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 disorder (ASD) is a neurodevelopmental disorder manifesting from the early ages and involving behavioral and cognitive impairments. So far, no univocal biomarker have been detected to identify this disorder, and the most accurate diagnostic tool remains a behavioural and attitudinal test. In this work we focus on brain functional connectivity data to study the difference between patients with ASD and healthy controls. To tackle this challenge, we employed machine learning and deep learning models. Our database are Magnetic Resonance scans provided by the ABIDE dataset consisting on structrural and functional scans of approximately 2200 subjects collected from different medical centers. One half of them belong to normal individuals 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 normalization to a common template, and extraction of temporal series of different brain areas for each patient from the fMRI scans.
- Creation of a connectivity map for each patient. This connectivity matrix is created computing a correlation coefficient for every pair of timeseries of a single patient. Correlation coefficients were computed using two different approaches: a linear correlation of the two timeseries known as Pearson correlation, and a correlation based on time-frequency analysis of a timeseries pairs by means of wavelet transform.
- Implementation of a harmonization procedure to correct for potential biases and distinctive traits of data, linked to acquisition scan procedures employed in a medical center which can differ between different centers.
- Study and classification of connectivity matrices with deep learning methods. The two different connectivity matrices described above are compared to assess which one is able to yield the best separability between Controls and ASD. Classification of raw data and harmonized data implemented with different procedures are compared, and in addition a harmonization implemented inside a deep learning model through an adversarial learning technique was tested.
- Deep learning model is compared to a Random Forest classifier, a standard machine learning model, to determine if a deep model brings a 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 discerning between controls and subjects with ASD.

Organization of contents

Background: In this section we explain relevant theory arguments that lay the groundwork to all the thesis work. 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: from data acquisition to data 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 models, the learning procedure, and the important metrics for a proper evaluation of 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: dataset, the preprocessing steps, and the data preparation pipeline employed to obtain the correlation matrices.

Chapter 5 describes and analyze the ABIDE dataset, which is a publicly available collection of more than 2000 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 scan to a common template to run meaningful statistical analysis and to exctract brain timeseries form different brain areas aligned to a common coordinate system. We describe C-PAC which is the software employed to this image preprocessing and how we used it to conduct our processing.

Chapter 7 describes the two methods we used to calculate connectivity coefficients and create a functional connectivity matrix for each patient. 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 a correlation coefficient from time-frequency data of two timeseries.

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

Chapter 9 describes the structure and the general idea behind a domain adversarial neural network: 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 results obtained applying methods to our materials are presented. We illustrate the results of harmonization applied to our data and classification result of Pearson-based correlation coefficients and wavelet-based coefficients and in the end we study the most relevant feature from our data, that played an important role in discrimination of control/ASD. Chapter 12 Presents the results of the analytical harmonization to our data to have a visual understanding of how this process affects a feature distribution. Chapter 14 shows all the classification results obtained with a deep neural network on harmonized and non-harmonized (raw) data and the results of the domain adversarial neural network and compares them with results obtained using a simpler Random Forest classifier. Results of data dimensionality reduction are presented as well to assess if this is a meaningful procedure to reduce dimensionality on this dataset. At the end, chapter 15 shows the implementation of a feature importance assessment procedure and the results obtained with machine learning in the previous chapter. We show what feature contributed the most to the outcome of a prediction, and then to the discrimination of healthy and ASD patients, and subsequently we study what brain areas are mainly involved in this discernment.

Chapter 1

Introduction

1.1 Autism spectrum disorder

Autism spectrum disorder (ASD) is a neurological disorder 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 differ-

ences in brain anatomy. It has been studied how ASD could alter the symmetry between the two emispheres [**?**] 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 [9].

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 an 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 [10] [11].

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 [12].

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 [13].

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 [14]. 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 [15].

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.

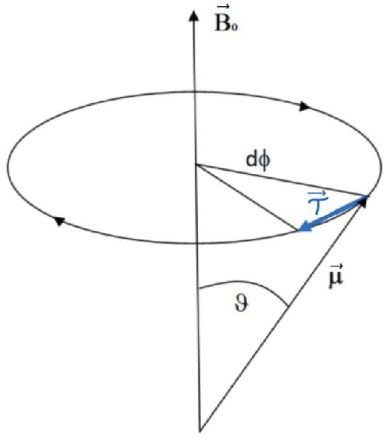


Figure 2.1: *Representation of a angular moment vector $\vec{\mu}$ under a static magnetic field $\vec{B}_0$ and the resulting moment $\vec{\tau}$ which causes the precession*

The physical principle this technique is based on, is the interaction of a nuclear spin of any tissue molecule, with an external magnetic field $\mathbf{B}_0$. Molecules that contribute the most to the signal are water, and specifically, hydrogen nuclei are the dominant source of signal in MRI.

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 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 2.2a 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 2.2b: 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 2.2b 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 2.2b 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.

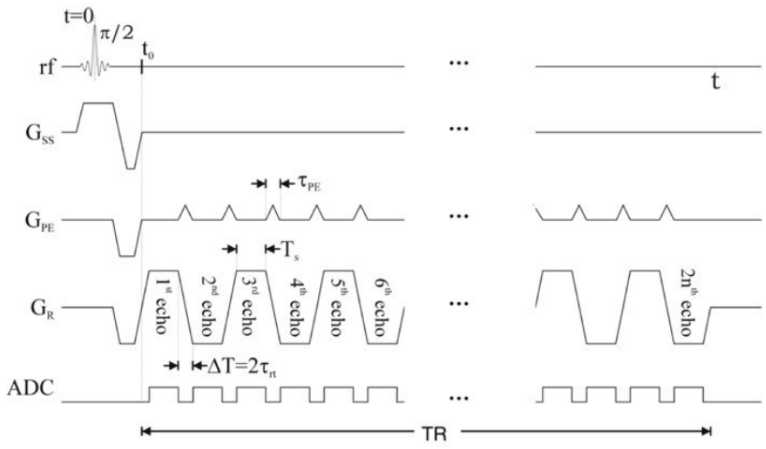
This acquisition technique allows the acquisition of a single slice in a short time (around 50-100 ms). The whole k-space for a single slice is acquired sequentially, following a snake-like pattern and the number of k-space line acquired after each single RF pulse is named differently according to the 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 $TE \approx T_2^*$.

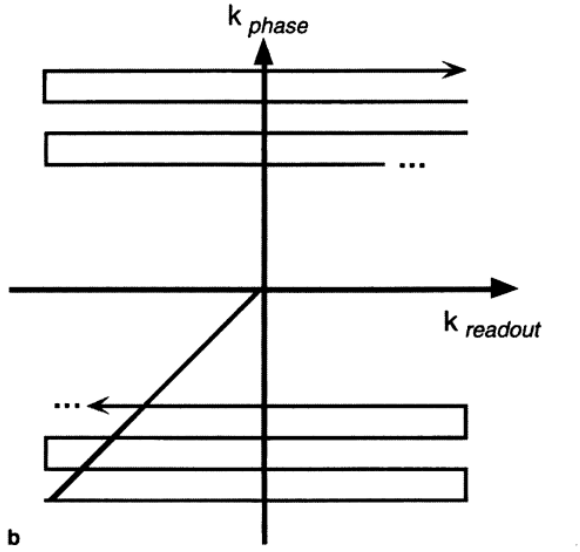
2.4 Functional MRI

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.



(a) *Gradients 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 peack.*



(b) *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.*

Figure 2.2: *Gradients of EPI acquisition sequence*

Blood can be portrayed as a colloidal mixture where blood cells constitute around 40-45 % of volume. [15]

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 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 [16].

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 2.3. 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.

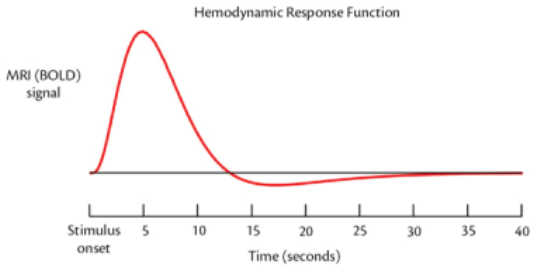


Figure 2.3: *Evolution of BOLD signal of an adult brain during time. At the instant t=0 the neuronal stimulus occurs; after that, it takes around six seconds to reach a maximum of signal, and a total of more than twenty-five seconds to return to its baseline value*

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. [17]

Chapter 3

Machine learning

Machine learning is a branch of Artificial Intelligence (AI) that aims to learn from data and apply this knowledge on new, unseen data. It is possible to make predictions or perform classification of data and, through the training process, gradually improve accuracy on the task a model performs. Regarding the training process of an algorithm, as a rule of thumb, the bigger and homogeneous is the dataset to train it, the better accuracy and generalization can be achieved.

According to the analysis we aim to perform, and the type of dataset we are working on, machine learning algorithms can be divided into two macro areas: supervised and unsupervised learning.

- From a dataset, consisting on a collection of data, each one containing a certain number of features, an **unsupervised** algorithm goal is to learn the principal properties from the dataset structure and to extract information from data without human labour to annotate and label each data. Analysis with unsupervised algorithms include clustering or dimensionality reduction.

  Clustering is maybe 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 source of variance across the data and create a representation in such a way that the dimensions of the new representation are statistically independent.

- On the other hand, when we are dealing with **supervised learning**, we work with a dataset where each data is associated with a label, specifying the class data belong to. Thus, the algorithm is trained to learn the common features for each class and to predict the correct label associated to each input data.

  Generally speaking, 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 of the human labour necessary to label each input data.

A subset of Machine Learning is Deep Learning, the main difference lies on the structure of its algorithms and on the learning techniques. Traditional machine learning methods consist on

algorithms like Random Forest, Support Vector Machines (SVM), K-nearest neighbor (KNN) and several more models, while deep learning includes models as 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 (see section 3.2) and to create a more powerful model that achieve good generalizability. Therefore, to properly understand a random forest algorithm, we need to start understanding how a Decision Tree classifier works.

Tree classifiers
A decision tree aims to learn distictive 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 *leave*, 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 different data samples, each one containing n features, the algorithm construct a tree following these simple steps: starting from the top node called *root*, it searches among the n 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 a single leaf is left.

The best split is performed according to a pre-defined objective function 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. By lowering the impurity of a node, it 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 Giny Impurity. [18]. 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 necessaries to understand the tree, 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 and the information of each node can be visualized in a clear plot. Other positive sides of decision trees are, their easy to use implementation which requires little data preparation and their rapidity since their complexity goes like $O(n_{features} n^2_{samples} log(n_{samples}))$ [19]

An example of a decision tree dealing with flower classification is shown in figure 3.1. We choose this example because is easier to understand how a decision trees works with this data where

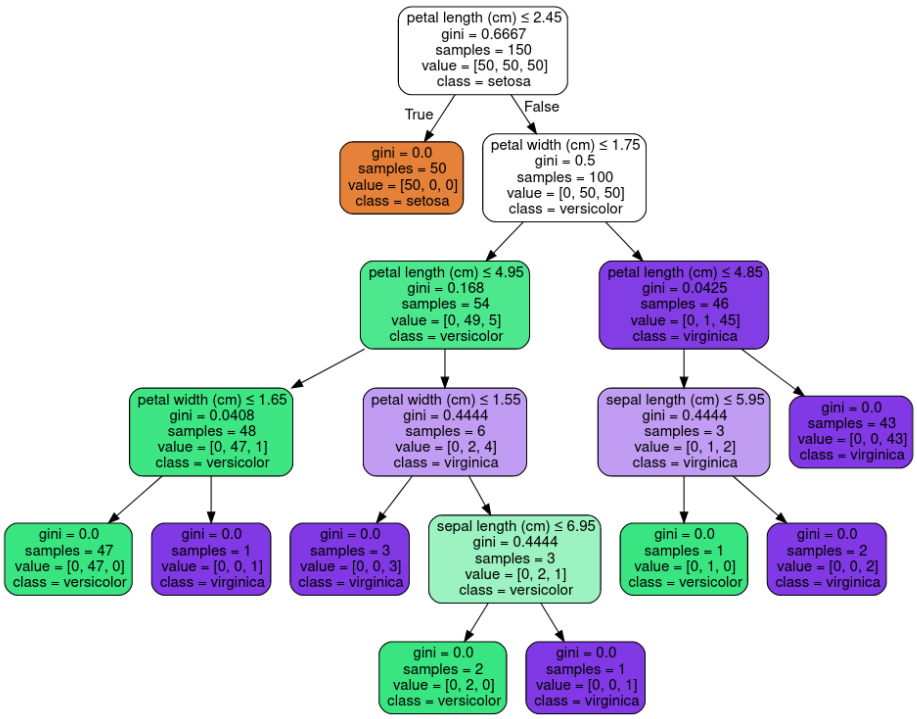


Figure 3.1: *A graphical representation of a decision tree for classification of the Iris dataset consisting on data belonging to three different classes of iris: Setosa, Versicolour, and Virginica. Each node of the tree shows the criterion used for its split, the Gini values, the total number of samples in that node and the main class that data belong to*

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]

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 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, where this probability is just the number of samples that reach that node divided by the total number of samples. [18] [20]

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 accurately learns the training dataset but is not able to generalize well to a test dataset. To prevent overfitting, one possible strategy is to early stop the iteration towards the leaf, and leave a node with more than a single sample left. An other strategy, which leads to the construction of a Random Forest, is the creation of different trees and put information together, in order to obtain stronger model, more prone to generalize information and reducing overfitting.

