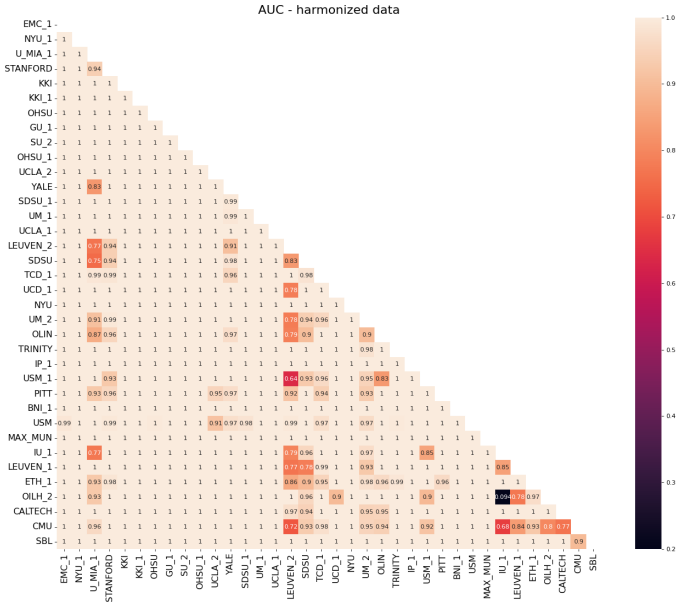


(b) *Heatmap with AUC score of a binary classification site vs site with harmonized data*

Figure 12.4: *Comparison of two heatmaps with AUC score of binary classification site vs site with* ***Pearson****-based correlation coefficients of non-harmonized (a) and harmonized data (b) classified using a Random Forest classifier*



(a) *Heatmap with AUC score of a binary classification site vs site with raw data*



(b) *Heatmap with AUC score of a binary classification site vs site with harmonized data*

Figure 12.5: *Comparison of two heatmaps with AUC score of binary classification site vs site of* ***wavelet****-based correlation coefficients of raw (a) and harmonized data (b) classified using a Random Forest classifier.*

Chapter 13

Deep neural network model for data classification

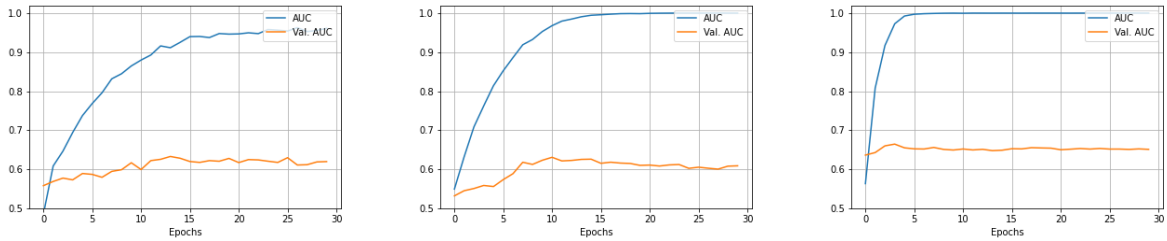
13.1 Definition of deep neural network architecture

In this work we performed a classification task, using a shallow neural network architecture originally built using Keras library[1] for Python. Since we are working under a permanent overfitting condition, we tried to build a network with as less capacity as possible, but still preserving good classification performances. To this end, we used Pearson FC measures to perform a control vs. ASD classification using the dataset ABIDE I + ABIDE II. We performed the optimization of our network using a shallow network comprising only 3 layers. By gradually adding neurons, we searched for their minimum number to employ in each layer. We started with a trivial network made by just 3 neurons in the first layer, 2 in the second and at last a single-output layer. Even if this was just an attempt to put a lower limit to the number of neurons, we noted that even with this configuration, the network tends to overfit our data. This is clear if we take a look at the learning curves in fig. 13.1 which shows the training and validation AUC values for three simple models: the first one (fig. 13.1a) corresponds the trivial structure just mentioned and it shows the classical trend of an overfitting condition. This trend is characterized by an increasing AUC curve on the training set over epochs, while the validation AUC score, after a few epochs, settles on a value and does not increase. We continued adding neurons to each layer and tried different configurations. With a configuration of 8-8-1 neurons (fig. 13.1b) the performances were slightly higher, so we set the second layer to 8 neurons and tried changing the number of the neurons in the first layer. Performances seemed to increase as the number of neurons increased, as it is possible to see in fig. 13.1c with a network consisting of 64-8-1 neurons that reached a validation AUC value of $\sim 70\%$. Adding more neurons, the network reached a stable performance value. Different attempts and the related classification scores are reported in table A1 .1 in Appendix. The first configuration that allowed stable performances was obtained with a structure 64-8-1. However, we chose to employ a slightly more complex network consisting of 264-8-1 and introduce regularizer layers such as dropout as we discuss in the following lines.

Beyond this number of neurons and layers performances were stable, and the addition of more neurons to create a more complex network was unjustified.

In our work we employed the neural network schematized in fig. 13.2. This network consists of 2 hidden layers made up of 264, and 8 neurons respectively. Both layers are activated by a ReLU

[1] https://keras.io/



(a) *3-2-1 neurons* ***(b)*** *8-8-1 neurons* ***(c)*** *64-8-1 neurons*

Figure 13.1: *AUC learning curve, shown of the train and validation sets, corresponding to deep neural networks trained on the entire non-harmonized dataset consisting of ABIDE I + ABIDE II. The different neural network architectures are: (a) 3-2-1 neurons; (b) 8-8-1 neurons; (c) 64-8-1 neurons*

function and separated by a Batch-Normalization and a Dropout layer with a dropout chance of 30%. After the 8-neuron layer, we added a second Batch-Normalization layer, before the output layer. At the end we put a single output layer with a single neuron for the classification output. This last layer is activated by a sigmoid function, which outputs a real number between 0 and 1.

To create the network we set the subsequent hyperparameters after a grid search on learning rate and number of epochs, setting a validation set of 25% of train data, and studying the evolution of the learning curves, to find the minimal parameters configuration which allow the best classification scores. We trained the model using binary cross-entropy loss function and optimizing parameters using Adam optimizer with a learning rate of 10^{-4}. Adam optimizer was chosen over Stochastic Gradient Descent (SGD) optimizer since with Adam we empirically achieved similar classification performances with a lower error than with SGD.

13.2 Definition of a domain-adversarial neural network architecture

As mentioned in chapter 11, a different approach to classification was fulfilled by using a domain-adversarial neural network (DANN). A domain-adversarial (or site-adversarial) network is a model able to learn from data both class and site information. This model allows to make predictions not influenced by biases due to sites.

Adversarial model with composition of loss functions

Our first attempt was to implement a model able to predict the category labels without any influence of site information. It consisted of two different branches, one for the control vs ASD prediction, and the other aimed to predict the site. With these two outputs, a loss is created by combining two different loss functions, one for the class and the other for site: the idea is similar to that proposed by Guan *et al.* [64]. This combined loss can be expressed in the form:

$$L = L_1 - \lambda L_2 \tag{13.1}$$

where L_1 is the loss for the binary control vs. ASD classification, that we want to minimize (a binary cross-entropy), while L_2 is the loss for site classification, (a categorical cross-entropy we aim to maximize in order to avoid to learn any information related to sites). The parameter λ is empirically set, to control the contribution of the site loss L_2 to the overall loss. This way, the

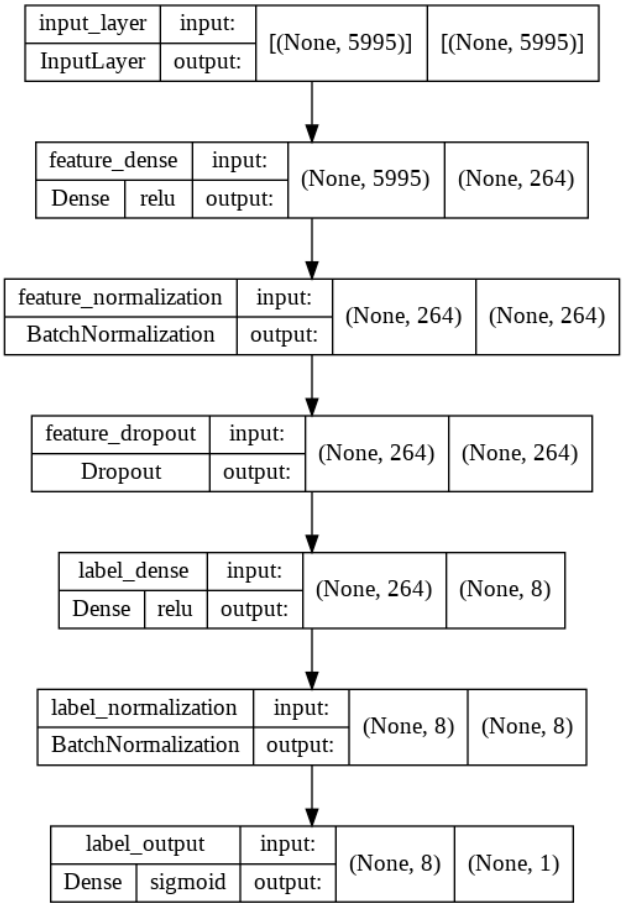


Figure 13.2: *Schematic structure of the deep neural network employed in this work for the discrimination between control subjects vs ASD patients. It follows a structure 264-8-1. Each box contains the layer name, layer task (Input, Dense, BatchNormalization etc..), activation function, input shape (number of neurons) and output shape.*

minimum of the total loss is reached with parameters that minimize the category classification loss and maximize the site classification loss. We empirically noticed that performances of this type of model are strictly linked to the value of the parameter λ. If a certain value gives an optimal performance on a certain dataset, on another dataset, for example the one created with different constraints to select the subjects, the best score is obtained for a slightly different value of λ.

Adversarial model with a gradient reversal layer

Our second approach was the creation of a domain-adversarial neural network following the idea described in chapter 9. Even though this network is based on the same principles of the neural network with combination of two losses (minimize the error on classification of controls vs ASD, and maximizing the one on site classification), it accomplishes its task in a different way. The model is trained to learn site-related patterns from data, and use these information in an adversarial way during the extraction of relevant feature for the classification. With respect to the model described in chapter 9, our model has a substantial difference in the structure of the site-classifier branch. The difference is due to the number of sites our data belong to: we deal with 36 sites when using the dataset of ABIDE I + II, while the domain-adversarial network of chapter 9 only concerns 2

domains.

To construct our domain-adversarial neural network, we employed the same number of layers and neurons as we used to build the deep neural network implemented in the previous paragraph. Then we embedded in this structure the domain-adversarial branch. We can describe the structure of this model as consisting on three parts: a feature extractor branch takes input data and creates an inner representation of them within its structure; at this point the network is forked into two branches: a label classifier branch for control vs ASD classification and a site classifier branch for sites classification. At the top of the site classifier branch, the gradient reversal layer was placed.

In details, the first branch namely the feature extractor, comprises the input layer, and the two hidden layer consisting on 264 and 8 neurons respectively. Between the two hidden layers we put a batch-normalization and a 0.3 dropout layer, as we did in the DNN. From this point on the network is splitted into two branches: the label classifier, similarily to the DNN consists of an output layer with a single neuron activated by a sigmoid function, for the output of control/ASD classification. The other branch, the site classifier consists of a gradient reversal layer at the top of it, followed by a layer with 16 neurons, a batch-normalization layer and the last layer: a multi-output layer with M neurons, being M the total number of sites that input data belong to. The effect of the gradient reversal layer can be regularized with a factor to weight the contribution of the site loss on the feature extractor parameters update during back-propagation. We empirically set this value to 0.3. The architecture of this network is illustrated in fig. 13.3.

The final layer of the site-classification branch is activated by a softmax function, and the loss function employed is the categorical cross-entropy, commonly used for multi-class classification tasks as explained in section 3.2.2.

To work with this loss function it is necessary to implement a **one-hot encoding** of the sites because functions like these can not deal with string variables such as site names. For this reason, firstly we have to assign a number to each site name like: site_a = 0, site_b = 1 and so on for all the M sites. However, if we stop to this point and use integer numbers $\{0, ..., M-1\}$ to compute the loss, our classification model would end up considering sites with higher number as "larger" than others. This would result in a suboptimal way to compute an error since, for example, the error between site 0 and site M-1 would be greater than the error between 0 and 1. This should be avoided in our case since all sites are equally important. For this reason working with increasing site label values is not the best way to encode site information. One-hot encoding provides a solution to this problem. This common approach converts each value to a vector of length M, containing all zeros but in the entry corresponding to the number of the site it is encoding, where it puts 1. For example if a datapoint belongs to a site m, its corresponding one-hot vector will be defined as $y_{im} = \begin{cases} 1 \text{ if } y_i = m \\ 0 \text{ otherwise} \end{cases}$ With this transformation, each site label becomes a binary vector, with just a single 1 places in correspondence to that specific site.

We tested and compared the two different adversarial models (the model with loss combination and the one with the gradient reversal layer) and we noticed that they showed similar performances. However, we choose to carry out all the subsequent analyses with this second model, for a main reason: the implementation of a network with a gradient reversal layer allows to create a site-classifier branch able to recognize sites, since the inversion of gradient only takes place at the top of this branch. This allows an update of weights related to this branch aimed to reduce the error on this branch for site classification and encourages it to learn site-related patterns. Then, during the back-propagation, the weights related to the site-classifier branch are updated in order to minimize the site loss. Conversely, as shown in equation 9.6, the weights related to the feature

extractor branch are updated in order to maximize this loss, thanks to the inversion of the sign of the gradient through the gradient reversal layer.

On the other hand, the creation of a network with combination of two losses would not allow this difference in parameters update, since the minus sign is related to the loss itself. This leads to an updating of both site-classifier and feature-extractor branches made in order to maximize the site loss. In this way, the site-classifier branch is discouraged to learn site related pattern, which is not what we aimed to create with this model.

In addition, there are studies. e.g. the one carried out by *Kamath et al.* [65], that compared the performances obtained with different implementation of adversarial networks, and, even if they work with different data (text data), they find the DANN the best performing implementation. Since this network appears to be more promising than the simple loss combination networks, we choose to employ this network in the following analysis.

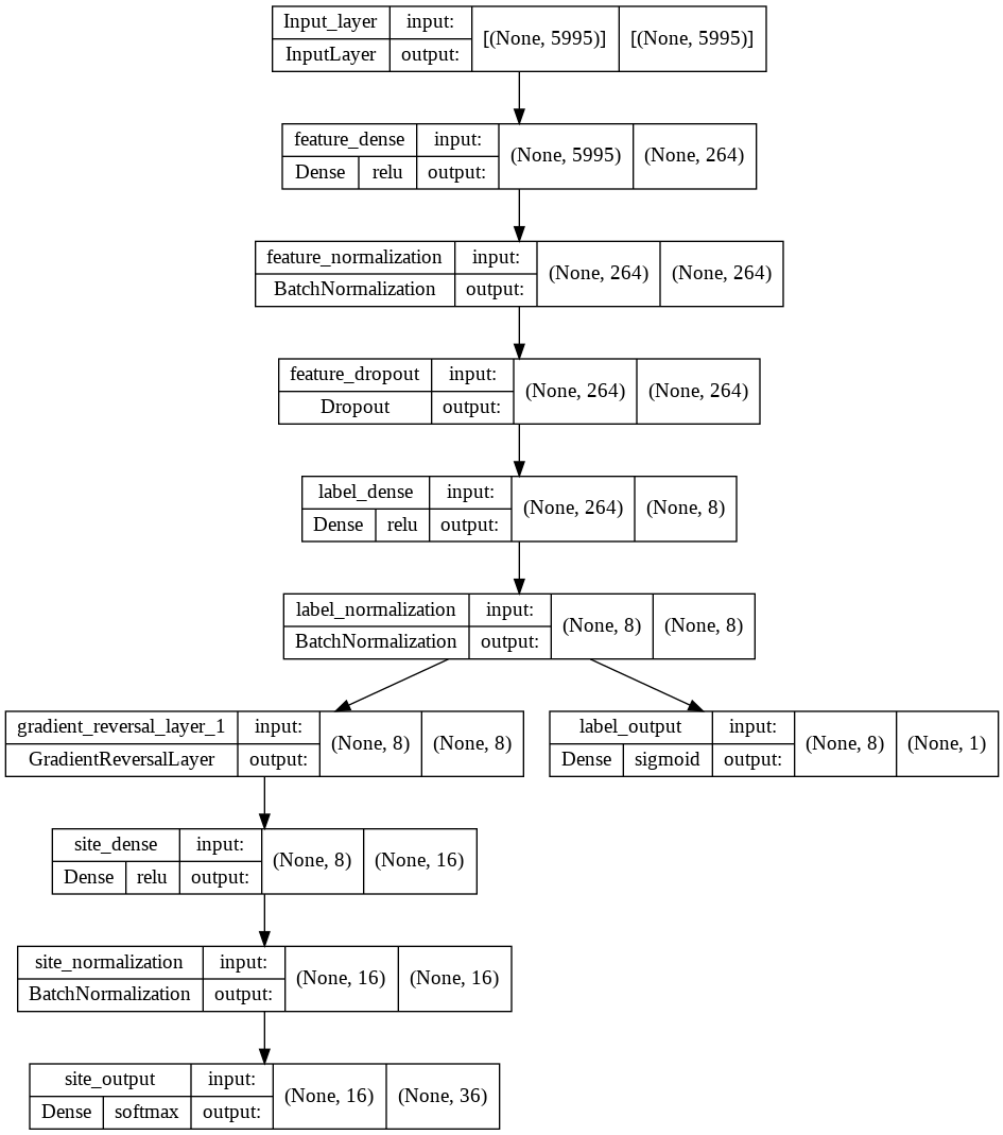


Figure 13.3: *Architecture of the domain-adversarial neural network with two outputs: a single-neuron output for binary classification of controls vs ASD and a multi-class classification for site classification. Each box contains the layer name, and the related task (Input, Dense, BatchNormalization etc..), activation function, input shape (number of neurons) and output shape.*

Chapter 14

ASD vs control discrimination performances

In the following sections we report results of classification of control subjects vs ASD patients. Classification is carried out by following different harmonization pipelines as explained in chapter 11. We mainly employed a deep neural network for classification of raw and harmonized data, and we compared results with those obtained with the adversarial neural network presented in chapter 13. In addition, we performed a restricted number of classification analyses making use of a Random Forest classifier. This allowed us to assess the advantages of employing a deep neural network rather than a conventional ML classifier.