Random Forest

A Random Forest is a collection of decision trees where each tree is different from the others,

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

each tree tends to overfit, but following different patterns. Taking advantage of this process, we could 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: when selecting the number of samples to grow each tree on, and when selecting 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≤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 limit the number of feature available and compute 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.
4. 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 important features from each tree and by averaging these results.

3.2 Deep Learning: ANN

In the previous paragraph we discussed one of the most important machine learning algorithm, but, as mentioned before, there is a subset of machine learning algorithm with a common typical structure, thanks to which they get their name of deep neural networks. The structure of algorithms belonging to this family are inspired by a brain neuronal structure, and even if in a simplified way, they try to emulate the learning process of a brain. These algorithms own the adjective "deep "to their structure: they are organised in layer, each one containing several fundamental unit called neuron. Neurons between layers are connected to each other like synapsis transmit a signal between neurons in a biological brain. Thanks to this structure which reminds a neuronal brain structure,they are called artificial neuronal networks

The foundamental unit which constiture a neuronal network is an artificial neuron, one of the most relevant example of artificial neurons is the *perceptron* shown in figure 3.2.

A 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 $y = f(z) = f\left(\sum_i w_i x_i + b\right)$. The function 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 3.3.

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 where each neuron acting like a perceptron, 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. Once collected every output

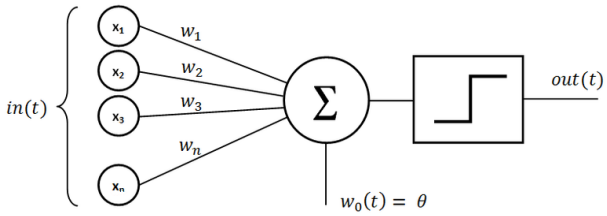


Figure 3.2: *A schematic representation of a perceptron: this perceptron receives an input vector t, with n features $x_1, ...x_n$, each one weighted with a different weight $w_1, ...w_n$ and a offset w_0, and compute a linear combination of them and returns an output, activated by a step function: it returns 0 if the linear combination's value doesn't reach a certain threshold, or returns the value itself if it does.*

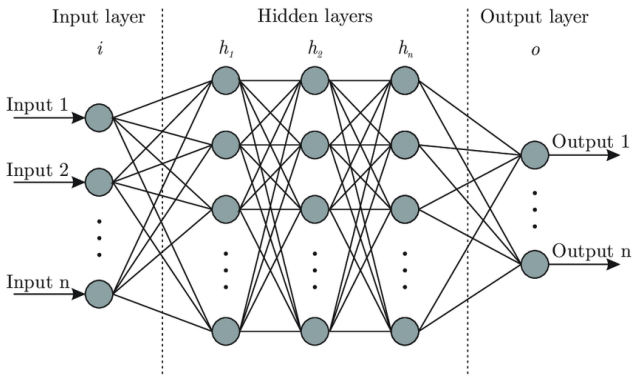


Figure 3.3: *Schematic representation of an artificial neural network containing an input layer, three hidden layers and an output layer: a single vector contained n features (input 1, ... input n) is given as input to the first Input layer i. All the inputs are linked to every neurons of the first hidden layer, which would compute a linear combination of its inputs. All the hidden layers are linked to the other hidden layers, and at the end, n different outputs are returned.*

from the first layer neurons, they are passed to the second layer becoming the input to each neuron belonging to this layer. This is reiterated through all the layers up to the final output layer. As schematized in figure 3.3 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. At the beginning, weights are randomly set, but during training they are updated in order to improve network performances.

Using matrix formalism the entire input-output process can be written as

$$\mathbf{y} = \sum_{j}^{n} w_{ij} x_j + b_i \tag{3.2}$$

where $\mathbf{x} \in \Re^n$ $\mathbf{y} \in \Re^m$ being m the output dimension determined by the number of neurons in the output layer. $\mathbf{y}$ represents the output, or prediction of the model. The process through which given an input data $\mathbf{x}$ we obtain an output $\mathbf{y}$ is called *forward propagation*.

In order to train a model, the prediction is compared with the actual value of the label associated to the input data point, and during a process called *backpropagation* the algorithm modifies its weights in order to minimize the difference between the actual and the predicted value.

The goal of a machine/deep learning algorithm is to learn from data using a train dataset, and generalize information in order to perform well on an unseen dataset called test dataset. Performing well means to produce a low error on the test dataset after minimizing the error on train dataset. To fully define a neural network we just need some 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)
- Learning rate

Number of layers and numbers of neurons are connected to one of the most important concept in machine learning that characterize a network: the concept of *capacity*. Capacity qualitatively refers to the level of complexity that a model is able to learn. This is 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 shallow network, when the model has a small number of parameters in regard to the number needed to explain the input features. To have low parameters results in poor performances because the model is not able to learn the underlying structure of a complex data set.
- On the contrary, overfitting occurs when the model has too many parameters compared to those required to learn the input features. What happens then is that the model memorizes all the data it sees during train but it is not able to generalize information. As a result, the model performs well on the train dataset and has low performances in the unseen test dataset.

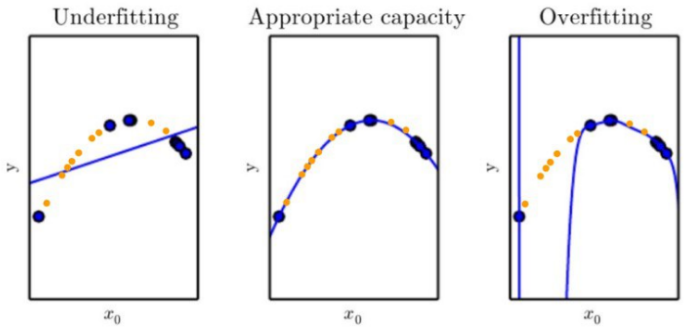


Figure 3.4: *Underfitting, overfitting and appropriate fit in a 2D dataset: three models with different capacities are trained on blue data, if the model is too simple, in this case if the fit function has a few free parameters, it is not able to properly fit data, on the other hand, if it has too much parameters in respect to how much we would need to fit our data, it perfectly match train data distribution, passing through each data point, but performs very poorly on the test dataset (orange points). With an appropriate number of parameters, (middle figure) the model is able to fit train data, and generalizes well to test data.*

To understand this concept is can be helpful to provide an example in a two-dimensional space. If we have to fit a dataset consisting of some points sampled from a quadratic curve, we can choose different functions, for example a linear function, a quadratic function and a polynomial function with grade equal to the number of data points. As shown in figure 3.4 the linear model is not able to describe the data distribution while the polynomial function has too many parameters, and it is able to fit the train distribution, but it does not suit to describe data points belonging to the test, sampled from the same quadratic distribution. Furthermore, if the number of parameters (the grade of the polynomial in this case) is greater than the number of data points, we obtain that an infinite number of different curves are suitable to fit our data, and finding the one which performs well even on the test dataset is an hard task. We should therefore pay attention when choosing the number of parameters on a model to avoid under- or more likely over- fitting.

To monitor the evolution of the training process of a model it is a common practice to split the train dataset into two subsets: one that will actual constitute the train dataset, and a smaller one, called validation dataset used to make constant checkups on how the model is learning with training data. The validation dataset is used for the fine tuning of hyperparameters and it typically consists of 10-30% of the train dataset. Just in this paragraph to distinguish between the two train dataset we will denote as "Train ", the whole train dataset and as "train "the subset of the Train dataset left after its split into a train and a validation set. The whole dataset will then be divided into approximately 70% Train and 30% test, and the Train dataset itself will be divided into 70% actual train and 30% validation. When the best combination of hyperparameters is found, it is possible to actually train the model using the entire Train 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, to avoid overfitting, implement some regularization techniques to achieve good generalization performances.

Some of the most popular are Dropout and Batch-Normalization

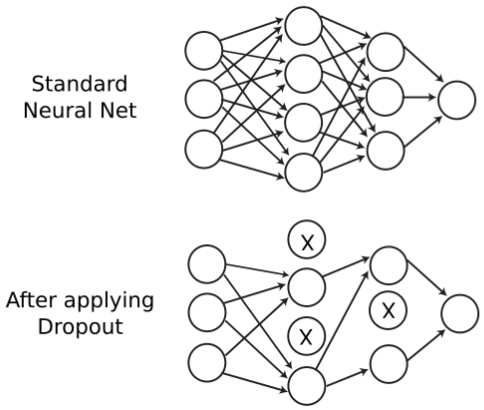


Figure 3.5: *Schematic representation of a 0.5 dropout procedure: in a standard neural network, each neuron is linked to all the others, but if we implement a 0.5 dropout, for each iteration, neurons on hidden layers have a 50% probability to be dropped and excluded from the computation of outputs*

- 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 drop 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 the missing connection; doing so, every neuron cannot rely on the the input of all the preceding 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 average all the outputs.

- BatchNormalizzation is a regularization scheme that has been commonly adopted since its introduction in 2015 [21]. It is based on the observation that a neural network works and performs better when its input are normalized because this prevents the saturation of its neurons. A neuron can in fact saturate and settle to a certain value because of an high input, this causes the neuron outputs value to be always close to the asymptotic end of its activation function (see section 3.3), 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 $x_i^l \rightarrow \tilde{x}_i^l = \frac{x_i^l - \mathbb{E}[x_i^l]}{\sqrt{Var(x_i^l)}}$

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 for a proper train procedure.

To work this out, a procedure called k-fold cross validation can be implemented, at the cost of increasing computational costs. It consists on the creation of k different partitions of the main whole dataset (before splitting it into train + test). 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

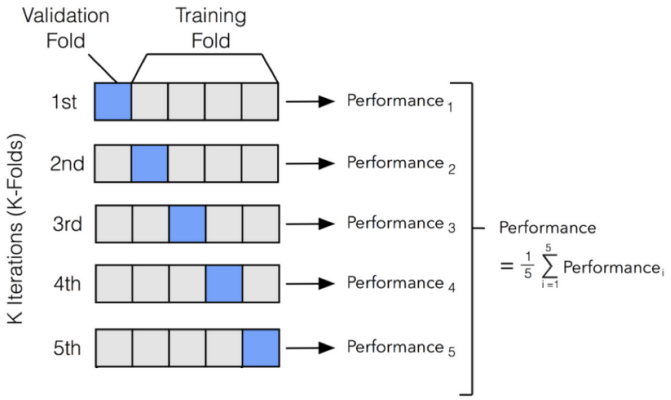


Figure 3.6: *Schematic representation of a k-fold cross validation procedure with k = 5: Dataset is partitioned into 5 subsets, four of which are used for training and the remaining one for the testing dataset. during each iteration a model is trained over 4 different folds and tested on the remaining one, until each fold has been used at least once as part of training and once as test.*

as test, and the others as train. Thus, for each k-th run there are two subsets (a train and a test) of the original dataset. Following this procedure, at each iteration, the train dataset is different, and the model is tested on a different dataset each time. An image showing this procedure is 3.6. In practice what is usually done, is not a sequential partitioning the dataset as shown in figure 3.6 but a shuffled partition of data. This avoids creating folder containing all the same label in case the dataset was ordered, but randomly picking data in order to create subsets containing the same proportion between classes as in the main dataset.

Overall, we can imagine this process like 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 of these models, and we take as a result the average score across all the outputs.

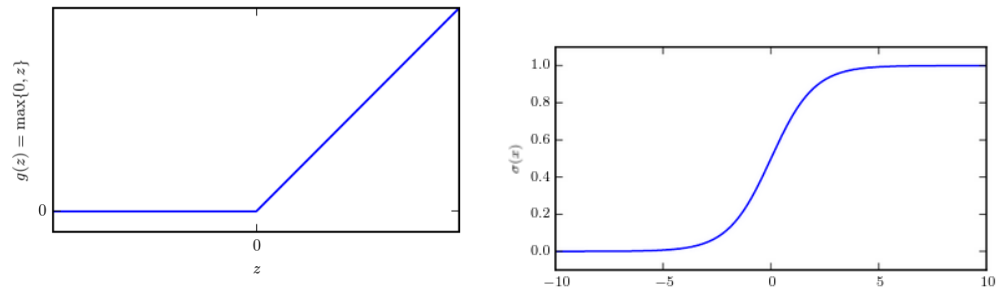
3.3 Activation functions

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. A linear activation function outputs a value $f(z) = a \cdot z + b$ where we denote as z the scalar product between an input vector $\mathbf{x}$ and the weights vector $\mathbf{w}$ plus an eventual offset w_0, and as $\mathbf{z}$ the matrix-vector product between the matrix of weights $\mathbf{w}$ and the input vector $\mathbf{x}$ and an offset vector $\mathbf{w_0}$.

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 what data we are working on, or on what kind of classification task we are performing. Some of the most popular are:

- The ReLU function shown in figure 3.7a is defined as

$$\phi(z) = max(0, z) = \begin{cases} 0 \ for \ z < 0 \\ z \ for \ z > 0 \end{cases} \tag{3.3}$$



(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 3.3.*

(b) *Sigmoid function: returns a value between 0 and 1, according to equation 3.4*

Figure 3.7

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

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

- The sigmoid function, or logistic function show in figure 3.7b is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.4}$$

  and outputs a real number between 0 and 1. For this reason is a common choice when we have 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 the class 0 or 1.

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

$$\phi(z) = p(y = i|z) = \frac{e_i^z}{\sum_{j=1}^{M} e^{z_j}} \tag{3.5}$$

  It is applied to the vector $\mathbf{z}$ and describes the probability of $\mathbf{x}$ to belong to the class i over a total of M classes. With 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 $\mathbf{z} = \{z_1, ..., z_M\}$

3.4 Loss functions

In order to train a network and improve its performances we compare the predicted output value with the 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 backpropagation, that we discuss in section 3.5

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

$$H(y, \hat{y}) = -\sum_{m=0}^{M-1} y_i \cdot log(\hat{y}_i) \tag{3.6}$$

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 is called a binary classification, and its associated label usually have values $y = \{0, 1\}$. If we explicit the sum of equation 3.6 for M = 2, cross entropy for a binary classification, for one observation can be calculated as

$$\ell_i = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \tag{3.7}$$

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

$$L = \frac{1}{N}\sum_{i}^{N} \ell_i = -\frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{3.8}$$

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*

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{m=0}^{M-1} y_{im} log(\hat{y}_{im}) + (1 - y_{im}) log(1 - \hat{y}_{im})) \tag{3.9}$$

In equations 3.8 and 3.9 $\hat{y}$ 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.

Relation of the loss function with a Maximum Likelihood Extimation

The estimation of the minimum of the loss function can be seen as a process of Maximum Likelihood Extimation. Given a probabilistic model depending on m different parameters $\theta_1 ... \theta_m$, the likelihood is a function that describes the probability of observing a value $\hat{y}_i$ as a function of set of parameters $\{\theta_i\}_1^m$. If we set values of these parameters we obtain the probability of observing the value $\hat{y}_i$ under the given model [22].

$$\mathcal{L} = (\theta_1, ..., \theta_m | \hat{y}_i) = P(\hat{y}_i | \theta_1, ... \theta_m). \tag{3.10}$$

For a subset of observed value we can write the likelihood as $\mathcal{L} = P(\hat{y}_1, ..., \hat{y}_N | \theta_1, ... \theta_m)$. If all the observed values are independent, we can write the total probability as the product of single probabilities of observing each sample $\hat{y}_i$ given a model with parameters $\{\theta_i\}_1^m$:

$$\mathcal{L} = \prod_{i=1}^{n} p(\hat{y}_i|\theta_1, ...\theta_m). \tag{3.11}$$

We consider the problem of a binary classification using a deep neural network. With this model, parameters $\{\theta_i\}_1^m$ correspond to the inner weights of the networks $\mathbf{w} = \{w_i\}_1^m$. Input data are a set of N datapoints $\{x_1...x_N\}$ each one associate with a label $\{y_1...y_N\}$ which can be either 0 or 1. Overall, the whole dataset can be denoted as D = $\{(\mathbf{x_i}, y_i)\}$.

Given an input data $\mathbf{x_i}$ and an activation function (sigmoid, for example, since we are performing binary classification) we model the probability of $\mathbf{x_i}$ to belong to the class $y_i = 1$ as:

$$P(y_i = 1|x_i, \mathbf{w}) = \sigma(z_i) = \frac{1}{1 + e^{-z_i}} \tag{3.12}$$

where z is defined as in section 3.3 $\mathbf{z_i} = \mathbf{x_i w}$ And since we are dealing with a binary classification, where y_i can be either 0 or 1, the probability to belong to one class is 1 minus the probability to belong to the other:

$$P(y_i = 0) = 1 - P(y_i = 1) \tag{3.13}$$

Using this result for a set of data $D = \{(\mathbf{x_i}, y_i)\}_1^N$ and substituting the formula for a single probability in the likelihood function 3.11 we obtain the likelihood of observing the dataset D under a model with parameters $\mathbf{w}$

$$\mathcal{L} = P(D|\mathbf{w}) = \prod_{i=1}^{N} \sigma(\mathbf{x_i w})^{y_i}(1 - \sigma(\mathbf{x_i w}))^{1-y_i} \tag{3.14}$$

To perform a Maximum Likelihood Estimation we look for a maximum in this function by computing partial derivatives. However, since computing the derivative of a product is a nontrivial task, we can consider the logarithm of $\mathcal{L}$ and compute its maximum since the logarithm is a monotonic function and compute its maximum corresponds to computing the maximum of the function itself. The logarithm of equation 3.14 is

$$log(\mathcal{L}) = \sum_{i=1}^{N} y_i log(\sigma(x_i \mathbf{w})) + (1 - y_i) log(1 - \sigma(x_i \mathbf{w})) \tag{3.15}$$

which taken with a negative sign, and averaged over all the N data samples corresponds to the cross-entropy function in equation 3.8.

Thus, minimizing the binary-crossentropy loss is the equivalent of a maximum likelihood estimation for the parameters of our model.

3.5 Gradient descent and Backpropagation

During the training of a model, after the calculation of the loss function, the network moves to the estimation of the best parameters through an algorithm called *backpropagation*. To perform the minimization of a loss function, even if it would be theoretically possible to find a minimum by means of an analytical way, 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

$$\mathbf{w}^{i+1} = \mathbf{w}^i - \eta \nabla_{\mathbf{w}} L(\mathbf{w}^i) \tag{3.16}$$

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 is been seen, resulting in a huge computational cost.

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

SGD works by approximating the gradient of the cost function calculated with 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 $0 \leq \gamma \leq 1$ and it 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 into account even the steepness all over the dimensions. To accomplish so the algorithm has to be improved by adding the calculation of second order momenta also called *uncentered variance*. This procedure, would ideally bring to the calculus of the hessian matrix but at the cost of increasing computational costs. Recently introduced algorithm can accomplish this task by approximating the calculus of the second momenta.

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.

One of the most popular among these algorithms is **Adam**: it accomplish this computation by making use of two different optimization algorithms: AdaGrad and RMSProp. It is a powerful algorithms since it adapts the learning rate taking into account both the first and the second momenta. This leads it to perform better and quicker in finding the minima than simpler SGD with momentum algorithm.

The process of computing gradients is an important step of the backpropagation algorithm, which is the process through which the weights of a netwrork 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 backpropagation 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 3.16.

3.6 Dimensionality reduction: PCA

Principal Components Analysis is a method to perform dimensionality reduction. It is an unsupervised learning algorithm which aims to reduce the dimensionality of input data, preserving as much information as possible from it. It does so by learning a new representation of data with lower dimensionality than the initial one and whose element have no linear correlation with each other. It performs then a projection of data on to this new space, created such that its axes lie on the directions along which data variance is the biggest.

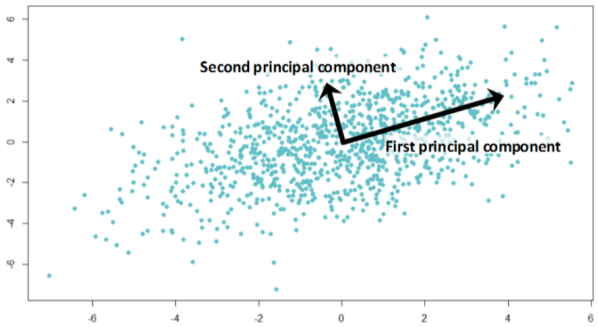


Figure 3.8: *Example of the first two principal components in a 2D dataset: the first principal component is placed along the direction where variance is greater, and the second component is in the orthogonal direction.*

PCA seeks to find an orthonormal basis so that the first base vector corresponds to the direction along which the variance of data is the greatest. 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 $(i-1)-th$ vectors that maximize the left variance. In figure 3.8 is reported a graphic example of two PC 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

$$\Sigma = \frac{1}{n-1} X^T \cdot X \tag{3.17}$$

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 x m matrix X that (in the case X is a real matrix), allows to rewrite it as $X = USW^T$ where U and V are respectively m x m and n x n orthogonal matrices whose columns are called *left* and *right singular vectors* of X, and S is a m x 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

$$\begin{aligned} X^T X &= (USW^T)^T (USW^T) \\ &= WS^T U^T USW^T \\ &= WS^2 W^T \end{aligned} \tag{3.18}$$

where we used the definition of orthogonal matrix for U $U^T U = \mathrm{I}$.

We therefore rewrite the covariance matrix

$$\Sigma = \frac{1}{n-1} WS^2 W^T. \tag{3.19}$$

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.

$$\begin{aligned} Var[Z] &= \frac{1}{n-1} Z^T Z \\ &= \frac{1}{n-1} W^T X^T XW \\ &= \frac{1}{n-1} W^T WS^2 W^T W \\ &= \frac{1}{n-1} S^2 \end{aligned} \tag{3.20}$$

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.

$$\frac{\lambda_i}{\sum_{k=0}^{m} \lambda_k} \tag{3.21}$$

Note: 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.

3.7 How to assess network performances

After the model has been trained, it is evaluated on the test set to assess its performances on a new dataset. A comomon 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 some important 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

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.22}$$

Even though accuracy may seem a good parameter, it is not the most accurate one, when we're dealing with a 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 data, with just 2 samples belonging to B and the rest to A, the model predicts every data belonging to the class A, yet we would get a score of 80 % accuracy, which is not a thruthful result.

For this reason there are other ways to evaluate performances of a network that overcome this problem. One of this is based on the introduction 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.