From now on, in all the tables where results are reported, we will denote with the following labels the different harmonization pipelines:

- **Harmon_k-fold** is the pipeline where the harmonization is implemented inside the k-fold CV and the harmonization model is created only on control train data, and then applied to the rest of the dataset. The process is outlined in fig. 11.2.
- **Harmon_upstream** is the pipeline where the harmonization is implemented upstream, namely before the partition of the dataset, for each k-fold, into a train and a test subset. With this pipeline, the harmonization model is created using data of all controls as outlined in fig. 11.3.
- **No_harmon** is the pipeline where classification is carried out using raw data and no harmonization is applied.
- **Adversarial** is the pipeline where we use the domain-adversarial neural network giving as input raw data.

Furthermore, when we list in tables the different dataset partitions, we will denote as AB I+II / AB I / AB II the datasets created with patients belonging to the two different release of ABIDE dataset, as described in chapter 5. The choices and restrictions on covariates such as sex and ages have been discussed in chapter 11. When from these subsets we select only patients who kept their eyes open, we expressly specify it as **eye = open**.

14.1 ASD vs control discrimination with Pearson-based FC

The first classification analysis was made on Pearson correlation coefficients.

The results of ASD vs control discrimination obtained when the Pearson-based FC measures are classified have been evaluated in terms of the mean AUC scores across the 5 folds in the 5-fold CV scheme. We listed in table 14.1 the results obtained with DNN model we have developed, following different harmonization implementation and with the domain-adversarial network.

In table 14.2 we compared some of these results with a Random Forest classifier to check whether or not it is appropriate to use a deep neural model instead of a conventional machine learning algorithm. The comparison with Random Forest were performed on the whole dataset ABIDE I+II, and on the subset consisting of the ABIDE I subjects with open eyes only, since on this dataset we obtained higher classification performances. Results of DNN and RF concern the three pipelines: hamon_k-fold , harmon_upstream, and no_harmon.

Dataset	Harmon_k-fold	Harmon_upstream	No _harmon	Adversarial
AB I+II	71±1	74±2	73±3	70±3
AB I	71±3	74±2	72±3	71±4
AB 2	63 ±5	68± 6	64± 4	66 ± 4
AB I+II, eye = open	72±3	75±4	72±3	71±3
AB I, eye = open	73± 1	76±2	72±1	72±4
AB II, eye = open	66±3	70±6	69±6	68±6

Table 14.1: *AUC score obtained with the deep neural network, using Pearson-based FC measures and following different harmonization procedures.*

Harmon. pipeline	AB I + II		AB I eye = open	
	DNN	RF	DNN	RF
Harmon_k-fold	71±1	66+2	73± 1	70± 3
Harmon_upstream	74±2	72+1	76±2	71± 2
No_harmon	73±3	65+1	72±1	68± 3

Table 14.2: *Comparison between AUC scores of a DNN and a RF classifier, using Pearson-based FC, for the datasets: ABIDE I + II and ABIDE I with open eyes*

Discussion

From table 14.1 it is possible to notice that using the whole dataset does not lead to an improvement of results with respect of the use of ABIDE I dataset only. On average, the best classification performances are obtained using ABIDE I data with only open eyes. Results obtained with ABIDE II dataset are significantly lower this can be an effect of a greater variability of data in this dataset. Even if almost all the AUC scores are compatible with each other, with the pipeline of upstream harmonization, the mean AUC value is systematically higher than the other three pipelines.

Table 14.2 shows the comparison between a Random Forest and a DNN. It appears clear the advantage in using the DNN since the mean AUC value is significantly higher than the one achieved with a Random Forest. Results increase when we limit the analysis to a more homogeneous dataset collecting only patients with open eyes belonging to ABIDE I. Random Forest performances as well are boosted when following the pipeline of upstream harmonization. Since this is a common trend even with different coefficients, for a proper discussion we refer to section 14.4.

14.2 ASD vs control discrimination with wavelet-based FC

Following the same workflow as with Pearson-based FC, we run the classification and harmonization pipelines using wavelet-based FC measures. We used different wavelet-based measurea as described in section 11.1. The results are listed in table 14.3.

Coefficients	Dataset	Harmon_k-fold	Harmon_upstream	No_harmon	Adversarial
w_in	AB I+II	65± 2	74± 2	66± 2	67 ±2
	AB I	67± 3	73 ±2	68± 3	68± 2
	AB II	62 ±4	68± 4	63± 4	62± 4
	AB I+II, eye=open	65 ±4	71 ±3	66 ±3	66± 3
	AB I, eye=open	69± 4	74± 6	67± 3	67 ±4
	AB II, eye=open	66 ±2	69 ±3	67± 2	68 ±2
w_out	AB I+II	60±1	71±3	63±2	60±1
	AB I	61±2	69±3	62±2	61±2
	AB II	56±3	69±4	60±4	59±4
	AB I+II, eye=open	63±2	68±3	64±2	63±3
	AB I, eye=open	63±5	74±3	71±4	72±2
	AB II, eye=open	63±5	70±3	63±3	62±3
w_stack	AB I+II	65± 2	71± 2	62± 2	64± 1
	AB I	66± 3	73 ±3	65± 3	66± 2
	AB II	62 ±3	65 ±3	61 ±4	60 ±2
	AB I+II, eye=open	64± 4	70 ±3	66± 3	65 ±3
	AB I, eye=open	64 ±5	68 ±5	64 ±3	64 ±4
	AB II, eye=open	66 ±2	68± 3	66± 4	65 ±5
w_diff	AB I+II	67± 3	70± 3	66 ±3	66 ±3
	AB I	70± 3	72 ±2	68 ±3	68± 3
	AB II	61± 4	65± 4	64 ±5	65 ±4
	AB I+II, eye=open	70± 1	73± 2	71 ±1	68 ±2
	AB I, eye=open	71± 4	74 ±3	71 ±3	68 ±2
	AB II, eye=open	62 ±4	66 ±3	68 ±3	64± 5

Table 14.3: *AUC score obtained with the deep neural network, using the 4 different wavelet-based coefficients. For each partition of the main dataset (AB I+II) results are listed following all the harmonization pipelines described in chapter 11*

As we did with Pearson-based FC, we compared the results obtained using a DNN with the results obtained with a Random Forest. They are listed in table 14.4.

Harmon. pipeline	AB I + II		AB I eye = open	
	DNN	RF	DNN	RF
Harmon_k-fold	65±2	62±5	69±4	62±5
Harmon_upstream	74± 2	74±3	74± 6	69±3
No_harmon	66±2	59±3	67±3	61±5

Table 14.4: *Comparison between AUC scores of a DNN and a RF classifier, using w_in coefficients, for the datasets: ABIDE I+II and ABIDE I with open eyes*

Discussion

Looking at table 14.3 we can notice that some coefficients are more significant than other in classification between controls and ASD. w_in outperform w_out and stacking together these two measures to obtain w_stack does not bring to an improvement in the average classification performances. The best way to combine information from w_in and w_out seems to be the element-wise subtraction in order to create w_diff. The latter is an effective way to create a measure that carries information in a similar way as Pearson correlation coefficients do. w_diff compresses the information held by w_in and w_out into a single coefficient without further increasing the dimensionality, as it happened with the definition of w_stack.

The best classification performances have been obtained using the dataset ABIDE I with only open eyes. While using just the ABIDE II dataset leads to the lowest classification scores. As we noticed from the results obtained with Pearson-based FC, the pipeline of upstream harmonization leads to an increase of classification performances. With the other three pipelines: harmon_k-fold, harmon_upstream and no_harmon we achieved on average similar performances.

14.3 Classification performances with PCA

As mentioned in section 11 we chose to run PCA analysis on the FC measures that gave the best classification results in the previous analysis. Comparing the average scores obtained with Pearson-based and with wavelet-based FC, we can notice that with Pearson we achieved a better separability between controls and subjects ASD. Furthermore, to not overload this table, we limited our choice of the datasets to the whole ABIDE I+II and ABIDE I subjects with open eyes only. We have chosen the latter because we obtained on it slightly higher AUC scores with respect to the other subset selections.

We started our analysis choosing the proper number of pricipal components to explain $> 90\%$ of variance and then gradually reduced this number as reported in table 14.5.

PCA explained variance [%]		
Number of PC	ABIDE I + II	AB I eye=open
800	93	–
400	78	98
200	62	77
100	48	58
50	36	41
20	24	26

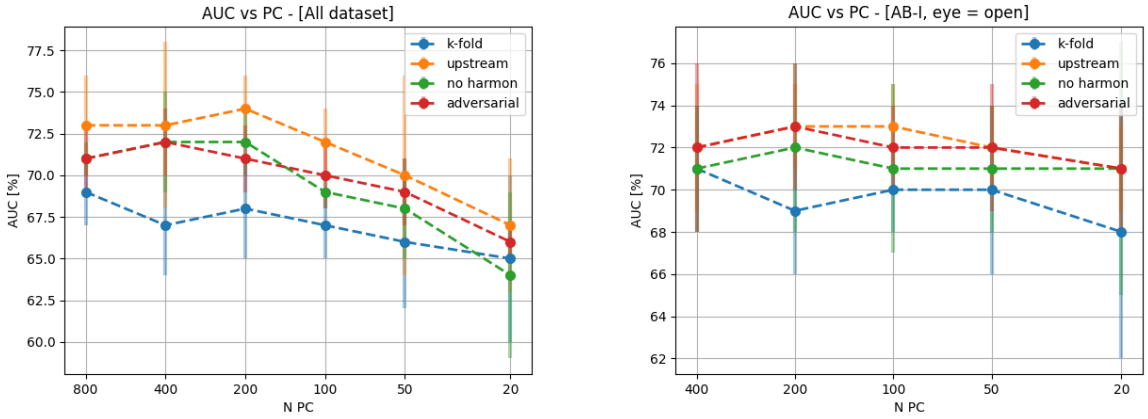
Table 14.5: *Explained variance [%] for different numbers of principal components (PC), and for the whole dataset of ABIDE I+II and the subset of ABIDE I subjects with open eyes.*

As explained in section 3.3 the number of principal components it is possible to extract is limited by the dataset size. It must be $N\_pc \leq min\{n\_samples, n\_features\}$. For this reason the first row of table 14.5 lacks an entry for the dataset ABIDE I with open eyes. Using this dataset, we have in fact, an amount of training subjects (from which principal components are extracted) lower than 800 samples.

We followed the same pipelines as we did for Pearson-based and wavelet-based FC, following the four different harmonization procedures. The results obtained from this analysis are reported in table 14.6 and shown in fig. 14.1.

PC	Dataset	Harmon_k-fold	Harmon_upstream	No_harmon	Adversarial
800	AB I+II	69±2	73±3	71±1	71±2
400	AB I+II	67±3	73±5	72±3	73±3
	AB I, eye=open	71±3	72±3	71±3	72±4
200	AB I+II	68±3	74±2	72±2	71±2
	AB I, eye=open	69±3	73±2	72±4	73±3
100	AB I+II	67 ±2	72±2	69±1	70±2
	AB I, eye=open	70±2	73±2	71±4	72±2
50	AB I+II	66±4	70±6	68±3	69±2
	AB I, eye=open	70±4	72±3	69±3	72±3
20	AB I+II	65±5	67±4	64±5	66±1
	AB I, eye=open	68±6	71±2	71±6	71±3

Table 14.6: *AUC scores obtained with the deep neural networks, using a decreasing number of principal components and following different harmonization procedures*



(a) *ABIDE I+II* ***(b)*** *ABIDE I - only open eyes*

Figure 14.1: *AUC scores obtained from in the control vs ASD classification of data of the ABIDE I_II sample (a) and on the sample of ABIDE I subjects with open eyes (b), using a decreasing number of principal components for Pearson-based FC measures. The results obtained with different analysis are represented with different colors: AUC scores on data harmonized inside k-fold are marked in* ***blue****, AUC scores on data harmonized upstream in* ***orange****, AUC scores on raw data (not harmonized) in* ***green*** *and AUC scores classification with the adversarial network in* ***red****.*

Discussion

As it is possible to notice from table 14.6, reducing the dimensionality of the problem using PCA is an effective way to keep relevant information from data and discard redundant or uninformative data. Comparing these results with those listed in table 14.1, we can notice that, especially with a higher number of principal components, classification performances are not affected by this reduction of dimensionality. The higher the percentage of variance explained with PCA, the higher the classification performances achieved for both the ABIDE I + II dataset and the ABIDE I subsample of subjects with open eyes only.

At a first glance we can notice from figure 14.1 that reducing the number of principal components leads to a gradual reduction of the classification performances. This trend is more evident using the whole dataset, while with the ABIDE I open eyes only dataset, we obtain a flatter performances curve. A reduction from 5995 to 800 or 400 feature is a substantial reduction since we are lowering our dimensionality by an order of magnitude still preserving high AUC classification scores, comparable to those obtained without PCA. In this analysis as well, following the pipeline of upstream harmozation, leads to an improvement of performances. The suspected reason for this common trend are explained in section 14.4.

14.4 Discussion on ASD vs control classification results

14.4.1 Comparison between Pearson-based and wavelet-based FC measures

From the results reported in section 14.1 and 14.2 we can observe that average AUC scores obtained with Pearson-based FC measures are systematically higher than the wavelet-based ones. We can therefore state that working with wavelet-based FC measures obtained through the processing described in section 7.2 does not bring advantages to this type of analysis. A possible reason for this outcome is the loss of information that occurs during the extraction of these FC measures. To this regard an important role is played by the introduction of redundancy and randomness into this analysis. Starting from two timeseries, the wavelet transform computes a 2D matrix for each timeseries and uses its values to calculate the cross-power spectrum, as explained in the dedicated paragraphs of section 7.2. Doing so, we create a redundant representation of timeseries data moving from one dimension (timeseries) to two-dimensional data (cross-power spectrum matrix). We furthermore compare the cross-power spectrum obtained from the wavelet transform of each pair of timeseries, with a red-noise cross-power spectrum to assess the significance of our data. In the end we shrink again our dimensions to reduce to a single data point representing a measure of FC.

Since differences of traits between controls and ASD are not very strong, this process can cause the loss of this weak information, which does not happen with the straightforward computation of the linear correlation by means of the Pearson coefficients.

The lower discrimination scores achieved by wavelet-based FC are found both for the in-phase and the counter-phase time percentage (w_in and w_out), and also for their combination in a double-sized array. The best way to combine wavelet information, however, seems to be the creation of what we called w_diff, obtained by subtracting, for each subject, coefficients of counter phase w_out from coefficients of in-phase w_in. However, average scores obtained with wavelet-based FC across all pipelines and all datasets are lower that those obtained with Pearson-based FC measures.

The use of wavelet-based FC measures still represents an important and different way to extract a measure of correlation between two signals. However, for this work, with our classification task, these FC measures are not the most suitable.

14.4.2 Effects of the heterogeneity of the sample composition on the classification results

To discuss the benefits of reducing the dataset size to run analysis on a more homogeneous cohort of subjects, we focus on table 14.1 and 14.3.

We can state that the eye status of patients during the scan has a non-negligible impact on the outcome of classification, since limiting the dataset only on subjects with open eyes systematically improves classification scores of more than 1% for all datasets (ABIDE I+II, ABIDE I or ABIDE II). As introduced in chapter 1.1, there can be differences in brain functional areas between patients with open eyes and patients with closed eyes. This may happen because of the activation of different brain areas, especially those related to the processing of visual stimuli. Another variability introduced by patients with closed eyes, is that during the scan, some of them may fall asleep, hence heavily modifying the functional connectivity network of their brain. For this reason, removing these sources of variability could bring to a cleaner distinction of relevant distinctive patterns between controls and patients with ASD.

We can also notice that results on the dataset ABIDE I + II are mostly driven by the data belonging to ABIDE I. This happens also if we restrict the analysis to patients with open eyes only. In fact, putting together the two datasets does not lead to a great improvement of scores than limiting the analysis just to ABIDE I. It is possible that this trend is due to two main factors: the greater number of ABIDE I subjects with respect to those in ABIDE II left after all the selection criteria inposed on covariates, and the greater number of subjects provided per site, in ABIDE I dataset with respect to ABIDE II. On average, as it is possible to notice from table 5.1, sites belonging to ABIDE II, with few exceptions, provided less populated sites and this leads ABIDE II data to present a greater variability.

14.4.3 Comparison between the deep neural network and the Random Forest classifiers

It is always a good practice to compare results obtained with a complex model, with the ones obtained with a simpler machine learning classifier. Instead of immediately opting for a complex model, simpler methods should be tried to establish a baseline performance and to allow a meaningful comparison.

The principle of Occam's razor, when applied to machine learning, demands that if two model perform quite the same, the simpler of the two should be picked [66]. This is especially true for our data, since we start from a situation of overfitting and there is no need to choose a complex model a priori if this choice is not supported by data. This is the same principle that lead us during the choice of the best architecture and the best number of layers and neurons of the deep neural network, as described in chapter 13.

By comparing the results obtained by the deep neural network model with those obtained by a Random Forest classifier, (see table 14.2 and 14.4), it is possible to deduce that a deep model is more suitable to find characteristic patterns to discriminate between controls and subjects with ASD than a simpler Random Forest classifier. Scores obtained with a deep model are systematically much higher than those obtained with a Random Forest classifier. The only exception is with data classified with the upstream harmonization pipeline. However, as explained in the following section, this procedure leads to biased data and, as a consequence, the results obtained with this pipeline are not reliable. We are still going to use Random Forest classifier in some of the remaining analysis just to make a comparison, even though we state that a deep model is more suitable to understand differences between the two classes of interest.

14.4.4 Effect of different harmonization pipelines

Another common trend that stands out from all the classification trials we performed is the systematic improvement of AUC scores when we use data harmonized with the pipeline of upstream harmonization (explained in section 11 and outlined on flowchart in fig. 11.3). This improvement disappears when we implement the harmonization inside the k-fold cross validation procedure (sketched in the flowchart 11.2). These two implementations differ on the order by which harmonization is implemented: with upstream harmonization we harmonize the whole dataset and later we split it into train and test to run the cross validation procedure. Conversely, when the harmonization implemented inside the cross validation procedure, we run the harmonization algorithm only after the division of the dataset into a train and test subsets, thus fitting the harmonization function on the train dataset only. We suspect that the higher results obtained when data harmonization is performed upstream are driven by a misleading way to proceed. The classification results are affected by a bias due to data leakage which occurs if we implement the harmonization upstream. It is also possible that some good results reported in similar studies, such as the paper by Ingalhalikar *et al.* [63], have been obtained with the harmonization implemented in this way.

In our analysis we have data leakage when we harmonize the entire dataset before splitting it into train and test. In this case, when we use the entire dataset to train the model, they already contain information about the test data. In fact, harmonized train and test data have been created using covariates, features and other information belonging to the whole dataset. We can then state that the right way to implement the harmonization process is inside the k-fold cross validation. In a more general case, if we are not running a cross validation procedure, it is important to implement harmonization of data after the splitting of the dataset into a train and a test subset. In this way, we can create the harmonization model only on control subjects belonging to train dataset. Thus, when the model is trained with this dataset, it is not biased by external information from the test data. We can then harmonize data belonging to test by applying the harmonization model with parameters estimated using only the train dataset.

Comparing results of harmonized data with results on raw data we did not notice any particular improvements. As we have seen in chapter 12, harmonization is an effective strategy to reduce site-related effects of features, but it has no particular beneficial effect on the ASD vs control classification performances.

This is likely due to the limited information related to ASD vs control discrimination contained in our data. Distinction between healthy and ASD subjects based only on functional connectivity data is not a straightforward task, and it is possible that these results are the best it is possible to achieve with these data. However, the harmonization procedure can become a useful tool to remove some noise from data and obtain a cleaner assessment of what features are the most discriminating between controls and ASD. This theme is entirely addressed in chapter 15, which is devoted to the study of feature importance.

14.4.5 Effect of dimensionality reduction

The same attention we paid in the unbiased implementation of the harmonization step was applied for the PCA. As mentioned in the overview chapter 11, it is possible to introduce a bias in the analysis by computing the principal components on the whole dataset, before it is separated into train and test sets. For this reason, we implemented the PCA by extracting the principal components from the training dataset only and by applying the transformation to both the train and test datasets.

Another aspect that comes up by comparing the results with PCA with the results obtained without implementing it, is a light improvement of the performances of the adversarial learning with

respect to the classification schemes based of other harmonization pipelines. An explanation to this is the reduced source of noise and, consequently, of errors deriving from dimensionality reduction, that leads to a reduced variability in site-related features. This makes the confounding feature more recognizable, since they are no longer affected by noise. Reducing the dimensions of data then, could lead to a smoother weight updates that does not affect classification weights as much as it does with the whole features data.

What arises from this analysis, is that PCA represents an important strategy to tackle the problem of dimensionality. It is plausible that among all the 5995 features, a big percentage of them are uninformative for control vs ASD classification and they just increase dimensions of data without any benefit, or worse, just adding noise. This is linked to a biological reason: we can in fact imagine that there are some brain areas whose activity is not affected by the ASD condition. Following this explanation, all the FC measures representing a the relationship between them are just similar between the two groups and do not bring any benefit in this discrimination problem. For this reason, reducing the source of noise due to this data-points leads to the creation of an equally informative dataset, but with a reduced source of error.

On the other hand, the reduction of dimensionality with PCA has a non-negligible downside in this work. By reducing dimensions of data, we lose some information about the feature. When an array containing 5995 feature is projected into a subspace of 800 or lower dimension, some of the information contained in it is discarded. This leads to the loss of biological information that each feature represents which is not possible to exactly retrieve. In fact, after a dimensionality reduction, it is only possible to retrieve an approximated value of the features in the original space. For this reason, to extract relevant features, linked to relevant altered functional connections, we need to work on the original high dimensional space of the whole set of features, as we did and reported in chapter 15.

14.4.6 Advantages in using deep adversarial neural networks

Looking at tables 14.1 and 14.3 we can notice that the adversarial model does not bring significant advantages in the AUC classification scores. AUC scores obtained with DANN are on average the same as obtained with other harmonization pipelines (except for the upstream harmonization pipeline that as we stated above, leads to biased results). However, a possible drawback of using this type of network with these data, is the possible confusion introduced by dealing with a huge number of sites. So far, in a number of different papers, adversarial networks are employed with only two domains [57, 65, 64]. The domains can be considered as the equivalent of our sites. In our data we deal with 36 sites when we consider ABIDE I+II, or, in the best case scenario, about 15 sites when we restrict the analysis to ABIDE I or II with open eyes only. In this way, even if the site-predictor branch is encouraged to learn site-distinctive traits, it is not always able to distinguish among 36 sites, so this can cause confusion during the back-propagation on the update of weights related to the feature extractor branch.

For this reason it is possible that when dealing with more than two domains (sites) this implementation of an adversarial network does not bring advantages in classification problems. Nevertheless, the approach of classification with a DANN could represent a promising tool to employ in a dataset of great dimensions. With respect to the classification with analytical harmonization, this approach is totally data driven, and does not require additional harmonization pipelines. This allows to perform classification avoiding the risk of introducing biases such as it happens with the upstream harmonization procedure. It is likely that if each site of the ABIDE dataset was more populated, this approach could have brought a real advantage to the classification of controls vs ASD. As a future development, if the dimensionality of the dataset allows the increasing of model

complexity, it would be possible to develop an improved version of a DANN, taking into account the age information as well. This would be possible with the implementation of a third output branch acting as an age regressor. The model employed in this Thesis lacks this information because of the reduced dimension of the dataset, not sufficient to allow the identification of this additional information among data samples.

Chapter 15

Explainability of ML models with SHAP

In this chapter we present and discuss the results obtained by implementing a ML explanation model using the DeepSHAP algorithm described in chapter 10. We also compared that approach with a more conventional Random Forest feature importance assessment. Here we present a brief summary of all the analysis we performed using the feature importance information.

We choose to run this analysis only on Pearson-based FC measures since they gave the best classification results, i.e. the best separability between controls subjects and ASD patients. We used the whole dataset consisting of ABIDE I + ABIDE II subjects with the same characteristics as we choose for classification: only males and with an age range of 5-40 years. In addition, we reported in the Appendix some additional results obtained on the subset of ABIDE I subjects with open eyes.

In this section we report on our investigation on what are the most important features that guide the prediction of a machine learning model. These features are representative of altered functional connections between brain areas. The study of altered connections allows to identify the biological areas relevant for the distinction between healthy controls and subjects with ASD. What we are looking for in this section is whether there are some features whose contribution was greater than others during the distinction between controls and ASD data. We check if these features are recurrent, regardless the type of analysis we are carrying out, or the machine learning classifier we use. In other words, we checked for the consistency of altered functional connections across different data processing pipelines.

As a first introductory analysis, in section 15.2.1 we started identifying the most important features that lead the classification of the DNN models, using DeepSHAP. We extracted the most important features for each of the harmonization pipeline discussed in section 11. Our goal is to check whether the presence of harmonization, implemented by different strategies, significantly alters the contribution of features in the output of the model. In section 15.2.2, following the same harmonization pipelines, we extracted important features from a Random Forest classifier. For this simpler machine learning model: a Random Forest classifier, feature extraction is implemented by the use of the feature_importances_ method[1] provided by *sklearn*.

This is, however, just a first, preliminary inspection of important features, and for this reason we limited our focus to the first 20 important features for each pipeline. We chose to limit to this number because in this way we can easily make a visual assessment of the contribution of these features and compare it across the different analysis pipelines.

[1] https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

A more quantitative analysis, was carried out afterwards. We selected only 5 pipelines among all the one considered earlier, because we regarded at these ones as the most significant ways to proceed. In section 15.2.4 we will explain more in details what these analysis are and why we excluded the other ones, and we will present the consistent results in terms of common important features among these 5 analysis pipelines. Starting from them, we examined the biological meaning of these features and the brain areas they involve, to create a histogram of the most involved brain areas in the discrimination of healthy and ASD subjects.

15.1 Implementation of the SHAP explainability method

15.1.1 Implementation of the DeepSHAP algorithm

To instantiate the desired class of SHAP called *DeepExplainer*, we required the trained deep model and a fraction of the training data to use as background. We can choose as many background data as we want, keeping in mind that the bigger is this background subset N_{bkg}, the more accurate is the computing of Shapley values, but the more computationally expansive this process is. In particular, using a large amount of background data, it would occupy a vast amount of RAM memory and eventually run out of it. However, since the error we make on Shapley values linked to this choice is $\sim 1/\sqrt{N_{bkg}}$, a background dataset consisting of 100 samples is already a good compromise to have a relatively small error. We chose though a background size of 500 samples for the entire dataset and a size of 100 samples when working on ABIDE I only open eyes, since we had a reduced number of train data to collect background data from.

Once the *explainer* class is created, we can input as many test dataset as we want to explain, and for each one of them, the *explainer* outputs an array containing a Shapley coefficient for all the n_features of each test data. Once a batch of test data is entered, we obtain a matrix of the same size of the test dataset we used, shaped n_samples x n_features, containing all the Shapley values for each feature of each test data. We chose to input all cases of the test datasets to obtain a more accurate estimate of feature importance. Since we are presenting our results following a k-fold cross validation scheme, we implemented this procedure inside the k-fold CV. To do so and to get a final and general result, we computed the Shapley values on the cases of the test set of each fold and collected them. At the end of the k-fold CV, we obtain a matrix containing Shapley values for each sample of the whole dataset. We can thus proceed to visualize the results.

15.1.2 Different visual representations of SHAP feature importance

For each test data we can visualize the contribution of each feature on pushing or pulling the predicted output from the baseline output of our model. An example is shown in figure 15.1. This type of plot, called *waterfall plot*, shows the result of the most important features on a single test data. We have the baseline value of the model in the lower right corner and on the top left the output of the model on this test subject. We recall that the baseline value (or reference value) is the average output of the model, computed using the background dataset chosen between train data. When a test sample is given as input to the model, a prediction is made. During the prediction, each feature of this sample acts by pushing or pulling the predicted value towards greater values (positive contribution) or lower ones (negative contribution) with respect the baseline value. The contribution of each feature of this sample, to the output, can be visualized with this plot. Positive Shapley values, representing positive contribution to our output are colored red and negative ones are blue. Even if this is just an example taken from a single test data, it is representative of all the analysis where there is not a feature whose contribution is way greater than the others. In fact

each feature makes a small contribution to the final outcome, but the sum of them push the model towards an output.

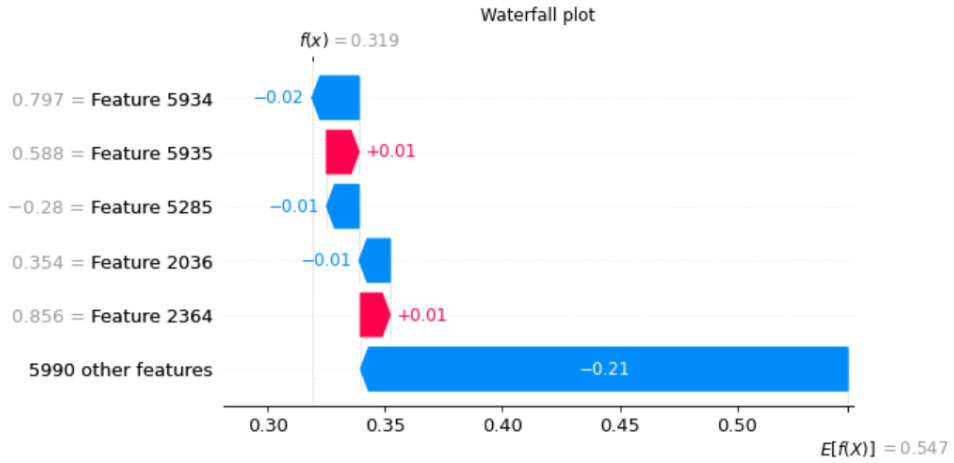


Figure 15.1: *Example of a waterfall plot obtained on a single test data. For each row the feature name, its value and its contribution to the final output are displayed: negative contributions are colored blue and positive ones red. The reference output is also marked in the bottom right corner, and the output of this instance on the top left.*

If we want look at the overall result on all the test dataset, it would be inconvenient to create a plot like 15.1 for each sample. For this reason we have to look at a summary plot called *force plot*.

The force plot of feature importance can involve either the module of Shapley values or the Shapley values themselves (positive and negative). An example of the latter is shown in fig. 15.2a: this plot shows the contribution of one particular feature to the output computed for each instance, i.e. each sample of the test data [2]. In other words, for each instance of the test dataset, we have an array of Shapley values: one for each feature, and these values (for each feature) are represented by a dot, placed to the right or to the left of the thin black line corresponding to the value of zero. The zero value indicates that for a certain test data, that corresponding feature did not contribute to the output. The position in respect to the black vertical line indicates the sign of that contribution: on the right we found positive coefficients that gave a positive contribution pushing the output towards values greater than the reference output, and on the left are placed those that had a negative contribution. Pushing the output towards greater values (closer to 1 than to 0), means that the input data is being classified as ASD (if ASD, as in our case, is associated to the label 1). This process of representing the Shapley values for each feature in a plot, is repeated for each instance of the test dataset to obtain a scatter plot of the Shapley values, where the features are ordered by importance. Each spot is also colored red or blue. The color indicates, the sign of the difference of the feature calculated with respect to the mean value of the feature across all the entries of the test dataset. This additional information is very useful to understand whether a greater or a smaller value of a feature pushes the model towards one output or towards the opposite (0 or 1). This information is useful to answer for example the following question, which is particularly relevant for clinician to interpret the results of a fully data drive analysis: "is a stronger or a weaker correlation between two areas that contributed the most to moving the prediction towards ASD with respect to control subjects?".