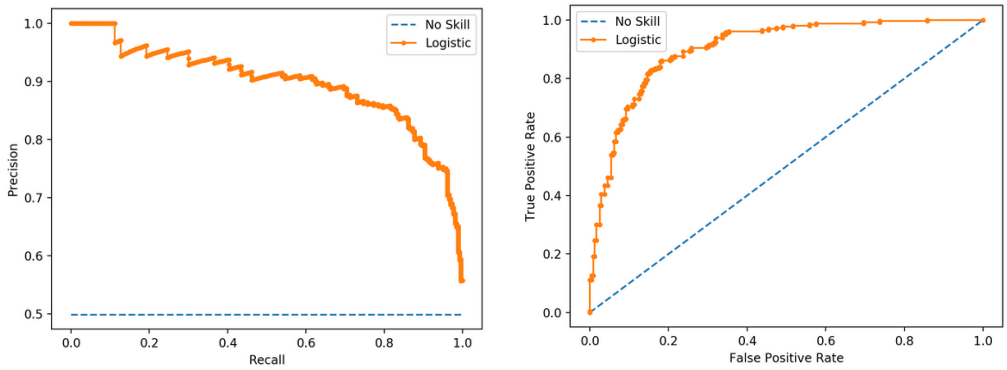
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{3.23}$$

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 set the recall to (100-x)%, this operation of establish a threshold is usually referred 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 3.9a. 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 precition "

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 (recall) and the false positive rate (FPR) and shows the evolution of TPR vs FPR for diefferent 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's 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 when a model is just randomly predicting and it corresponds to a curve laying on the bisector line.

The AUC can be regarded as the probability that a randomly picked point from the positive class, will have an 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 3.9b



(a) *Precision-recall curve representing values of precision and recall for different thresholds values*

(b) *ROC curve representing values of True positive and False positive rates for different thresholds values*

Figure 3.9

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, 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 collected 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 important concepts related to this field, that were subsequently readapted, for the implementation of this important 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 [23]

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-coperative and cooperative games.

- Non-cooperative games concern competition between individual players who don't cooperate each other and can't form coalitions. The main task on non-cooperative games can be summarised as the search for a good strategy for each player. Each player's objective 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 [24].

- 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 were named after him: Shapley values [25][26]. 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 of.

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 between each player of the subset, where "fair "is to be intended as, proportional to his 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's score with player i and the coalition's score without him marginal contribution $= \nu(S \cup \{i\}) - \nu(S)$. [27]

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

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

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

MATERIALS & METHODS

Chapter 5

Dataset: ABIDE I & II

Data we are going to work on belong to the ABIDE dataset (Autism Brain Images Data Exchange): a project founded with the aim of investigating ASD using magnetic resonance structural 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 number like _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 patient, such as age, sex, Full Intelligence Quotient (FIQ), eye status during the scan (open or closed), and every additional clinical information provided by patients.

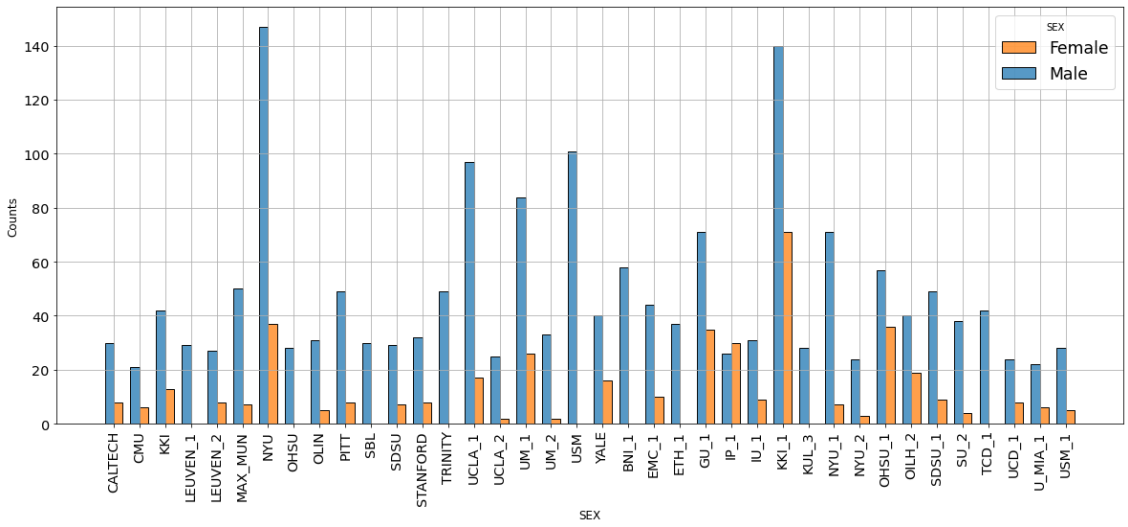
The vast majority of patients are males as shown in figure 5.1a, for a total amount of 1804 males and 422 females. The number of males is greater than the number of females as a consequence of the greater probability for male to be affected by ASD. The ratio between males and females has been estimated to be approximately 4:1 [28]. 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).

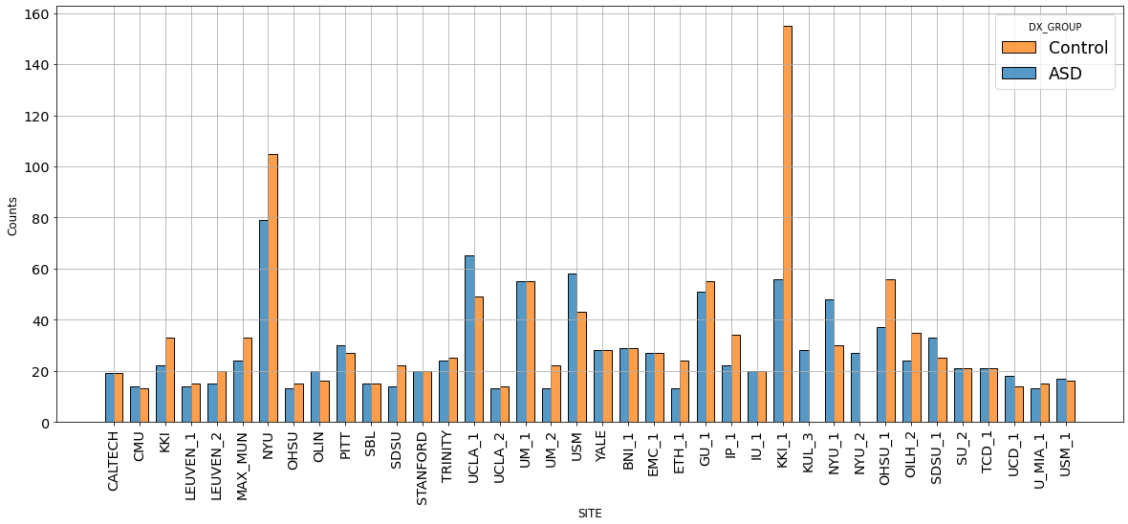
Information about full intelligence quotient, whose distribution is shown in figure 5.3a, are not

Acronym	Site name	Scanner name	TE [ms]	TR [s]	B_0 [T]
CALTECH	California Institute of Technology	Siemens TrioTim	30	2	3
CMU	Carnegie Mellon University	Siemens Verio	30	1.5	3
KKI	Kennedy Krieger Institute	Philips Achieva	10	2.34	3
LEUVEN_1	University of Leuven	Philips Intera	33	1.68	3
LEUVEN_2	University of Leuven	Philips Intera	33	1.65	3
MAX_MUN	Ludwig Maximilians University Munich	Siemens Allegra	30	3	3
NYU	NYU Langone Medical Center	Siemens Allegra	15	2	3
OHSU	Oregon Healt and Science University	Siemens TrioTim	30	25	3
OLIN	Olin Neuropsychiatry Research Center	Siemens Allegra	27	1.5	3
PITT	University of Pittscurgh School of Medicine	Siemens Allegra	25	1.5	3
SBL	Social Brain Lab	Philips Intera	30	3.2	3
SDSU	San Diego State University	GE MR750	30	2	3
STANFORD	Stanford University	GE SIGNA	30	2	3
TRINITY	Trinity Centre for Health Sciences	Philips Achieva	28	2	3
UCLA_1	University of California Los Angeles	Siemens TrioTim	28	3	3
UCLA_2	University of California Los Angeles	Siemens TrioTrim	30	2	3
UM_1	University of Michigan	GE Signa	30	2	3
UM_2	University of Michigan	GE Signa	30	2	3
USM	University of Utah School of Medicine	Siemens TrioTim	28	2	3
YALE	Yale Child Study Center	Siemens TrioTim	25	2	3
BNI_1	Barrow Neurological Institute	Philips Ingenia	25	3	3
EMC_1	Erasmus University Medical Center Rotterdam	GE MR750	30	2	3
ETH_1	ETH Zurich	Philips Achieva	25	2	3
GU_1	Georgetown University	Siemens TrioTim	30	2	3
IP_1	Institute Pasteur & Robert Debrè Hospital	Philips Achieva	45	2.7	1.5
IU_1	Indiana University	Siemens TrioTim	28	0.813	3
KKI_1	Kennedy Krieger Institute	Philips Achieva	30	2.5	3
KUL_3	Katholieke Universiteit Leuven	Philips Achieva	30	2.5	3
NYU_1	NYU Langone Medical Center	Siemens Allegra	78	5.2	3
NYU_2	NYU Langone Medical Center	Siemens Allegra	78	5.2	3
OHSU_1	Oregon Healt and Science University	Siemens TrioTim	30	2.5	3
OILH_2	Olin Institute of Living Hartford	Siemens Allegra	27	1.5	3
SDSU_1	San Diego State University	GE MR750	30	2	3
SU_2	Stanford University	GE Signa	30	2	3
TCD_1	Trinity Centre for Health Sciences	Philips Achieva	27	2	3
UCD_1	University of California Davis	Siemens TrioTim	24	2	3
U_MIA_1	University of Miami	GE	30	2	3
USM_1	University of Utah School of Medicine	Siemens TrioTim	28	2	3

Table 5.1: *Table of the sites included in ABIDE dataset, with the corresponding acronym. This acronym labels the site and the corresponding sample with a number such as _1 or _2. For each site, the main information about scan type, static magnetic field B_0 echo time TE and repetition time TR are provided.*



(a) *Histogram of male and female subjects per site, belonging to ABIDE I and ABIDE II dataset*



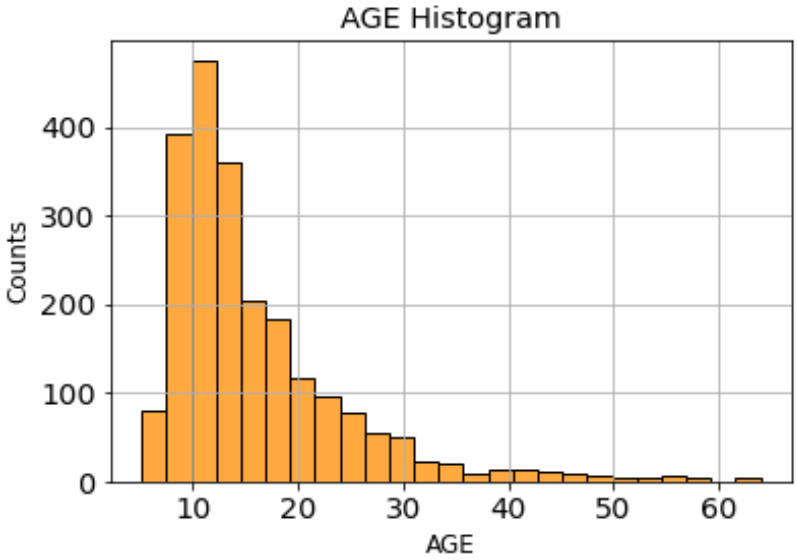
(b) *Histogram of control and ASD subject per site, belonging to ABIDE I and ABIDE II dataset*