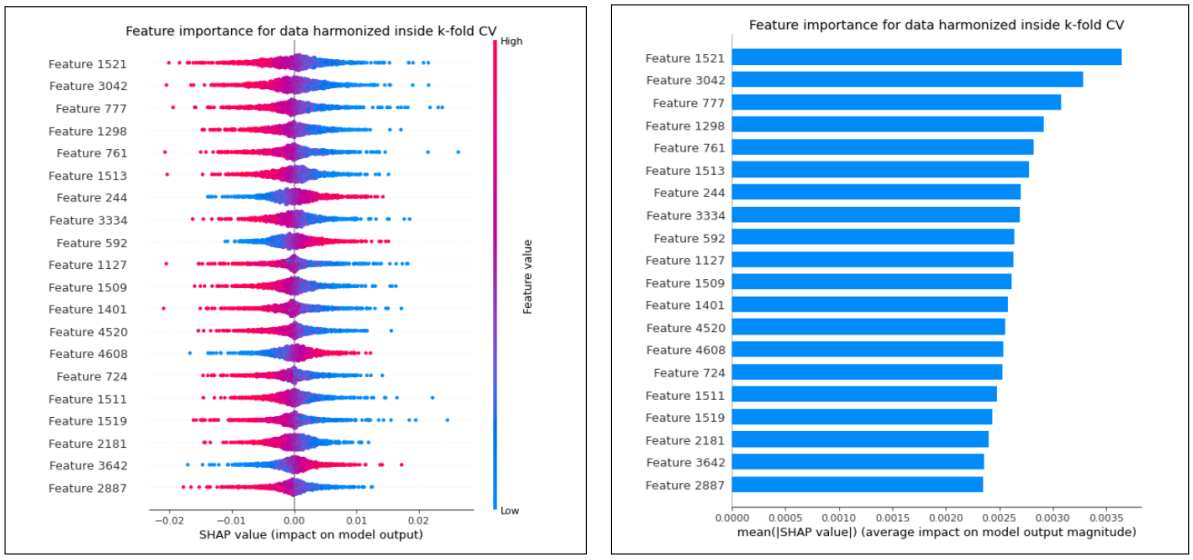
[2] https://shap.readthedocs.io

However, for our data, this kind of plot is not the most suitable for readability because of the large amount of features and the small contribution of each feature to the final outcome, with respect to other features. This results in a smoothed scatter plot for each feature, containing many but unnecessary details which make this kind of plot not the clearest to assess how much a feature is important in an absolute sense for a model.

Alternatively, we can visualize the contribution of each feature, by considering the absolute Shapley values. In fig. 15.2b a bar plot representing all the first 20 important features is reported. Each bar represents a feature importance, computed from equation 10.6.

For a better readability, we preferred to report the results in terms of absolute Shapley values, regardless its positive or negative contribution to the output.

From this plot, we already have a more immediate understanding of what the most important features are. As we noticed in commenting fig. 15.1, there is not a standing out feature, or a group of features that show a contribution way greater than the others, but in fact feature importance has a smoothed descending trend.



(a) *Plot of dotted Shapley values*

(b) *Bar plot of Shapley values*

Figure 15.2: *Two different but closely related force plots of Shapley values showing the first 20 most important features obtained from data where the harmonization procedure was implemented inside the k-fold CV scheme. (a) For each feature, each instance (each test sample) is represented by a dot, whose position indicates a positive or negative contribution to the output, while the color represents the magnitude of the feature, compared with the average value of that feature across all the test dataset. (b) For each feature the mean absolute value of all the Shapley values computed for each instance is represented.*

15.2 Identification of the most relevant features for DNNs and Random Forest

15.2.1 Feature importance for DNNs

In fig. 15.3 we show the first 20 most important features for the DNNs, obtained from each harmonization pipeline described in section 11. From this plot, we can notice that some features seem to

be persistent through all four pipelines.

We find that: feature 592 and 1521 appear, even if in different order of importance, in all our four analyses. If we consider only common features between harmonization in k-fold, harmonization upstream and raw data, two more features appear to be persistent: 1513 and 3334. We run this latter analysis excluding the adversarial network approach to allow comparison with results of the next section 15.2.2, extracted from Random Forest.

Since each feature is representative of a correlation between two brain areas, we report here the corresponding areas involved:

- Feature 592: correlation between Right Middle Temporal Gyrus (anterior division) and Left Superior Temporal Gyrus (anterior division)
- Feature 1521: correlation between Left Angular Gyrus and Right Middle Temporal Gyrus (posterior division)
- Feature 1513: correlation between Left Angular Gyrus and Right Temporal Pole
- Feature 3334: correlation between Right Parahippocampal Gyrus (posterior division) and Right Accumbens

15.2.2 Feature importance for Random Forest

In this section we repeated the same analysis we showed for a DNN in fig. 15.3, with a Random Forest classifier. We plot the first 20 most important features obtained from the three main pipelines with harmonization in k-fold, harmonization upstream, and no harmonization, followed by a Random Forest classifier. Features are extracted using the function *feature_importance_* provided by *scikit-learn* library. The results are shown in fig. 15.4.

From the three plots we notice that 7 different features are common to the three analysis pipelines: feature 710, 1127, 1703, 748, 2735, 1400 and 2811. As we did for DNNs we report the brain areas involved in these correlations:

- Feature 710: correlation between Right Middle Temporal Gyrus (temporo-occipital part) and Right Thalamus
- Feature 1127: correlation between Left Postcentral Gyrus and Right Postcentral Gyrus
- Feature 1703: correlation between Right Lateral Occipital Cortex (inferior division) and Right Supramarginal Gyrus (anterior division)
- Feature 748: correlation between Left Middle Temporal Gyrus (temporo-occipital part) and Right Thalamus
- Feature 2735: correlation between Right Precuneous Cortex and Right Middle Temporal Gyrus (anterior division)
- Feature 1400: correlation between Left Supramarginal Gyrus (posterior division) and Right Inferior Frontal Gyrus (pars triangularis)
- Feature 2811: correlation between Left Precuneous Cortex and Right Middle Temporal Gyrus (posterior division)

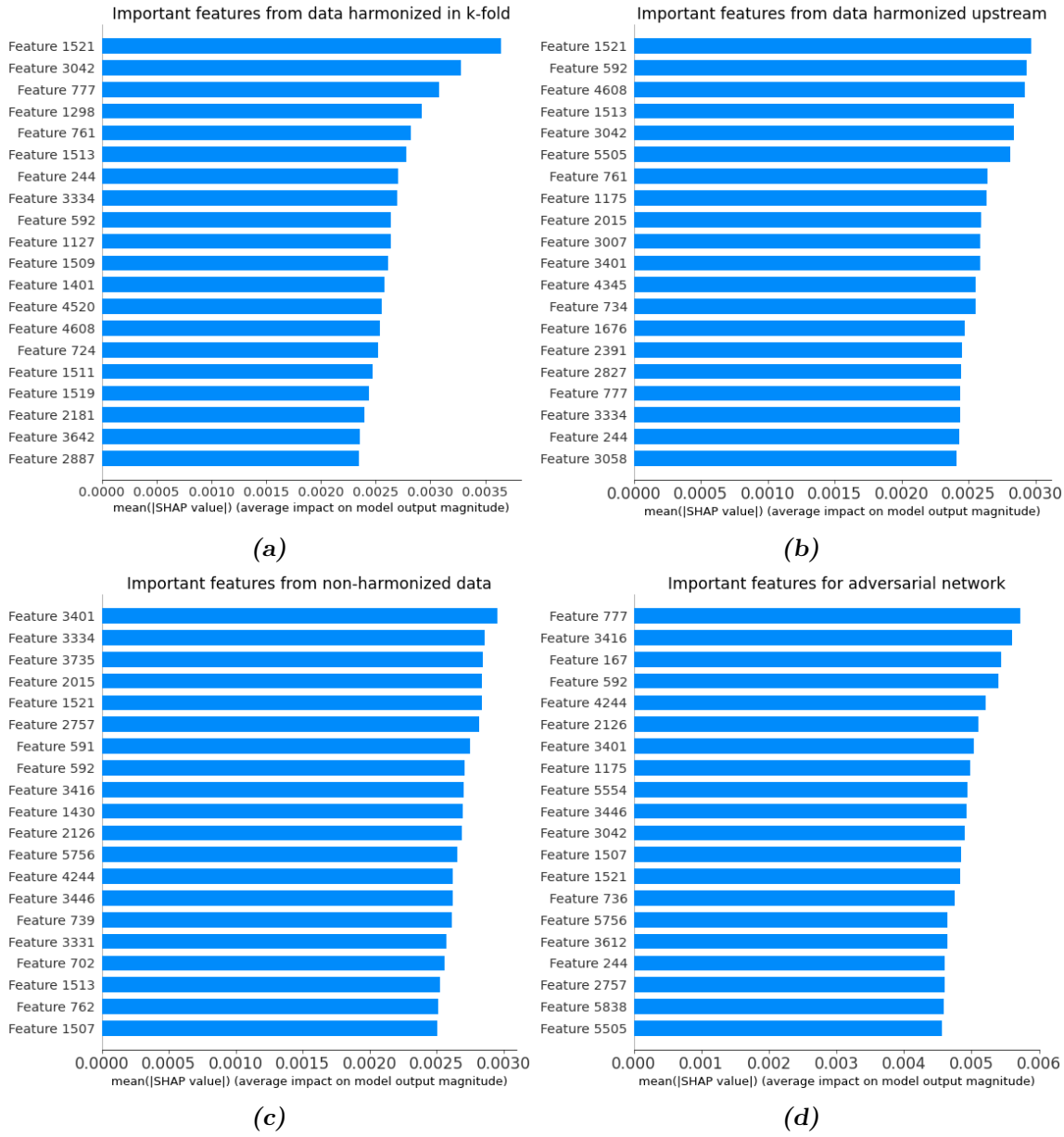


Figure 15.3: *First twenty most important features plotted with a bar plot of the mean absolute Shapley values. Extracted from each of the following four analysis pipelines using DeepSHAP: harmonization within k-fold CV + DNN (a); harmonization upstream + DNN (b); DNN on non-harmonized data (c); deep adversarial network (d).*

15.2.3 Comments on feature importance

Comparing the results obtained using different classifiers, a DNN and a Random Forest classifier, we notice from fig. 15.3 and 15.4 that there are no common features among the first 20 ones. This is most likely due to the inner differences in the algorithms that carry out the decisional process. A DNN and a Random Forest have very different modes of operation. This can lead them to find different patterns within data whose results do not (or just partially) overlap. We are prone to consider the DNN most reliable than the Random Forest because of the higher classification performance

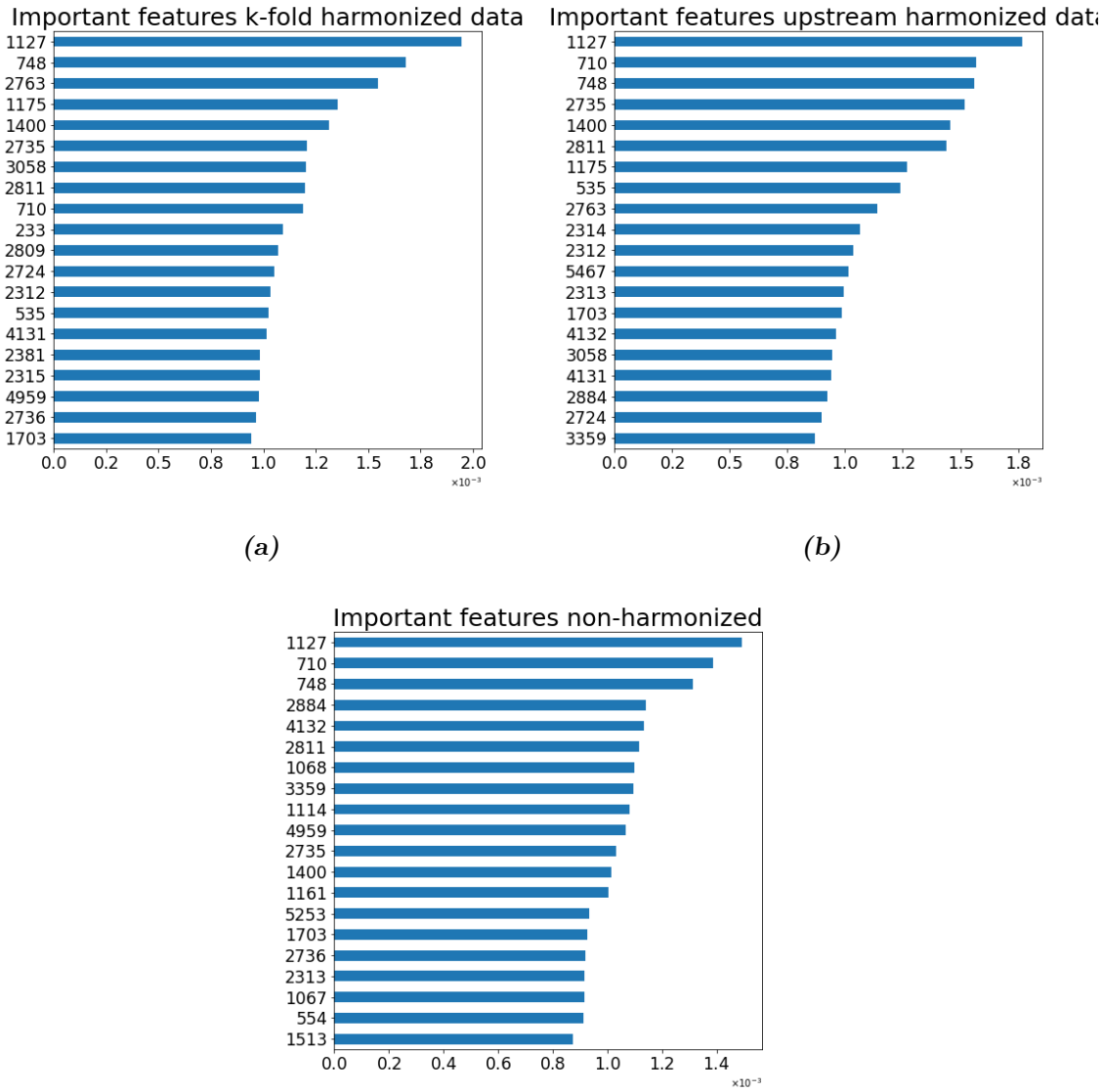


Figure 15.4: *Feature importance obtained with a Random Forest classifier trained on the entire (ABIDE I + ABIDE II) dataset where: the harmonization procedure was implemented inside* ***k-fold*** *CV (a); the harmonization procedure was implemented* ***upstream*** *with respect to the k-fold CV (b); the classification was implemented on non-harmonized data (c).*

achieved (i.e. 71 ± 1 vs. 66 ± 2 on the harmonized sample including ABIDE I + II, as reported in Table 14.2). However, as we can notice from the plots, Random Forest feature importance is able to identify as important some features whose importance follow an abrupt descending trend.

The same characteristic is not visible with features extracted from the DNNs except for the pipeline where the harmonization was run inside k-fold. In that case it seems that some features provide a greater contribution than others and the descending degree of feature importance is less smooth. A possible reason for this trend, in accordance of what we assumed during the discussion of classification results, is that the harmonization procedure, even if it does not bring up new information, helps in data cleaning, and removes some noise due to site-distinctive patterns. Harmonization thus helps to obtain a better and a more defined assessment of what are the most important features in the discrimination between subjects with ASD and controls.

Despite there are no overlapping features between the analysis with DNN and with Random Forest, if we take a look at the lists of the brain areas involved in these altered correlations, we can notice that certain brain areas, mainly related to the temporal lobe, are in common between these two results. This can suggest that even if the lists of involved features provided by the two models is not the same, brain areas more affected than others in the ASD conditions are consistently identified by both approaches. The following analysis aims to study this aspect and determine what are the most important brain areas that allow a discrimination between healthy and ASD subjects.