Figure 5.1

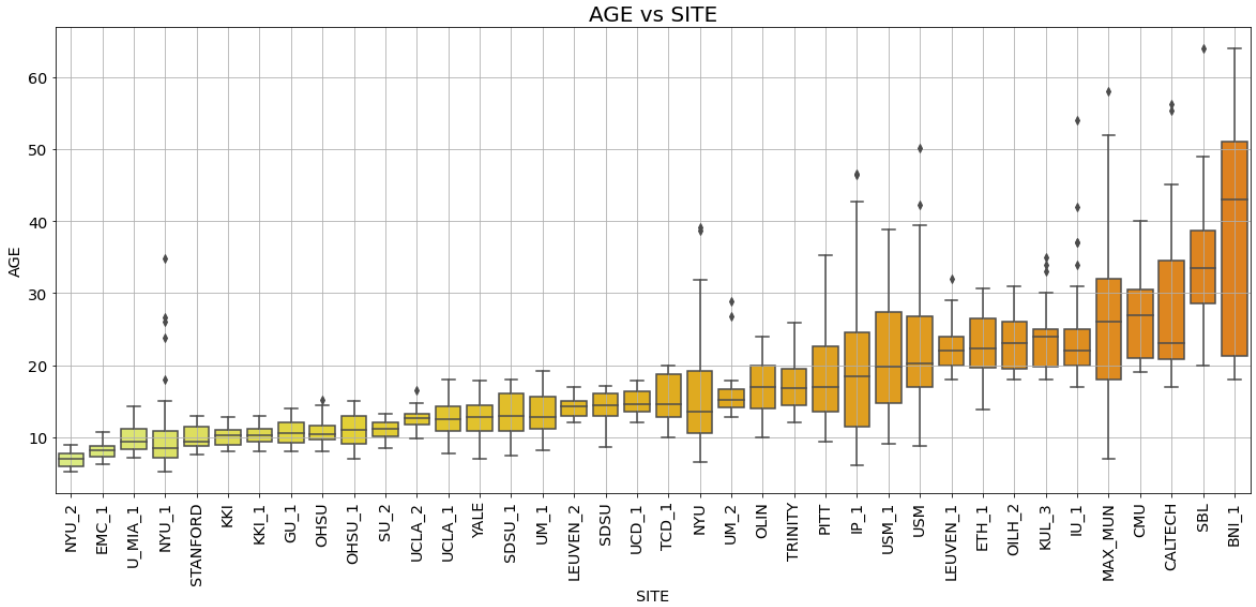
provided for every participant, in fact 171 patients out of 2226, coming from sites UM1 and EMC1 (figure 5.3b) lack this information. In our further analysis, before proceeding these values were replaced by the average value of all the other values.

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 patient 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.

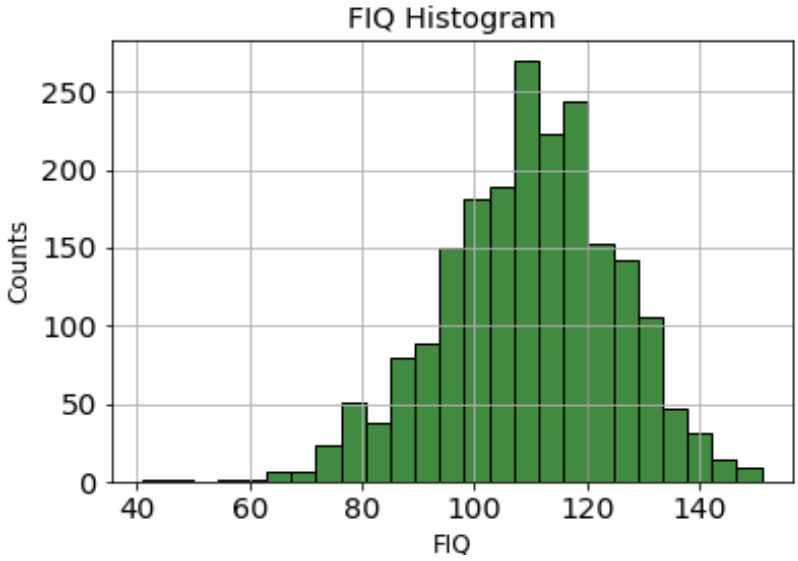


(a) *Ages distribution across the whole ABIDE dataset.*

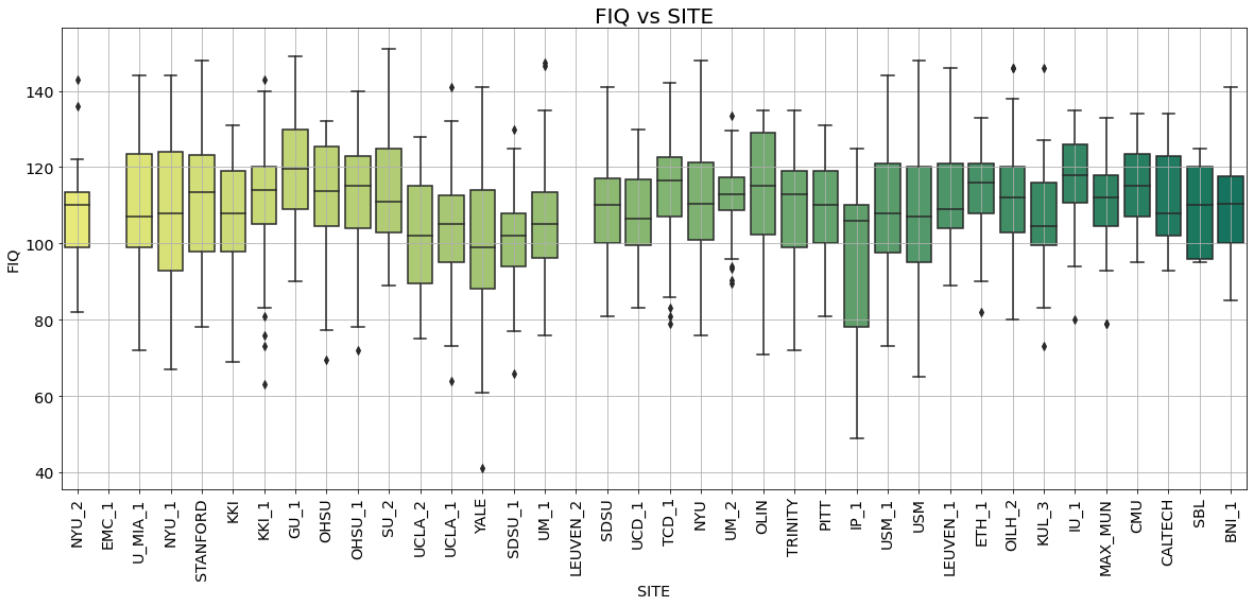


(b) *Boxplot of ages per sites, sorted by increasing median age.*

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

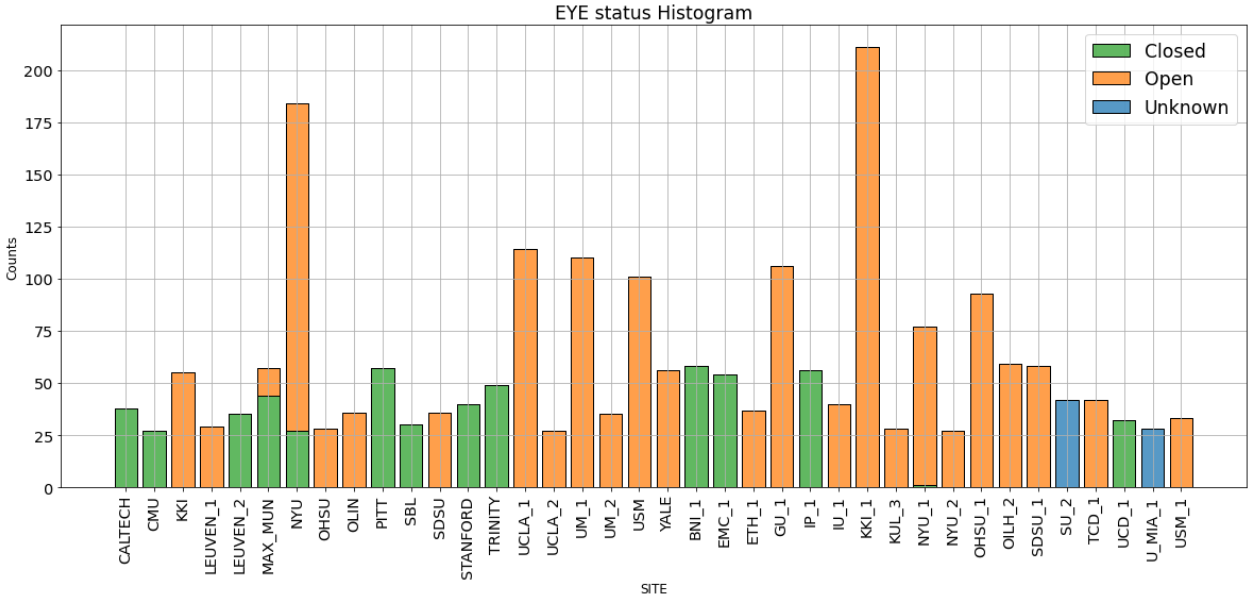


(a) *FIQ distribution across the whole ABIDE dataset*

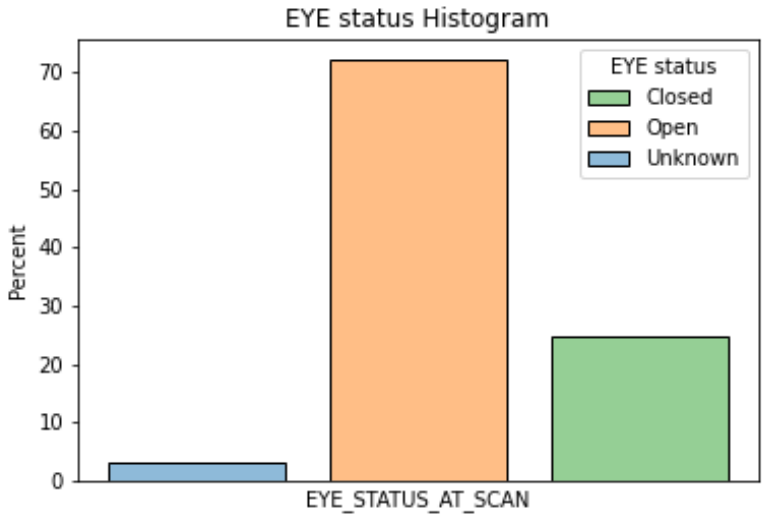


(b) *Boxplot of patient's FIQ per site*

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 analysis. 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 [29].

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, and 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 distortion in EPI images caused by inhomogeneities in the magnetic field arising from flaws of the external magnetic field, or caused by different tissue within the brain which cause localized distortion. 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 neighbored voxels. To this end, a gaussian kernel with a chosen FWHM is applied to create the weights. 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 from hardware such as an effect called "drift "namely the slow change of the BOLD baseline signal over time. This is accomplished by applying a high-pass filter to remove very low frequency components. This filtering operation is usually referred as a removal of linear trends, since we are removing from a signal only components with a frequency lower than the low-frequency fluctuations of BOLD signal. At the same time, a low-pass filter is applied to remove all 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 to universally describe 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 Talairach and Monreal Neurological Institute's MNI152, adopted by the International Consortium of Brain Mapping (ICBM) as the international standard, replaced the previous Talairach and Tournoux template. Because the adoption as international standard it is also called ICBM152 but 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.[30]

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

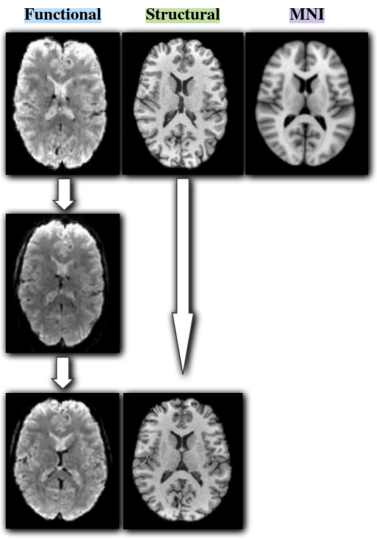


Figure 6.1: *Registration steps: starting from the first row there are the acquired functional and structural images and the MNI template respectively The second row shows the functional image registered on the structural space The third row shows the final images both registered to the MNI152 template*

From Talairach-Tournox to MNI152 template
The Teilarch 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 which so far had limited the accuracy of statistical analysis. They introduced a common coordinate system to identify different brain locations, based on some anatomical landmarks, and introduced a spatial transformations to match different brains. They choose 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 transformation, 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 transform 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 [31] [30].