15.2.4 Consistency of important features across DNN and Random Forest classifiers

So far, we discussed results linked to images just to have a visual comparison of feature importance between DNN and Random Forest classifiers. In this section we systematically quantify the degree of overlap between the features identified in the different analysis pipelines Since, as explained in section 14.4, the pipeline which includes upstream harmonization is a misleading way to proceed as it creates a bias in the results, we exclude it from our analysis in what follows. In this section, when we refer to harmonized data, we are referring to data where the harmonization procedure has been implemented inside the k-fold CV.

To assess what are the consistent features among the different analysis pipelines, we choose to limit our analysis to the 1% of important features among the 5995 ones. To this end, we collected the first 60 most important features and we limited our analysis to the 5 classification procedures listed below:

1. Classification of harmonized data using the DNN
2. Classification of harmonized data using the Random Forest classifier
3. Classification of raw data using the DNN
4. Classification of raw data using the Random Forest classifier
5. Classification using raw data with the adversarial neural network

Among them, we specify that the pipelines 1, 2 and 5 can be considered fully unbiased, as we implemented all necessary analysis steps to remove site-related biases, whereas the pipelines 3 and 4, are not free in principle from site-related effects, thus could be considered as less reliable than the other three.

Firstly we checked if and how many common features are found among the 60 most important features among these five classification pipelines. We found out that there are no common features among all the pipelines, at least when we only take into account only the first 60 features.

Then, we carried out the analysis on subgroups of classification methods and we report below the number of consistent important features among the 60 most relevant ones as a number and as a percentage. This kind of analysis is useful for example to assess whether the harmonization procedure significantly modifies the list of the important features for a same classifier. It is also aimed to quantify the common features between a DNN and a Random Forest classifier.

The results of this analysis can be summarized as follows:

- Focusing on the choice of a **DNN**, we compared the pipelines 1 and 3 finding out that 22 features (37 %) out of 60 are consistently identified as important in the analysis with DNN of harmonized and raw data.
- Focusing on the choice of a **Random Forest** classifier, we compared the pipelines 2 and 4 finding out that 33 features (55%) out of 60 are consistently identified as important in the analysis of harmonized and raw data with a Random Forest classifier.
- Focusing on the choice of analyzing **harmonized data**, we compared the features defined as important by a **DNN** (pipeline 1) with those highlighted by a **Random Forest** classifier (pipeline 2), and we found out 13 consistent features (22 %) out of 60
- Focusing on the choice of analyzing **raw data**, we compared the features defined as important by a **DNN** (pipeline 3) with those highlighted by a **Random Forest** classifier (pipeline 4), and we found out 7 consistent features (12 %) out of 60
- Comparing the list of the important features generated by the **Adversarial network** (pipeline 5) with that generated by the DNN on **harmonized data** (pipeline 1), we identified 21 consistent features (35%) out of 60
- Comparing the list of the important features generated by the **Adversarial network** (pipeline 5) with that generated by with DNN on **raw data** (pipeline 3), we identified 28 consistent features (47%) out of 60

15.2.5 Discussion on consistent important features across the different analysis pipelines

By checking the common features between different machine learning model and harmonization procedure, we can have a better comparison on how much impact these factor have on the assessment of feature importance.

Looking at the results listed in section 15.2.4 we can assert that DNN and Random Forest definitely have different ways to use features to make prediction and to assess their importance. We find that just a low percentage (12% and 22%) of features are common between a DNN and a Random Forest for both raw and harmonized data. This has a non-negligible impact when choosing the best classificator for any classification task. An important aspect to keep in mind is that there is not "the best" classifier in absolute, but each classifier is able to find different patterns among data. So the best classifier depends on the specific dataset and the classification task.

We can also state that harmonization procedure changes, especially in a DNN, the most important features. As we pointed out when discussing the plots of feature importance, harmonization is able to remove noise from data and reveals in a cleaner way what features are really important for the network, without the contribution of site-related noise. It is possible that the number of common feature in a Random Forest between harmonized and non harmonized data is bigger because Random Forest is a simpler algorithm and it is no able to find complex relations between data. In

fact a DNN is able to learn more complex pattern, and in doing so, it is more sensitive to noise and to the discovery of more subtle pattern that can drive to this difference in feature ranking. This finer searching for patterns, though, is what lead a DNN to achieve better classification performances with respect to a Random Forest. Thus, harmonization changes the most important features, but making it more reliable, since features are less affected by noise and results are more meaningful.

This change, resulting from a better definition of feature due to harmonization, also appears when comparing results with the adversarial network. In fact we observe that the number of common feature between adversarial network and raw data is similar to the number between harmonized and raw data determined using the DNN. The reason for this lies in the mechanism of the adversarial network: it results in a reduction of some site-related noise thanks to the adversarial branch, but at the same time, the introduction of some confusion due to the flawed definition of the correct site.

Between raw data and adversarial network a similarity of feature importance $\approx 47\%$ that is greater than what we obtained with raw data and harmonized data $\approx 37\%$. This means that with the adversarial network data are modified and harmonized according to some inner processes, which occur in a different way than the analytical harmonization.

15.3 Deriving the involvement of brain areas from the lists of the important features

As mentioned before, each feature is representative of a correlation between two brain areas, and, since for brain parcellation we used the Harvard-Oxford atlas with 110 ROIs, each brain area is involved in 109 features. In this section we investigate what are the most recurrent brain areas which have a role in the discrimination between subjects with ASD and healthy controls.

For this purpose, we plot a histogram to represent the number of occurrences of each brain area among the first 60 features for each classification procedure. Subsequently, we create the histogram of the overall important brain areas pooling all these results into a single histogram.

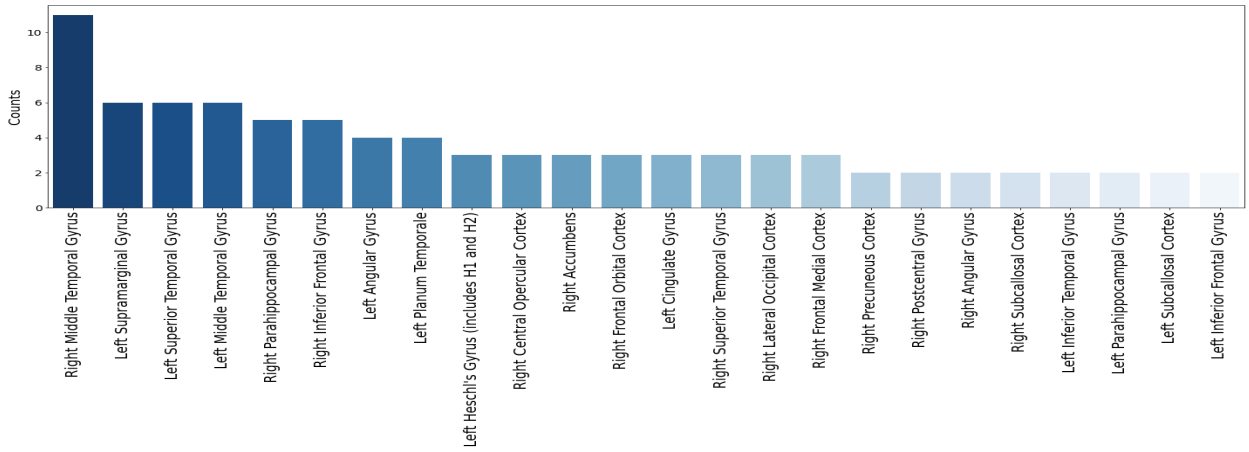
Figure 15.5 shows for each pipeline, what are the brain areas recurrent among the 60 most important features. In fig. 15.6 the overall histogram comprising all the results is shown.

It is immediate to notice that there are recurrent brain areas across all the different analysis pipelines. In fig. 15.6 the overall histogram of the most recurrent brain areas between these common features is reported.

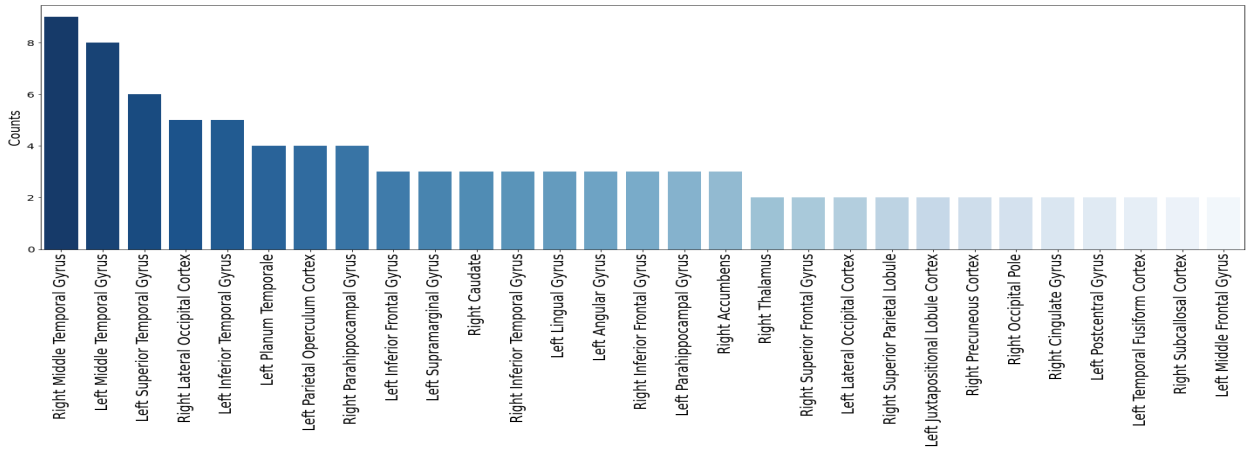
15.3.1 Discussion on consistently involved brain areas across the different analysis pipelines

From the analysis of important brain areas in terms of their recurrence within altered functional connections, we can identify in fig. 15.6 a list of ≈ 30 among 110 areas that contributed the most in discrimination between controls and subjects with ASD. We compared the brain areas extracted from these analysis with the brain areas obtained in the study on structural brain alterations by Saponaro *et al.* [67]. In that study, the authors identified with a Random Forest classifier the most discriminating features extracted from structural images of patients from the same ABIDE dataset. They used the atlas Desikan-Killiany-Tourville implemented in Freesurfer [3] to parcel the brain. It consists in 62 total ROIs. This number differs from the number of ROIs implemented in the Harvard-Oxford atlas implemented here, which is equal to 110. For this reason for some features, we can compare the two studies only identifying the main lobes involved in them both. In the following lines, we present the brain areas found in common between our analysis and Saponaro's

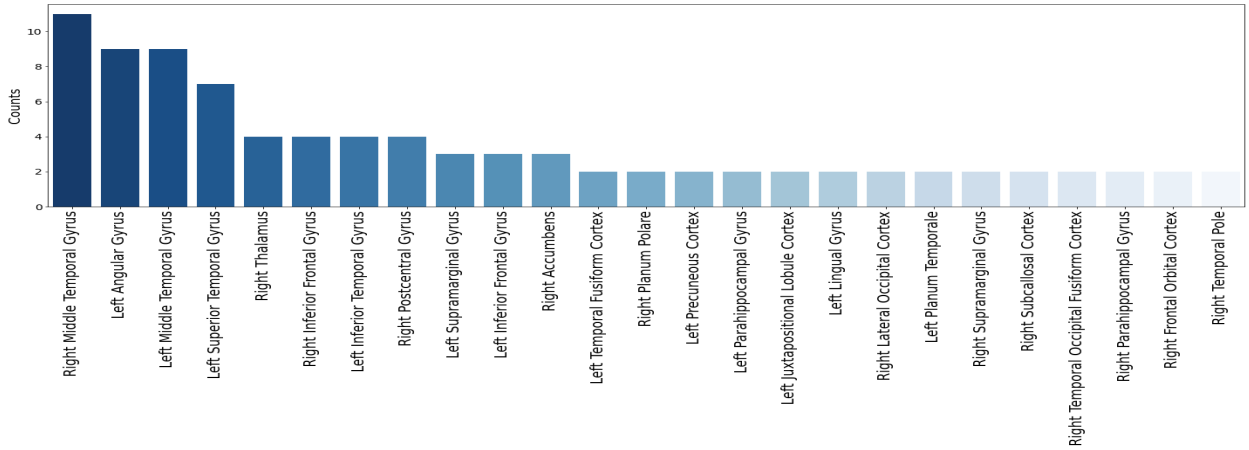
[3] https://surfer.nmr.mgh.harvard.edu/



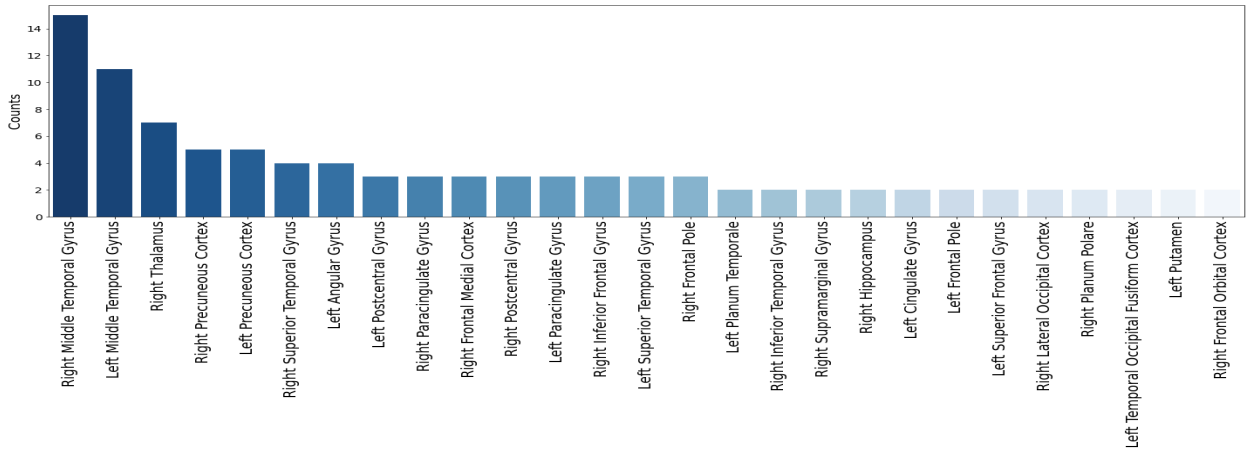
(a) *Important areas from raw data classified with DNN*



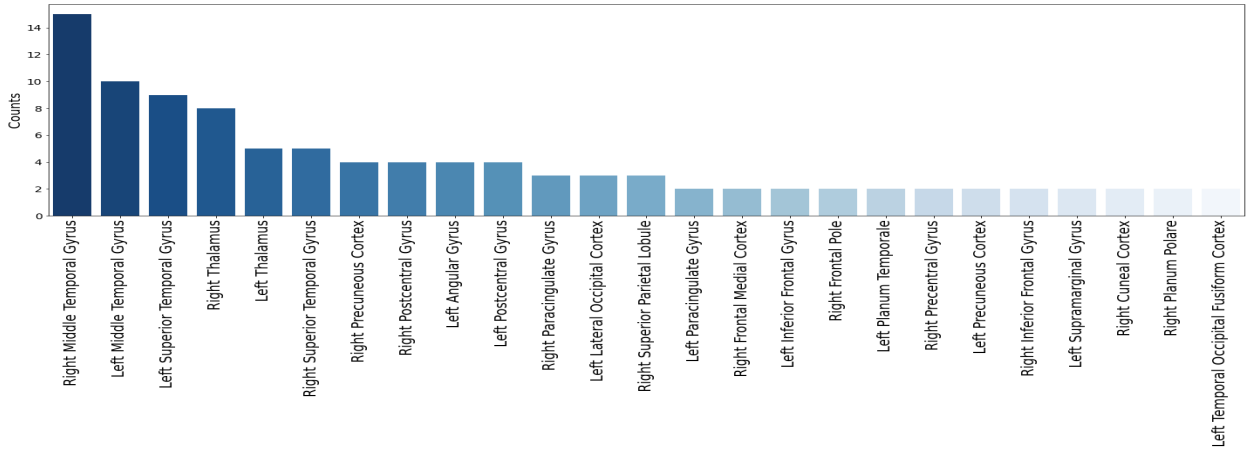
(b) *Important areas from raw data classified with Adversarial network*



(c) *Important areas from harmonized data classified with DNN*



(d) *Important areas from harmonized data classified with Random Forest*



(e) *Important areas from Raw data classified with Random Forest*

Figure 15.5: *Histograms of important brain areas extracted from the first 60 important features identified by the five different classification procedures*

paper. We also provide a brief explanation about the role of different brain regions and how their alterations could be responsible of some traits related to the ASD condition. The purpose of this brief investigation is not to achieve a medical accurate analysis and interpretation of the fully data driven results achieved in this Thesis, but only intends to provide some glimpses of the function of brain regions, whose alterations have been consistently found in the analysis of structural and the functional data of the same dataset.