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 we are carrying out. Some of the most popular atlases are:

- The H-O atlas was first employed in 2006 [32] and consists of 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) parcellation is shown in figure 6.2.
- The Automated Anatomical Labeled, an atlas created in 2002 [33] 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 [34] for functional parcellation, consisting of 200 and 392 ROIs respectively [2].
- The Desikan atlas [35] originally created in 2006 and consisting on 68 regions also underwent different transformation towards a finer parcellation of the brain.

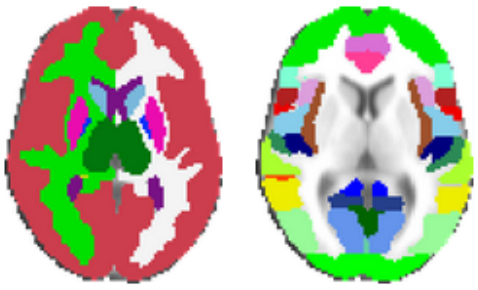


Figure 6.2: *Horizontal section of Harvard-Oxford subcortical (sx) and cortical (dx) atlased*

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 [36]. 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)

[2] http://ccraddock.github.io/cluster_roi/atlases.html
[3] http://preprocessed-connectomes-project.org/abide/C-PAC.html

- 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. 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 analysis 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 [37].

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

- 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 fulfilled 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.

[4]https://afni.nimh.nih.gov/
[5]https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslOverview
[6]http://stnava.github.io/ANTs/

- Distortion correction using PhaseDiff and AFNI 3dQWarp.
- Create a brain-only mask of the functional data using AFNI 3DAUTOMASK. **ForestGreen**
- 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 (HO), AAL Toolbox for Automated Anatomical Labeled (AAL), pyClusterROI for CC200, CC400 etc.

The AAL atlas employed in C-PAC is an intermediate version among the different releases of this atlas and consists on 116 ROIs[7].

redVa bene parlare di desikan DesikanKilliany senza dire qual è il software che lo fornisce? Non si trovano informazioni.. C-PAC employs three different Desikan atlases, the one we found more information about is a modified intermediate version named DesikanKilliany consisting on 94 ROIs of which 32 cortical regions each side, 3 ROIs belonging to cerebellar vermis and 29 subcortical regions.

According to previous studies [11] the atlas that gives best classification performances is Harvard-Oxford, so this is the atlas we choose to employ for our work.

The H-O atlas employed by C-PAC is provided by FSL library [8], and as includes both subcortical and cortical atlases, even though there's only 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 [9]. 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 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 analysis. 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 it they follow a similar trend, there are some local differences between the two plots. This slight difference is most likely due to two main reasons: the differences in software version, and the lacking of a detailed step-by-step pipeline legend showing the value of all the sub parameters employed during the C-PAC analysis pipeline. 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 found though the old pipeline configuration file employed on ABIDE preprocessed [10], but since it belongs to a previous version of C-PAC, it lacked 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.

[7] http://preprocessed-connectomes-project.org/abide/Pipelines.html
[8] https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/
[9] Subcortical ROIs https://neurovault.org/images/1700/ Cortical ROIs https://neurovault.org/images/1705/
[10] https://github.com/preprocessed-connectomes-project/abide

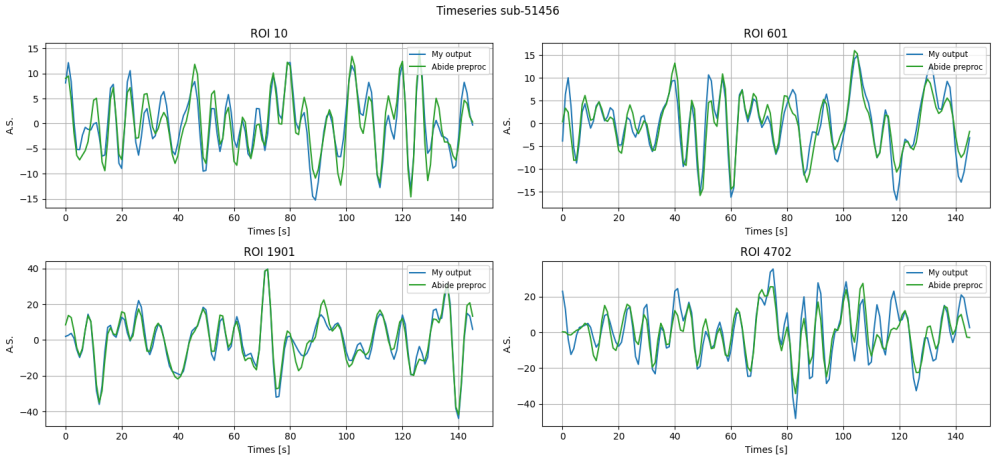


Figure 6.3: *Comparison of 4 timeseries between ABIDE-preprocessed and the ones obtained by our implementation of C-PAC. The four plots show four randomly chosen ROIs belonging to patient 51456 from ABIDE I. They corresponding to ROI: 10, 601, 1901, 4702 which are the corrisponding lables on the Harvard-Oxford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

Chapter 7

Connectivity measures

As described in chapter 6.1 if we choose an atlas on fMRI data, for brain parcellation, we can 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, by pairwise comparing timeseries and extracting a coefficient representative of 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 would not carry any information. Accoridng 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 combination each one expressed by a correlation coefficient.

In the following section we discuss two different approaches to extract a correlation coefficients 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, so that, if a linear relation lies between the two sets, a high absolute value of Pearson coefficient indicates that the two series are highly (positively or negatively) correlated. [22]

Given two series x and y, of length n correlation can be easily computed by

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

For each patient, Pearson correlation coefficient was computed for all timeseries pairs. Before further analysis, a common way to proceed is to transform each coefficient with Fisher z-transformation.[11] The reason behind this common operation appears clear when we are dealing with highly correlated variables. When Pearson correlation distribution results in an 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.[38]

Thus, this transformation allows us 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 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. The Fisher transformation is defined as:

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

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

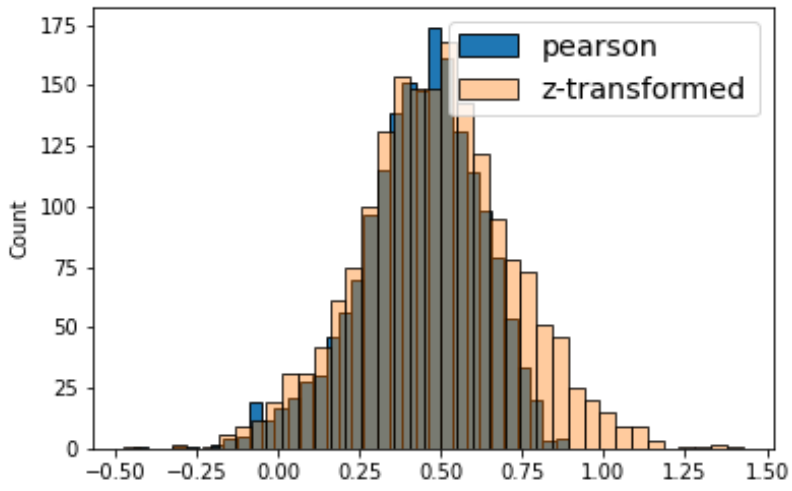
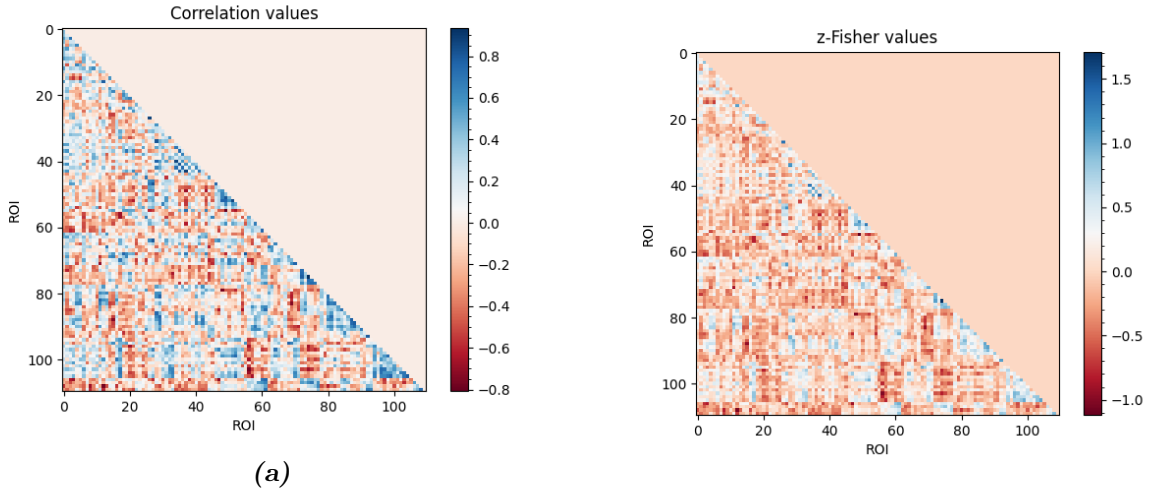


Figure 7.1: *Distribution of feature 3: Pearson correlation values in blue and Fisher transformed values in light orange*

7.2 Wavelet analysis

A different approach measure a correlation between two timeseries is by making use of wavelet analysis [39] [40]. Wavelet transformation is a mathematical tool for analyzing time series or images,



(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* $(-\infty, \infty)$

and provides a comprehensive way for investigating the bivariate relationship between 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 us to expand a periodic function f(t) into a series, an ideally infinite sum of weighted sines and cosines with different frequencies:

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

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 don't 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 time or space 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.

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

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

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 we can denote as $\tilde{x_k}$

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

where N is the total number of points of the timeseries x(t). 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 is 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 us to analyze non stationary signals and obtain information both on time (or space) and frequency domain. Since we are working on signal 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. These functions are called *windowing function* and are 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,

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

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 type of analysis, by simply shifting the windowing function throughout the entire timeseries x(t), it is a that differentiates the most the different analysis. 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 analysis, according to the convolution term employed and the effect of parameter a:

1. Windowed Fourier Transform (WFT): $\psi_{ab}(t) = e^{it/a}\phi(t-b)$ 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): $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$ 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.

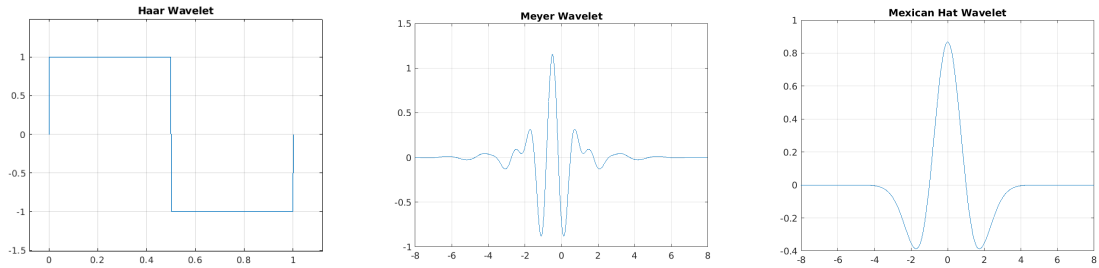
In Wavelet Transform, the equivalent to the window function in WFT is a wavelet function ψ.

Wavelet

A wavelet, literally "small wave "is a wave function limited both in space and period: begins at zero, grows and decrease 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: have zero mean (it must be oscillating) and unitary squared norm [41].

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



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

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

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) [42], 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. For each wavelet, there's 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 this wavelet the trade-off between time and frequency resolution can be controlled by the choice of ω_0.[43]. We choose to run our analysis with a value of $\omega_0 = 6$, since it provides a good balance between the two resolutions [44] and besides, this value gives the best ratio between Fourier Period and the scale parameter a during a Transform, equal to $\lambda = 1.03$. 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.