The areas concerning the right superior temporal, right middle temporal and right inferior temporal gyri are among the most interesting regions. They belong to the temporal lobe, which is involved in processes like language comprehension and emotion recognition. In particular, the right superior temporal gyrus has been identified as a crucial area for social cognition.

Left angular and left supermarginal gyri belong to the inferior parietal lobule. This lobule is generally involved in perception of emotions and interpretation of sensory information. However, these two structures are particularly involved in language processes and mathematical operations. Regions like the right orbito-frontal and the right accumbens are both involved in rewards related to decision making. The right nucleus accumbens also concerns themes like motivation, aversion and the feeling of pleasure subsequent to a reward. The left lingual gyrus is related to the processing of visual letters, and it plays and important role in the analysis of logical conditions as well.

Alterations in the precuneous cortex region are typically related to atypical socio-cognitive functions. This region is involved in cognitive processes such as empathy and self-awareness and increased dimensions of it, have been identified both on male and female subjects with ASD [68].

Other regions such as the thalamus, which is associated to processes like the regulation of consciousness and the processing of sensory signals, are involved in so many processes that reducing them to a singular function linked to ASD would be too simplistic.

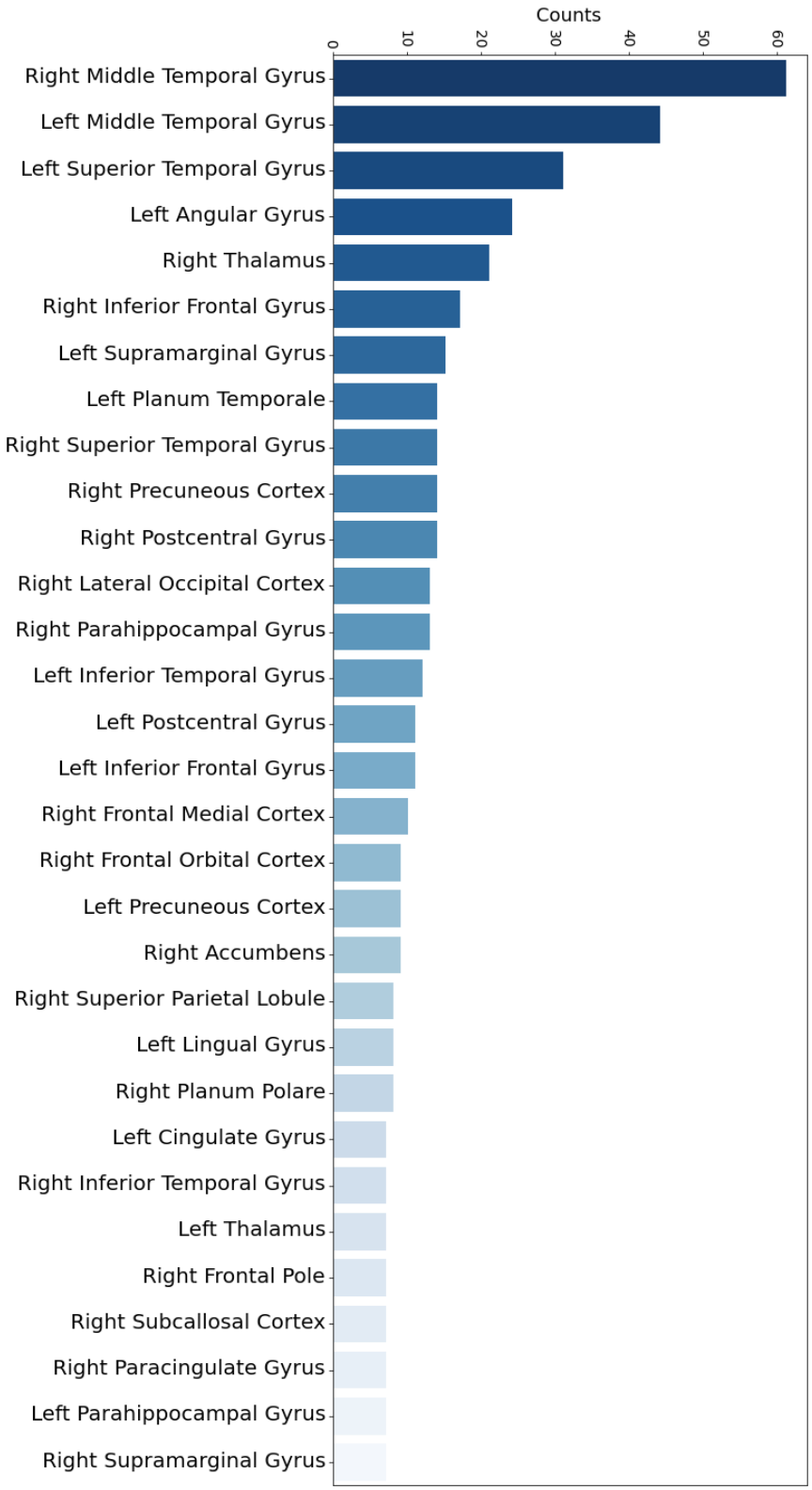


Figure 15.6: *Histogram of the most important brain areas identified by merging the results obtained from all the five analysis pipelines listed above.*

Conclusions

This Thesis is focused on the study of functional brain connectivity using machine learning and deep learning techniques. We applied our analysis to the study of Autism Spectrum Disorder, which is an extremely heterogeneous neurodevelopmental disorder with a strong social impact on the quality of life of affected individuals and of their families. For the diagnosis and characterization of ASD no univocal instrumental biomarkers have been identified so far. Discordant results have been reported due to the broad range of manifestations of ASD.

In this work we deepened the investigation of ASD by analyzing the resting-state functional MRI data of subjects with ASD and control subjects provided by the international ABIDE initiative. Brain functional connectivity alterations in ASD have already been reported. However, the choice of the components of the data processing pipeline and of the set of parameters in each analysis module can have a strong impact on the identification of possible differences between subjects with ASD and healthy controls.

The data analysis pipeline we followed to extract a measure of functional connectivity from rs-fMRI raw data starts with the preprocessing of the raw data with C-PAC, which allowed to assemble a robust and reproducible processing pipeline to extract the signal timeseries from each brain area of interest, for each patient. To investigate the functional connectivity we compared two different approaches, aiming at finding the best strategy to identify possible differences between healthy controls and patients with ASD. One is based on Pearson correlation between the timeseries of different areas of the brain, while the other employs a time-frequency analysis using the wavelet transforms. Using machine learning techniques, Pearson-based correlation coefficients allowed to achieve a better case-control separation.

We covered in this Thesis also a currently much debated topic which is related to the harmonization of multicentric data. Raw fMRI data were provided by the ABIDE initiative, consisting of a collection of scans acquired in different medical centers. We compared two different procedures with the intent of removing site-related patterns from data, which can bias the machine learning training process and thus the results. One consists of an analytical data harmonization approach, already implemented in the publicly available NeuroHarmonize package. Following this approach, we also assessed that the best way to implement it during a classification task with a machine learning model is within a cross validation procedure. In this way it is possible to avoid data leakage and overestimation of the classification performances. The second harmonization strategy employs an originally developed deep neural network specifically designed to perform an adversarial learning and remove site-related biases. It allows to perform a classification of controls vs ASD data that does not comprise any biases toward sites. The implementation of a DANN to remove the site effect in multicenter MRI datasets is a novel approach. So far, no previous studies concerning the ABIDE dataset were carried out using this approach to data harmonization. The two different implementation of multicenter data harmonization allowed to obtain similar discrimination performances.

Despite the ABIDE initiative includes structural and functional data of more than 2200 subjects,

the FC connectivity measures we intend to analyze with machine learning tools is extremely high ($\sim$ 6000). Thus, the number of investigated features exceeds the number of total subjects. This leads to an inconvenient setup when working with machine learning models since it may bring to overfitting conditions. We have tried to deal with this problem in different parts of this work.

In the future, it is likely that with the collection of new data and the creation of a bigger dataset, studies on functional connectivity using machine learning methods may reach greater classification performances and lead towards the development of a accurate investigation tool to identify reliable biomarkers of the ASD condition. Moreover, the vast majority of studies about ASD carried out so far with the ABIDE dataset, only focus on male subjects. This is a consequence of the greater number of males with respect to female subjects with ASD. For this reason, increasing the dataset size would represent an important progress to allow meaningful analysis on female subjects as well.

As a further topic covered in this Thesis, a specific attention has been devoted to the implementation of explainability methods for machine learning models. When working with machine learning algorithms, especially when dealing with clinical data, it is of great importance to assess what features are decisive for the prediction of a certain condition (e.g. the presence of a disease the outcome of a treatment, etc). To this end, methods of explainable AI were implemented using the SHAP package. The algorithm we used, DeepSHAP, allowed to rank by importance all the features a deep learning model analyzes. By applying this explainability method to functional connectivity data it is possible to identify the alterations which are responsible for the discrimination between healthy subjects and patients with ASD.

Both the procedures, i.e. the deep learning classification of analytical harmonized data and the deep adversarial learning, lead to a common set of important features. This similarity became even more evident when we converted functional connectivity measures into the brain areas mainly involved. We identified a set of brain areas consistently altered in subjects with ASD, and compared them with a recent study carried out on the structural MRI data of the same population. We observed that a great number of areas that we found as relevant in our discrimination problem based on brain functional data were actually identified as important also in the analysis of structural MRI data and are related to ASD traits.

Although this work is focused on the study of ASD, the analysis we carried out is general and can potentially be applied to a number of different disorders and disease conditions where an altered functional connectivity is suspected.

Appendix

Appendix A1

Classification results with different neural networks

In table A1 .1 we report the AUC scores obtained with different configurations of neurons in a deep neural network, in order to find the most suitable for our classification tasks. We chose to employ a network with a configuration 264-8-1, even if with a structure 128-8-1 we obtained similar results. We selected this network a little more complex than the necessary one, in order to insert some regularizer layers such as dropout.

Structure	k-fold	upstream	no harmonization
3-2-1	60±2	63±3	60±4
8-8-1	64±1	65±4	64±5
64-8-1	69±1	72±2	68±1
64-32-8-1	68±1	72±2	69±1
128-8-1	70±2	71±3	69±2
128-64-1	70±1	72±3	69±2
264-8-1	70±2	73±3	70±2
512-8-1	71±2	73±2	70±2
1024-8-1	70±1	74±2	71±2
1024-32-1	70±2	74±2	71±3

Table A1 .1: *AUC score obtained with different model structures for the three main harmonization pipelines we carried out during this entire work. Colored green is the structure we choose to employ for all our analysis. The structure column reports the number of neurons in each layer, separated by a dash.*

Appendix A2

Feature importance results on ABIDE I open eyes dataset

The same tests we did to assess feature importance on ABIDE I + II dataset, was carried out on ABIDE I with only patient with open eye. In figure A2 .1 we report the results obtained from SHAP, related to the first twenty most relevant features. We can notice at a first glance that there are less feature common to all the four analysis: There are no common features among them that we can find across all these four analysis, if we limit our study on the first twenty. We have to search among the first thirty features to find something in common, which is still a good procedure since we are dealing with 5995 features and the first 30 are just the 0.5% of them.

We find that, among the first 30 features, only feature 762 is common to the four analysis, while if we exclude the adversarial network we add feature 1417 and 748

- Feature 762: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Left Middle Frontal Gyrus
- Feature 1417: correlation between Left Supramarginal Gyrus (posterior division) and Left Middle Temporal Gyrus (temporooccipital part)
- Feature 748: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Right Thalamus

With a Random Forest we obtain more coherence between important features: searching among the first 20 features, we find that are common to the 3 pipelines: 2690, 1127, 2313, 2314, 748, 2712, 2735, 2582, 1400

- Feature 2690: correlation between Left Cingulate Gyrus (posterior division) and Right Frontal Medial Cortex
- Feature 1127: correlation between Left Postcentral Gyrus and Right Postcentral Gyrus
- Feature 2313: correlation between Right Paracingulate Gyrus and Left Middle Temporal Gyrus (anterior division)
- Feature 2314: correlation between Right Paracingulate Gyrus and Right Middle Temporal Gyrus (posterior division)
- Feature 748: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Right Thalamus

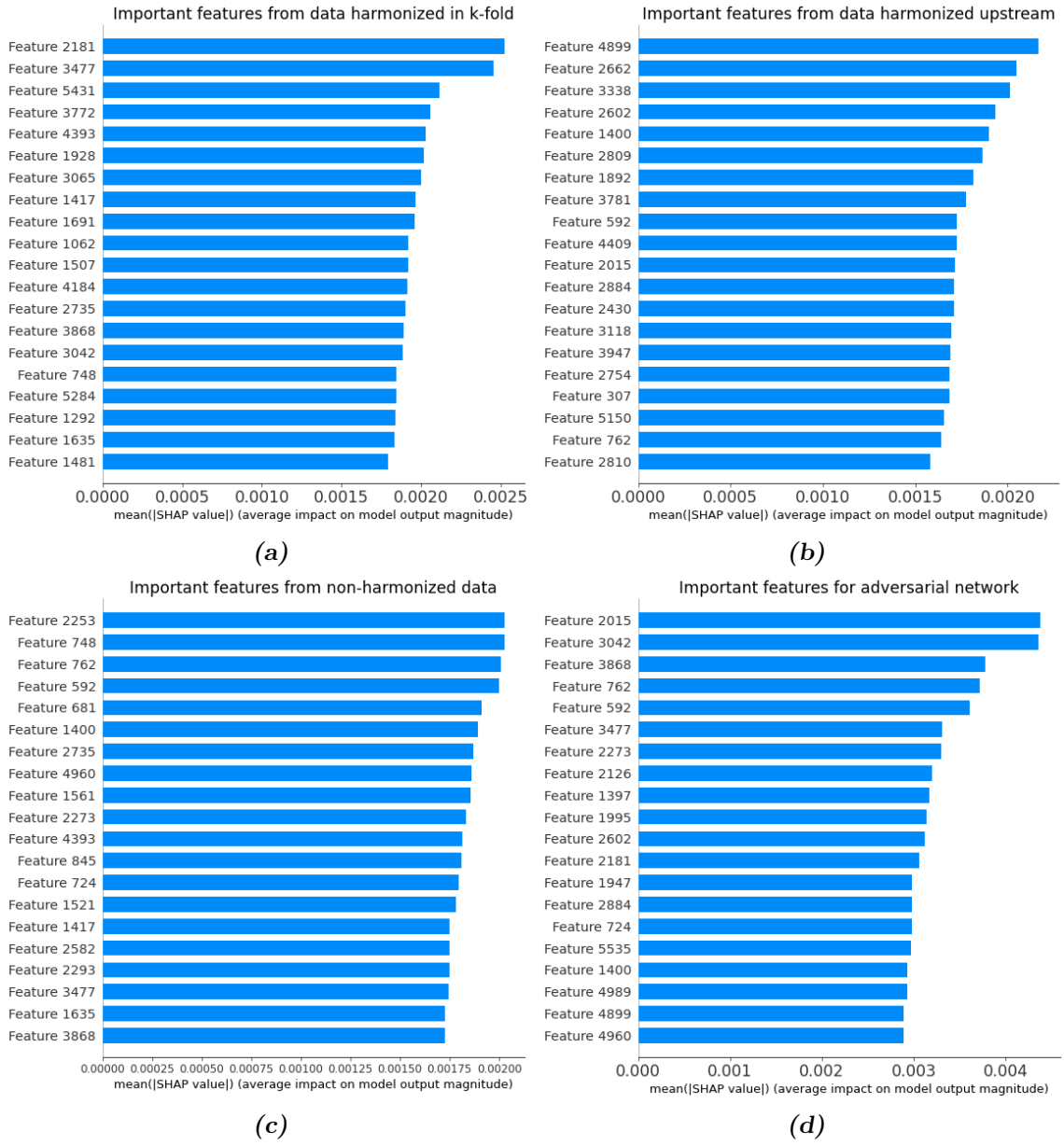


Figure A2 .1: *First twenty most important features plotted with a bar plot of the mean absolute Shapley values. Extracted from the ABIDE I only open eyes dataset, from each of the following four analysis pipelines using DeepSHAP: harmonization within k-fold CV + DNN (a); harmonization upstream + DNN (b); DNN on non-harmonized data (c); deep adversarial network (d).*

- Feature 2712: correlation between Right Precuneous Cortex and Right Hippocampus
- Feature 2735: correlation between Right Precuneous Cortex and Right Middle Temporal Gyrus (anterior division)
- Feature 2582: correlation between Right Cingulate Gyrus (posterior division) and Right Precentral Gyrus
- Feature 1400: correlation between Left Supramarginal Gyrus (posterior division) and Right

Inferior Frontal Gyrus (pars triangularis)

The same cross analysis we did on the entire dataset:

- Focusing on Random Forest classifier, we compared important features between 2 and 4 and found that 34 features out of 60 are common (57%).
- Focusing on DNN we compared pipelines 1 and 3 finding that 19 features (32 %) out of 60 are common.
- Comparing the same analysis pipeline with DNN and Random Forest we find that with harmonized data: procedure 2 and 1 there are 11 common features (18 %)
- Comparing results on raw data obtained with DNN 3 and Random Forest 4 we obtain 13 common features (22%)
- Comparing the adversarial results 5 with the harmonized data with DNN 1 we obtain 19 common features (32%)
- Comparing the adversarial results 5 with DNN classification of raw data 3 we obtain 23 common features (38%)

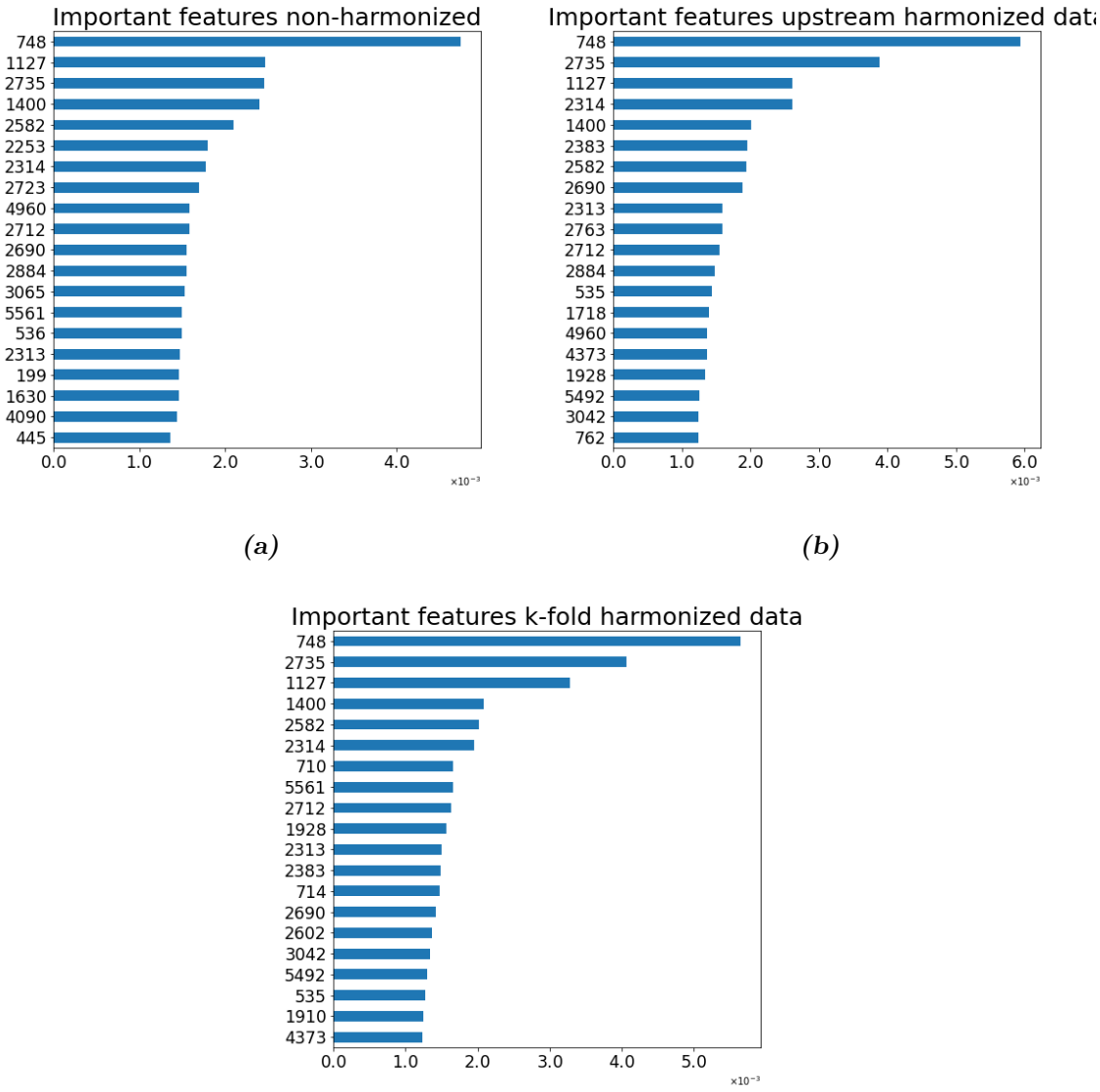
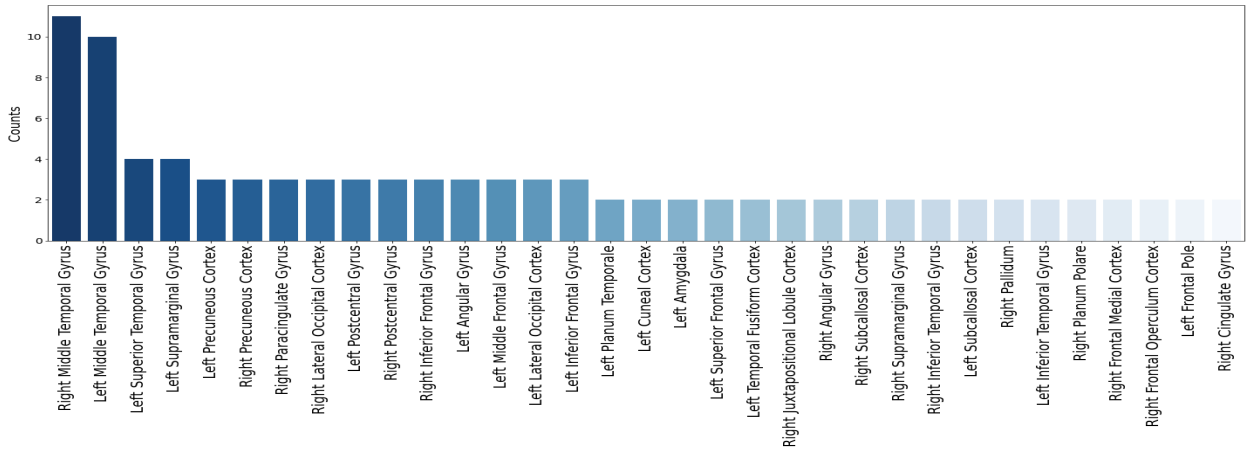
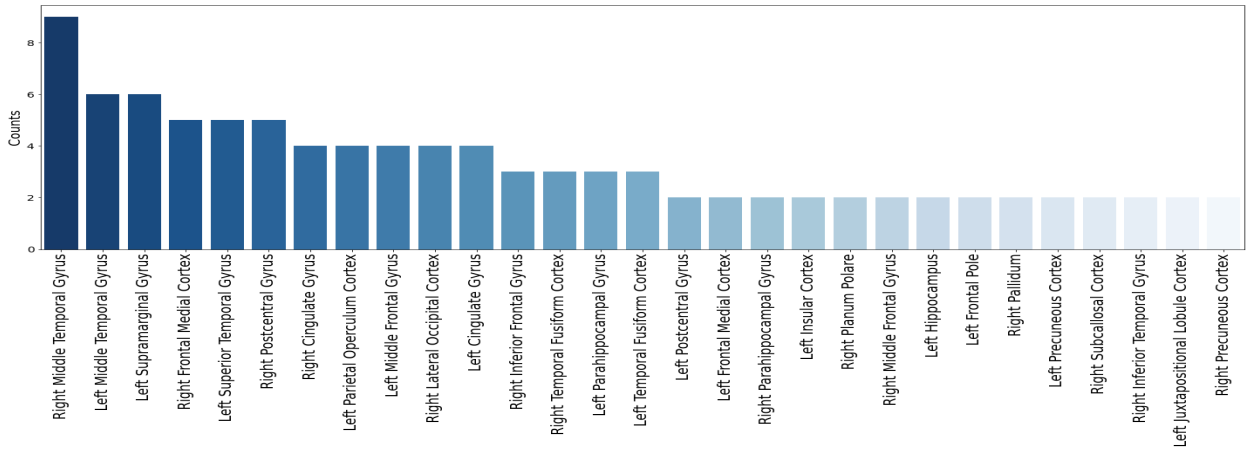


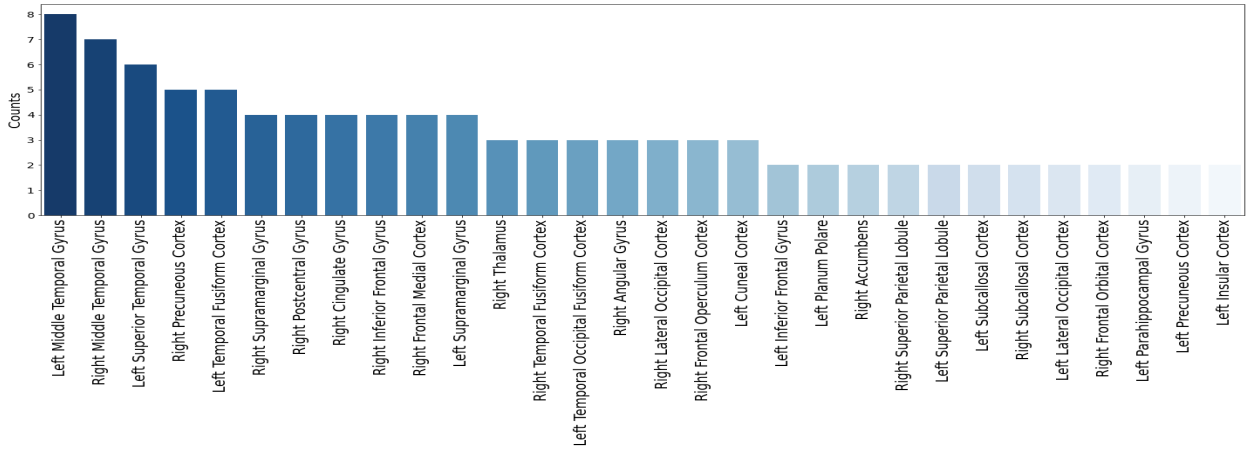
Figure A2 .2: *Feature importance obtained with a Random Forest classifier trained on the ABIDE I only open eyes dataset where: the harmonization procedure was implemented inside* ***k-fold*** *CV (a); the harmonization procedure was implemented* ***upstream*** *with respect to the k-fold CV (b); the classification was implemented on non-harmonized data.*



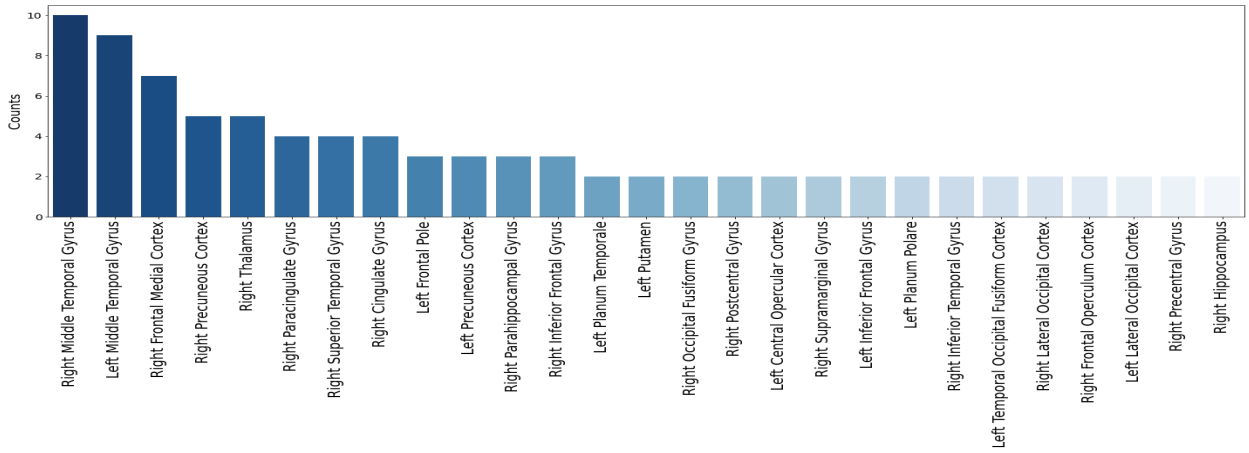
(a) *Important areas from Raw data classified with DNN*



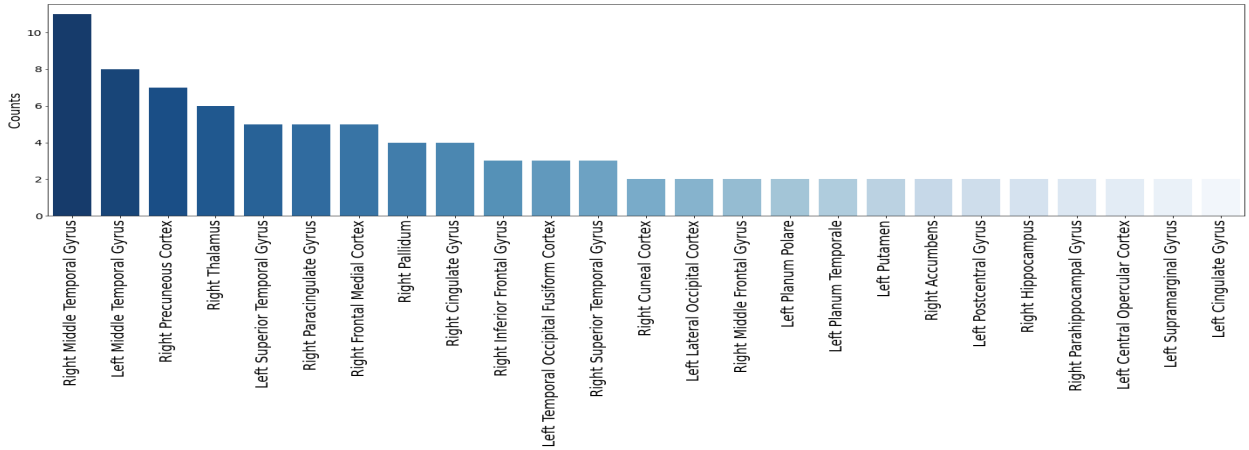
(b) *Important areas from Raw data classified with Adversarial network*



(c) *Important areas from harmonized data classified with DNN*



(d) *Important areas from harmonized data classified with Random Forest*



(e) *Important areas from Raw data classified with Random Forest*

Figure A2 .3: *Histograms of important brain areas extracted from the first 60 important features of different classification procedures*