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

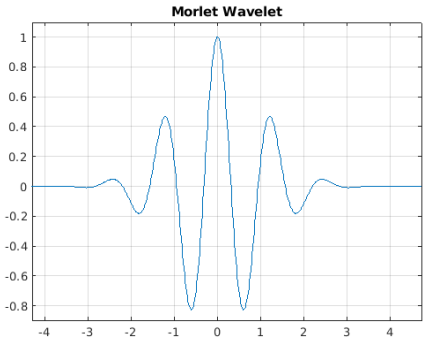


Figure 7.4: *Morlet wavelet*

Two main type of analysis can be performed using wavelets: continuous wavelet transform (CWT) and discrete wavelet transform (DWT) and the CWT is what we employed for our analysis.

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

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 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.

Formally, a wavelet transform, can be described as a mapping from $L^2(\Re) \to L^2(\Re^2)$.

To computationally implement CWT, a discretized version was implemented on MATLAB tools Wavelet toolbox, but it should not be confused with the Discrete Wavelet Transform [2], a similar type of analysis but with some important differences. Metto questa parte che parla delle differenze in un box? per separarlo dal resto del testo. 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 $= 2^{j/\nu}$ ([3]) [45] [42] where ν, j are integers. 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.

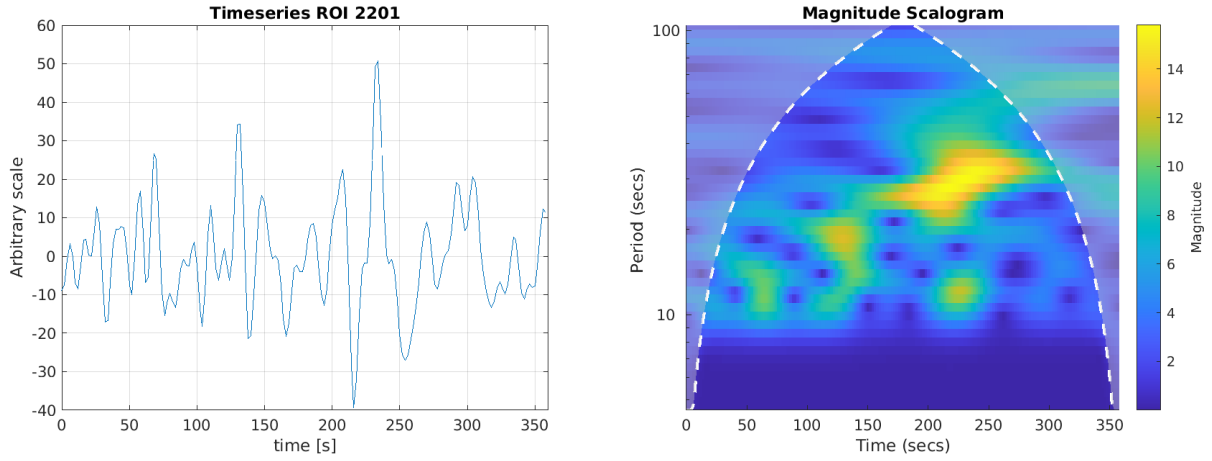
In our analysis we are going to use DTW implemented in MATLAB with the default parameters of 12 voices per octave.

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 using methods such as zero padding which assumes that the signal is zero outside its original support, or symmetrization which extend 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

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

[3] This way, 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 7.5a) of ROI 2201: right angular gyrus of patient 51056 from ABIDE I, and its corresponding Continuous Wavelet Transform (fig 7.5b). x axis corresponds to time points and y axis the periods 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.*

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 the 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 this analysis 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:

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

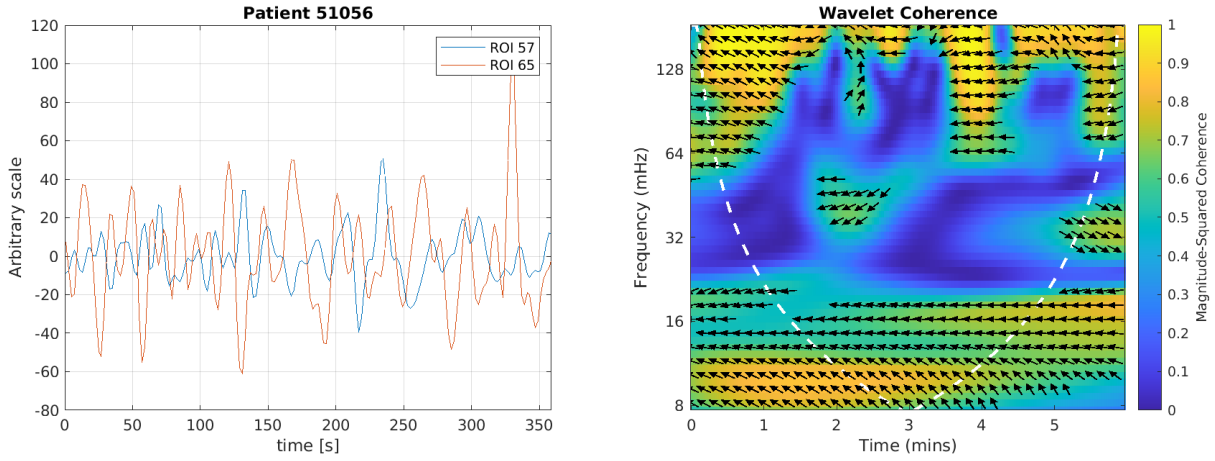
where the * represents the conjugate transpose. Similarly, the **Cross Wavelet Transform (XWT)**, sometimes also called *cross wavelet spectrum*, of two time series X and Y is defined as

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

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 possible to compute the the complex argument

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

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 $[-\pi,\pi]$.



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

Figure 7.6: *Plot of two timeseries from patient 51056 in figure 7.6a and their relative wavelet coherence scalogram. 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. The color represents the magnitude of the cross-power coefficients and the relative phase shift is represented as arrows pointing to the right is phase shift is zero ad to the left is is 180 degrees. Starting from 0 degrees arrows rotate counterclockwise.*

The wavelet coherence $R^2(a,b)$ is finally defined as reported in equation 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.

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

In equation 7.15 S is a smoothing operator, both in frequency and time, defined as [46] [44]

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

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 [47].

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.

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.

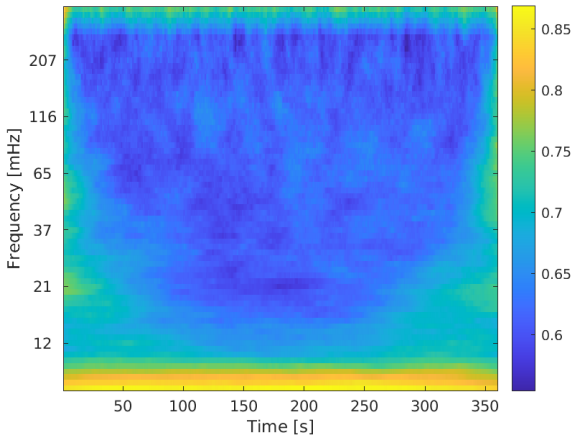


Figure 7.7: *Noise matrix obtained by collecting the $95-th$ percentile of the distribution of each entry of the 1000 noise matrices stack. It determines the threshold to assess significance of the coefficients of the wavelet coherence matrix computed for two timeseries*

The theoretical procedure [44] [48] [49] 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) of the two timeseries under examination. Then, for each pair of red-noise timeseries, we should compute the wavelet coherence matrix.

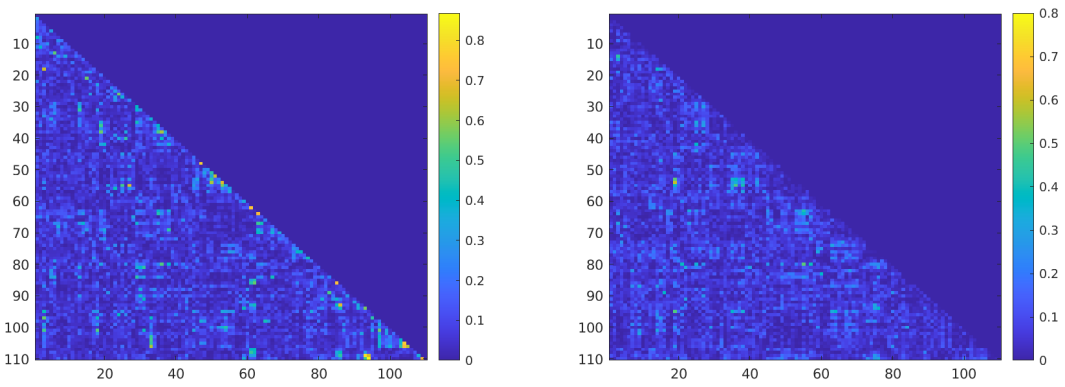
However, since AR1 coefficients have little impact on the significance level [44], we choose to generate a single sample of 1000 pairs of red noise and for each pair we computed the wavelet coherence matrix. For each pair of noise, the corresponding wavelet coherence matrix from equation 7.14 is stacked to obtain a $A \times B \times x \times 1000$ 3D noise matrix to extract the distribution of the entries. For each entry of a row 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 matrix visualized in figure 7.7

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. 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. [48] This time of in-phase coefficient is defined as:

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

where the indices i and j refer to the two timeseries i and j; 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:



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

Figure 7.8: *Correlation matrix created with wavelet coefficients of in-phase and off-phase percentage. Patient 51056, ABIDE I*

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

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

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

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

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 equation 7.16 both coefficients in-phase and anti-phase matrix coefficients were computed. The correlation coefficients matrix for each subject, was then used as input to the neural networks. 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

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 *scanner effects*, related to differences in the acquisition system such as the scanner model or the acquisition parameters. When analyzing data either with statistical analysis 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 inhomogeneity of data because of scanner effects, 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 kind of datasets such as genes microarray data, diffusion tensor imaging or structural MRI [50] [51] [52].

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 [50], for gene expression studies to compute batch effect corrections, later implemented by Fortin et al [53] for the harmonization of cortical MRI volumes, and finally in its current improved version, developed by Pomponio et al. [54] 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 and uses empirical Bayes estimate to improve the estimation of the site-related parameters for small-sized datasets [53]. 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 assumes that a feature can be modeled as a linear combination of the biological variables plus the

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

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 entry 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

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

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, $\hat{\sigma}_f^2 = \frac{1}{n}\sum_{it}\left(y_{ijf} - \hat{\alpha}_f - X\hat{\beta}_f - \hat{\gamma}_i f\right)^2$ being n the total number of patients.

   The standardized data are then calculated by equation 8.3

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

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 $\hat{\gamma}_{if}$ $\hat{\delta}^2_{if}$, making the model able to deal with small-sized dataset as well.

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

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

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

$$y^{ComBat}_{ijf} = \frac{\hat{\sigma}_f}{\delta^*_{jf}}(Z_{ijf} - \gamma^*_{jf}) + \hat{\alpha}_f + X\hat{\beta}_f \tag{8.4}$$

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. [54] is an improved version of it. It differ in the modeling of the biologically covariates, which in the formulas above are expressed by the terms $\hat{\alpha}_f + X\hat{\beta}_f$ which is 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

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

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, if we specify a covariate as non-linear, it will be treated using thin plate regression splines to find the right smoothing function (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 on the network proposed by Ganin, Ustinova et al [55]. Their work addresses the issue of working with two different datasets. They call them *source* and *target* dataset, where each dataset contains data following a distribution which is different between the two datasets. 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 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 of 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 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 represent 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.

A 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 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 be now 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 was accomplished by embedding the domain regressor G_d into the neural network consisting on G_f and G_y. The resulting neural network will include two branches: one for label classification and the other with the domain regressor. They linked these parts using a *gradient reversal layer* and exploited a classic stochastic gradient descent procedure to update weights. A gradient reversal layer allow 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 backpropagation, 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.