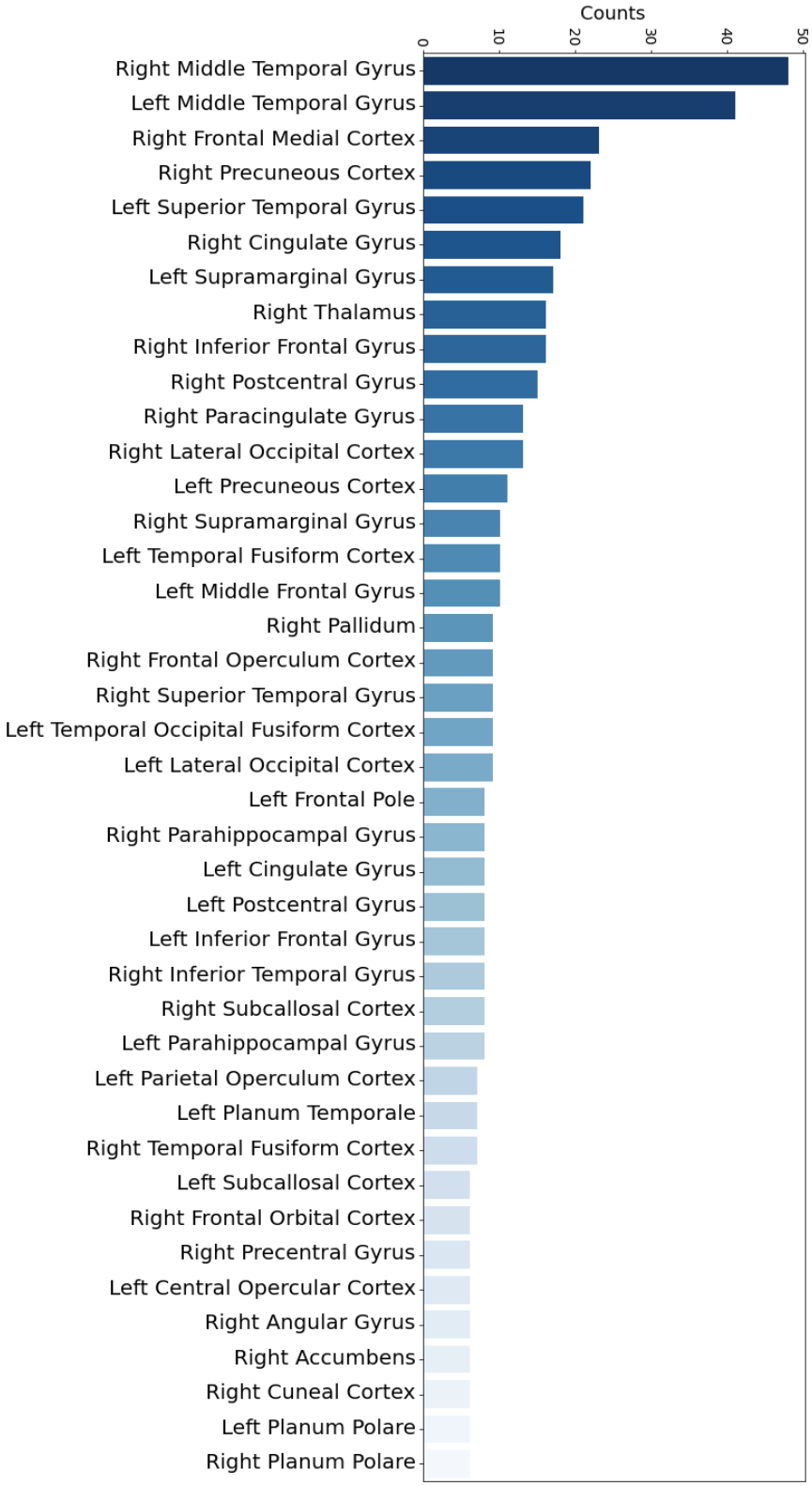


Figure A2 .4: *Histogram of the most important brain areas obtained by merging the results obtained from all the five analysis listed above.*

Bibliography

[1] Matthew J. Maenner et al. "Prevalence and Characteristics of Autism Spectrum Disorder Among Children Aged 8 Years — Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2018". *MMWR. Surveillance Summaries* 70 (11 Dec. 2021), pp. 1–16. DOI: http://dx.doi.org/10.15585/mmwr.ss7011a1.

[2] Samuel B. Guze. "Diagnostic and Statistical Manual of Mental Disorders, 4th ed." *American Journal of Psychiatry* 152 (8 Aug. 1995), pp. 1228–1228. DOI: 10.1176/ajp.152.8.1228.

[3] World Health Organization. "The ICD-10 classification of mental and behavioural disorders : diagnostic criteria for research" (1993), 248 p.

[4] Isabelle Rapin and Roberto F. Tuchman. "Autism: Definition, Neurobiology, Screening, Diagnosis". *Pediatric Clinics of North America* 55 (5 Oct. 2008), pp. 1129–1146. DOI: 10.1016/j.pcl.2008.07.005.

[5] Diana L Robins, Deborah Fein, and Marianne Barton. *Modified Checklist for Autism in Toddlers, Revised, with Follow-Up (M-CHAT-R/F) TM*. 2009.

[6] Sally Ozonoff, Beth L. Goodlin-Jones, and Marjorie Solomon. "Evidence-based assessment of autism spectrum disorders in children and adolescents". *Journal of Clinical Child and Adolescent Psychology* 34 (3 2005), pp. 523–540. DOI: 10.1207/s15374424jccp3403_8.

[7] Ann Le Couteur et al. "Diagnosing Autism Spectrum Disorders in Pre-school Children Using Two Standardised Assessment Instruments: The ADI-R and the ADOS". *Journal of Autism and Developmental Disorders* 38 (2 Feb. 2008), pp. 362–372. DOI: 10.1007/s10803-007-0403-3.

[8] C. M. Freitag. "The genetics of autistic disorders and its clinical relevance: A review of the literature". *Molecular Psychiatry* 12 (1 Jan. 2007), pp. 2–22. DOI: 10.1038/sj.mp.4001896.

[9] Merel C. Postema et al. "Altered structural brain asymmetry in autism spectrum disorder in a study of 54 datasets". *Nature Communications* 10 (1 Dec. 2019), p. 4958. DOI: 10.1038/s41467-019-13005-8.

[10] Kaitlin Riddle, Carissa J. Cascio, and Neil D. Woodward. "Brain structure in autism: a voxel-based morphometry analysis of the Autism Brain Imaging Database Exchange (ABIDE)". *Brain Imaging and Behavior* 11 (2 Apr. 2017), pp. 541–551. DOI: 10.1007/s11682-016-9534-5.

[11] Kaustubh Supekar et al. "Brain Hyperconnectivity in Children with Autism and its Links to Social Deficits". *Cell Reports* 5 (3 Nov. 2013), pp. 738–747. DOI: 10.1016/j.celrep.2013.10.001.

[12] Giovanna Spera et al. "Evaluation of Altered Functional Connections in Male Children With Autism Spectrum Disorders on Multiple-Site Data Optimized With Machine Learning". *Frontiers in Psychiatry* 10 (Sept. 2019). DOI: 10.3389/fpsyt.2019.00620.

[13] Chao Zhang et al. "Sex and Age Effects of Functional Connectivity in Early Adulthood". *Brain Connectivity* 6 (9 Nov. 2016), pp. 700–713. DOI: 10.1089/brain.2016.0429.

[14] C. Edward Coffey et al. “Sex Differences in Brain Aging”. *Archives of Neurology* 55 (2 Feb. 1998), p. 169. DOI: 10.1001/archneur.55.2.169.

[15] Víctor Costumero et al. “Opening or closing eyes at rest modulates the functional connectivity of V1 with default and salience networks”. *Scientific Reports* 10 (1 Dec. 2020), p. 9137. DOI: 10.1038/s41598-020-66100-y.

[16] Robert W. Brown et al. *Magnetic Resonance Imaging*. John Wiley and Sons Ltd, Apr. 2014. DOI: 10.1002/9781118633953.

[17] Scott A. Huettel, Allen W. Song, and Gregory McCarthy. *Functional Magnetic Resonance Imaging*. OUP Higher Education Division, 2009.

[18] Mark Jenkinson and Michael Chappell. *Introduction to Neuroimaging Analysis*. Oxford University Press, 2018.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Ed. by MIT Press. MIT Press, 2016.

[20] Pankaj Mehta et al. “A high-bias, low-variance introduction to Machine Learning for physicists”. *Physics Reports* 810 (May 2019), pp. 1–124. DOI: 10.1016/j.physrep.2019.03.001.

[21] Vahid Mirjalili Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 2019.

[22] Sarah Guido Andreas C. Müller. *Introduction to Machine Learning with Python*. O’Reilly Media, Inc., 2017.

[23] F Pedregosa et al. “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[24] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift” (Feb. 2015). DOI: 10.48550/arXiv.1502.03167.

[25] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization” (2014). DOI: 10.48550/arXiv.1412.6980.

[26] Don Ross. *Game Theory*. Ed. by Edward N Zalta. 2021. URL: https://plato.stanford.edu/archives/fall2021/entries/game-theory/.

[27] John F. Nash. “Equilibrium points in n-person games”. *Proceedings of the National Academy of Sciences* 36 (1 Jan. 1950), pp. 48–49. DOI: 10.1073/pnas.36.1.48.

[28] Shapley Lloyd S. *Notes on the N-Person Game I: Characteristic-Point Solutions of the Four-Person Game*. RAND Corporation, 1951. DOI: 10.7249/RM0656.

[29] L. S. Shapley. *17. A Value for n-Person Games*. Dec. 1953. DOI: 10.1515/9781400881970-018.

[30] Encarnación Algaba, Vito Fragnelli, and Joaquín Sánchez-Soriano. *Handbook of the Shapley Value*. Chapman and Hall/CRC, 2021.

[31] Eric Fombonne. “Epidemiology of Pervasive Developmental Disorders”. *Pediatric Research* 65 (6 June 2009), pp. 591–598. DOI: 10.1203/PDR.0b013e31819e7203.

[32] Janine Bijsterbosch, Stephen Smith, and Christian Beckmann. *Introduction to Resting State fMRI Functional Connectivity Oxford Neuroimaging Primers*. Oxford University Press, 2017.

[33] Matthew Brett, Ingrid S. Johnsrude, and Adrian M. Owen. “The problem of functional localization in the human brain”. *Nature Reviews Neuroscience* 3 (3 Mar. 2002), pp. 243–249. DOI: https://doi.org/10.1038/nrn756.

[34] Nikos Makris et al. “Decreased volume of left and total anterior insular lobule in schizophrenia”. *Schizophrenia Research* 83 (2-3 Apr. 2006), pp. 155–171. DOI: 10.1016/j.schres.2005.11.020.

[35] N. Tzourio-Mazoyer et al. “Automated Anatomical Labeling of Activations in SPM Using a Macroscopic Anatomical Parcellation of the MNI MRI Single-Subject Brain”. *NeuroImage* 15 (1 Jan. 2002), pp. 273–289. DOI: 10.1006/nimg.2001.0978.

[36] R. Cameron Craddock et al. “A whole brain fMRI atlas generated via spatially constrained spectral clustering”. *Human Brain Mapping* 33 (8 Aug. 2012), pp. 1914–1928. DOI: 10.1002/hbm.21333.

[37] Rahul S. Desikan et al. “An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest”. *NeuroImage* 31 (3 July 2006), pp. 968–980. DOI: 10.1016/j.neuroimage.2006.01.021.

[38] Craddock Cameron et al. “The Neuro Bureau Preprocessing Initiative: open sharing of preprocessed neuroimaging data and derivatives”. *Frontiers in Neuroinformatics* 7 (2013). DOI: 10.3389/conf.fninf.2013.09.00041.

[39] Xin Yang, Paul T., and Ning Zhang. “A Deep Neural Network Study of the ABIDE Repository on Autism Spectrum Classification”. *International Journal of Advanced Computer Science and Applications* 11 (4 2020). DOI: 10.14569/IJACSA.2020.0110401.

[40] Luca Baldini et al. *Introduzione all’analisi dei dati per il laboratorio di Fisica*. URL: https://bitbucket.org/lbaldini/statnotes.

[41] Rick Wicklin. *Fisher’s transformation of the correlation coefficient*. 2017. URL: https://blogs.sas.com/content/iml/2017/09/20/fishers-transformation-correlation.html.

[42] Isidoro Ferrante. *Elaborazione dei segnali per la fisica*. Pisa University Press, 2015.

[43] J.C. van den Berg. *Wavelets in Physics*. Ed. by J. C. van den Berg. Cambridge University Press, Aug. 1999. DOI: 10.1017/CBO9780511613265.

[44] Donald B. Percival and Andrew T. Walden. *Wavelet Methods for Time SeriesAnalysis*. Cambridge University Press, 2000. DOI: 10.1017/CBO9780511841040.

[45] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Jan. 1992. DOI: 10.1137/1.9781611970104.

[46] Karsten Müller et al. “Investigating the wavelet coherence phase of the BOLD signal”. *Journal of Magnetic Resonance Imaging* 20 (1 July 2004), pp. 145–152. DOI: 10.1002/jmri.20064.

[47] A. Grinsted, J. C. Moore, and S. Jevrejeva. “Application of the cross wavelet transform and wavelet coherence to geophysical time series”. *Nonlinear Processes in Geophysics* 11 (5/6 Nov. 2004), pp. 561–566. DOI: 10.5194/npg-11-561-2004.

[48] Paul C. Liu. *Wavelet Spectrum Analysis and Ocean Wind Waves*. Springer Berlin, Heidelberg, 1994, pp. 151–166. DOI: 10.1016/B978-0-08-052087-2.50012-8.

[49] Christopher Torrence and Peter J. Webster. “Interdecadal Changes in the ENSO–Monsoon System”. *Journal of Climate* 12 (8 Aug. 1999), pp. 2679–2690. DOI: 10.1175/1520-0442(1999)012⟨2679:ICITEM⟩2.0.CO;2.

[50] Christopher Torrence and Gilbert P. Compo. “A Practical Guide to Wavelet Analysis”. *Bulletin of the American Meteorological Society* 79 (1 Jan. 1998), pp. 61–78. DOI: 10.1175/1520-0477(1998)079⟨0061:APGTWA⟩2.0.CO;2.

[51] Antoine Bernas, Albert P. Aldenkamp, and Svitlana Zinger. "Wavelet coherence-based classifier: A resting-state functional MRI study on neurodynamics in adolescents with high-functioning autism". *Computer Methods and Programs in Biomedicine* 154 (Feb. 2018), pp. 143–151. DOI: 10.1016/j.cmpb.2017.11.017.

[52] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. "Adjusting batch effects in microarray expression data using empirical Bayes methods". *Biostatistics* 8 (1 Jan. 2007), pp. 118–127. DOI: 10.1093/biostatistics/kxj037.

[53] Jean-Philippe Fortin et al. "Harmonization of multi-site diffusion tensor imaging data". *NeuroImage* 161 (Nov. 2017), pp. 149–170. DOI: 10.1016/j.neuroimage.2017.08.047.

[54] Angela Lombardi et al. "Extensive Evaluation of Morphological Statistical Harmonization for Brain Age Prediction". *Brain Sciences* 10 (6 June 2020), p. 364. DOI: 10.3390/brainsci10060364.

[55] Jean-Philippe Fortin et al. "Harmonization of cortical thickness measurements across scanners and sites". *NeuroImage* 167 (Feb. 2018), pp. 104–120. DOI: 10.1016/j.neuroimage.2017.11.024.

[56] Raymond Pomponio et al. "Harmonization of large MRI datasets for the analysis of brain imaging patterns throughout the lifespan". *NeuroImage* 208 (Mar. 2020). DOI: 10.1016/j.neuroimage.2019.116450.

[57] Yaroslav Ganin et al. "Domain-Adversarial Training of Neural Networks" (2015). DOI: 10.48550/arXiv.1505.07818.

[58] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier". Vol. 13-17-August-2016. Association for Computing Machinery, Aug. 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[59] Scott Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions" (May 2017). DOI: 10.48550/arXiv.1705.07874.

[60] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences" (Apr. 2017). DOI: 10.48550/arXiv.1704.02685.

[61] H. P. Young. "Monotonic solutions of cooperative games". *International Journal of Game Theory* 14 (2 June 1985), pp. 65–72. DOI: 10.1007/BF01769885.

[62] Hugh Chen, Scott Lundberg, and Su-In Lee. "Explaining Models by Propagating Shapley Values of Local Components" (2019). DOI: 10.48550/arXiv.1911.11888.

[63] Madhura Ingalhalikar et al. "Functional Connectivity-Based Prediction of Autism on Site Harmonized ABIDE Dataset". *IEEE Transactions on Biomedical Engineering* 68 (12 Dec. 2021), pp. 3628–3637. DOI: 10.1109/TBME.2021.3080259.

[64] Hao Guan et al. "Multi-site MRI harmonization via attention-guided deep domain adaptation for brain disorder identification". *Medical Image Analysis* 71 (July 2021), p. 102076. DOI: 10.1016/j.media.2021.102076.

[65] Anush Kamath, Sparsh Gupta, and Vitor Carvalho. "Reversing Gradients in Adversarial Domain Adaptation for Question Deduplication and Textual Entailment Tasks". Association for Computational Linguistics, 2019, pp. 5545–5550. DOI: 10.18653/v1/P19-1556.

[66] Pedro Domingos. "The Role of Occam's Razor in Knowledge Discovery". *Data Mining and Knowledge Discovery* 3 (4 1999), pp. 409–425. DOI: 10.1023/A:1009868929893.

[67] Sara Saponaro et al. "Multi-site harmonization of MRI data uncovers machine-learning discrimination capability in barely separable populations: An example from the ABIDE dataset". *NeuroImage: Clinical* 35 (2022), p. 103082. DOI: 10.1016/j.nicl.2022.103082.

[68] Alessandra Retico et al. “The effect of gender on the neuroanatomy of children with autism spectrum disorders: a support vector machine case-control study”. *Molecular Autism* 7 (1 Dec. 2016), p. 5. DOI: 10.1186/s13229-015-0067-3.