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

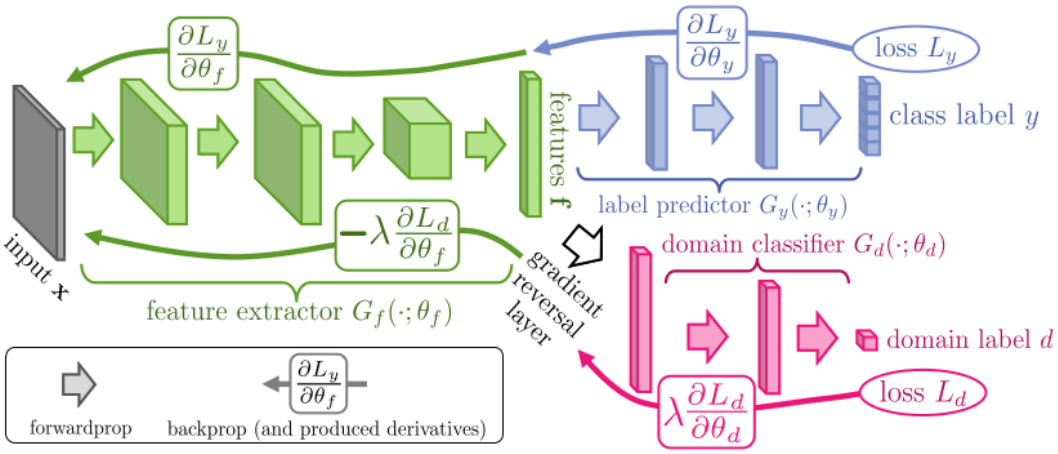


Figure 9.1: *A schematic representation of a Domain Adversarial Neural Network, from the presentation article [55]. The network mainly consists of three parts: a feature extractor, which takes inputs and creates an internal representation of data. From it the network is splitted into two branches: a 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.*

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).

Chapter 10

Machine learning model explanation with SHAP

Figure 10.1: *Graphic representation of how a deep model works, it typically acts as a black box, taking some data as input, such as age, sex, blood pressure, and uses them to produce an output which differs form a reference value. 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*

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 try to understand its deep mechanism and the reasons why a model returns a certain output, related to an input sample like a vector of features or an image. As it is shown in figure 10.1, just as 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, we are able to quantify the contribution of each feature of our input data and assess what and why are the most important features and how much they contributed to the final outcome.

The idea behind SHAP is to use Shapley values (see chapter 4) to explain every single feature contribution in our 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 our input data. For a single feature, its Shapley value is defined as the average marginal contribution of that feature across all possible coalitions.

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

There are two main classes of explainatory algorithms called local and global methods In general, the objective of a local explainatory model can be defined as the attempt to explain an output f(x) after a single instance x that in our case 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 dataset and extract 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 they are included. They 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' [56]. Local methods try to obtain a model g that faithfully approximate f, and that, if $g(x') \approx f(x)\text{if}x = h_x(x')$ This simplified model g' should also be a linear combination of binary variables of x' [57]:

$$g(x') = \phi_0 + \sum_{i=0}^{F'} \phi_i x'_i \tag{10.1}$$

Several methods before SHAP were created that satisfy this condition such as LIME [56] or DeepLIFT [58] but they lacked additional properties that SHAP has:

1. **Local accuracy**:

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

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

2. **Missingness**:

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

   Practically if a feature is equal to zero, it has no attributed impact on the model outcome

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

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

   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 [59] 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 an 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 model we are working on. Some algorithms such as TreeSHAP, are able to compute the exact Shapley coefficients. Others just compute an aproximate Shapley value. Algorithms we are going to focus on are KernelSHAP which is a model agnostic explanation method, and DeepSHAP which is optimized to work faster on deep models by making use of the knowledge of the structure of a neural network. Each one of these algorithms is built upon previous algorithms like LIME or DeepLIFT. In the following section we are going to explain some important features from LIME that were employed in KernelSHAP, and from DeepLIFT employed for DeepSHAP.

10.1 LIME and KernelSHAP

LIME (Local Interpretable Model-agnostic Explanations) is a local, model-agnostic, interpretability model, made to explain the prediction of any classifier in a faithful, but only local way. This means that it can accurately explain a single prediction but it is not the most suitable to generalize to many of them [56].

Since KernelSHAP is built upon LIME algorithm, we briefly discuss an important strategy implemented by LIME to avoid the computation of all the permutations.

Starting from a single instance $x \in \Re^n$, LIME creates a perturbed dataset X and evaluates the model over this dataset. Perturbed dataset is created by making use of several binary vectors containing just zeros and ones. In fact, it randomly creates different binary vectors of the same lenght of x, but randomly set 0 and 1 as its entries. From this binary dataset it is possible to come back to the space of features value, recalling that "1 "means that we are including the corresponding feature from x, while "0 "is associated to the missingness of the corresponding feature. LIME replaces missing features in different ways according to the type of data it is dealing with (images, text data, tabular data). In the case of tabular data, LIME repleaces missing values by randomly sampling a value from the distribution of that feature from the training data [60].

KernelSHAP works in a similar way, but introduces a background dataset from which missing values are randomly picked and replaced in the perturbed vector.

LIME uses the synthetic dataset just created, and the model is evaluated on each vector. These outputs are subsequently used to accomplish a minimization task to finally find the importance coefficients. Minimization is performed in a regression, but before it, each output value is weighted according to the distance of the synthetic vector with the original vector x by using coefficients π_x defined as

$$\pi_x = \frac{p-1}{\binom{p}{|z'|} \cdot |z'|(p-|z'|)} \tag{10.5}$$

Where we indicated as p the total number of features in the original feature vector x anc with $|z'|$ the number of element in that subset, and cosequently $(p-|z'|)$ is the number of features not

included in the subset.

To perform a fit, the loss function to minimize is in the form $L(f, g, \pi_x) = \sum_{z \in Z}[f(h(z)) - g(z')]^2 \ \pi_x$ where $g(z')$ is the simplified function expressed in equation 10.1, given by a combination of coefficients ϕ_i that are the parameters of the fit.

LIME is regarded as local method, because it explains the prediction of a black box model by making use of a local model: in this case a regression, which is an interpretable model computed in the neighborhood of the instance we want to explain. KernelSHAP implementation was strongly influenced by the LIME algorithm but,thanks to the introduction of Shapley values, it also comes with theoretical guarantees about consistency and local accuracy, from game theory.

As mentioned above, KernelSHAP is a model-agnostic algorithm, in the sense that be easily used with every kind of model, but SHAP includes some model-specifics algorithms that make use of a previous general knowledge of a model structure to optimize the explainer performances and speed up the process. DeepSHAP is one of these model-specific algorithms and it is what we are going to use in our analysis.

10.2 DeepLIFT and DeepSHAP

DeepSHAP alghorithm works with some similar implementation of KernelSHAP, making use of a background dataset to simulate missing values, but it is also optimized to perform better on deep models by taking advantage of the idea behind the DeepLIFT algorithm. It can be considered as an enhanced version of DeepLIFT, thanks to the introduction of the Shapley values, rather than the coefficients employed in DeepLIFT called *DeepLIFT multipliers* [58]. The strategy implemented by DeepLIFT to explain features is to compute the difference between the output of a deep learning model obtained with the "true" data and a reference value computed as the output of our model with some reference data. The choice of the reference depends on what kind of data we are working on: images or genomic data, for example. For genomic the most common way to produce reference output in DeepLIFT is to shuffle some training data, evaluate the model on them and average across all the scores. With the implementation of DeepSHAP, this reference value is chosen from the backgound dataset and represents an uninformative value for a feature.

As mentioned before, deepLIFT is also shaped to perform on deep learning algorithm computing its feature importance values called DeepLIFT multiplier during the process of backpropagation. DeepLIFT attributes to each feature x_i a value $C_{\Delta x_i, \Delta y}$, that represents the effect of that input being set to a reference value rather than its actual value.

In a few words the idea behind DeepLIFT (and DeepSHAP) can be summarised as: let t represent the output of some inner neurons and let $x_1, x_2, ..., x_n$ be some preceding neurons necessary to compute t, if we label t_0 the reference output, we can compute the value $\Delta t = t - t_0$ and use them to define the DeepLIFT coefficients $C_{\Delta x_i \Delta t}$ associated with Δx_i which is the difference between the true feature and the reference value for that feature. DeepLIFT coefficients are chosen to follow the property called *summation to delta* property

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t$$

For a given input neuron x with difference from reference Δx and a target neuron t, is defined the *DeepLIFT multiplier*

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x}$$

which represents the contribution of Δx to Δt. Once computed the multipliers of each neuron in a layer it is possible to compute the multiplier of any target neuron during the backpropagation. For a more detailed explanation of how DeepLIFT works we refer to its presentation article [58].

DeepSHAP exploits the same principles as DeepLIFT, and combines Shapley values for smaller components of a network to compute values for the whole network. It accomplish this by recursively passing Deep LIFT multipliers now defined in terms of Shapley values during backpropagation. To not weight down this discussion, we explain the main idea of how this procedure works by making use of a simplified neural network. We suppose to have a single input vector x consisting only on two features: $x = (x_1, x_2)$ 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.

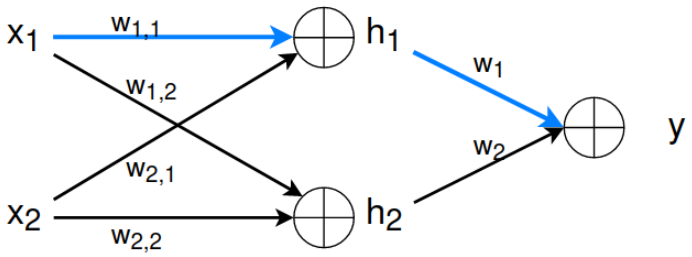


Figure 10.2: *Representation of a linear network and important quantities involved during forward propagation.*

$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 \qquad y = w_1h_1 + w_2h_2$$

Giving a look at the model, we can retrieve the Shapley values for each node by summing all the contribution 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 x_i and the background value b_{x_i}. In practice, if we focus on the colored blue line, we have that the contribution from this path to $\phi(h_1)$ is given by

$$\phi(h_1) = w_1(f_{h_1} - b_{h_1})$$

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

$$\phi(x_1) = (f_{x_1} - b_{x_1})w_{1,1}w_1 = (f_{x_1} - b_{x_1})w_{1,1}\frac{\phi(h_1)}{f_{h_1} - b_{h_1}} \tag{10.6}$$

adding all contribution from all the possible paths. In figure 10.2 are represented for a greater clarity only two paths, deriving from the two input features $x_1 textand x_2$, 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.

10.3 Shap values as feature importance

To express the significance of a feature, we can rely on the relation between feature importance in a qualitative way and Shapley value: features with high absolute Shapley value are important. 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 used to calculate these values

$$I_f = \frac{1}{n} \sum_{i=1}^{n} \|\phi_i^{(f)}\| \tag{10.7}$$

SHAP is implemented as a free Python package with MIT license, developed and maintained by Scott Lundberg [2]. This package includes different classes such as KernelExplainer, which is the class name of KernelSHAP mentioned in section 10.1, TreeExplainer tuned to perform rapidly on tree and forest-like models, linearExplainer which deals with linear model and is able to compute the exact shapley values and not just an approximation, and the class we are using: DeepExplainer, suitable to explain deep learning models and to compute an approximate value of Shapley values.

[2] https://github.com/slundberg/shap

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 obtained 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 (DNN). We compared different harmonization pipelines implemented during the classification process.

To this end, before running any classification procedure, in chapter 12 we show some results obtained through the process of harmonization 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.

In chapter 13 we present the structure of the deep neural network employed for classification and how we reached this structures.

In chapter 14 we present results obtained from classification of data using deep neural networks. We follow different classification and harmonization pipelines and we compare the performances between a DNN and a random forest classifier. We seek to find the best strategy to obtain the best separability between controls and ASD data.

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

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

11.1 Coefficients selection

All the different classification and harmonization pipelines explained in this chapter are carried out using different correlation coefficients, created following the analysis explained in chapter 7, or obtained through a combination of them. In details, we used the following coefficients as input to DNNs 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 an 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. Coefficient selection: select whether to use Pearson-based correlation coefficients, or wavelet-based correlation coefficients among the ones listed in section 11.1;

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

3. Collect the raw data left after the previous steps;

4. Classification of controls/ASD can be fulfilled by a deep neural network, a random forest classifier, or the domain-adversarial neural network;

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

6. For each pipeline, the final classification score is reported as the mean and standard deviation 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/

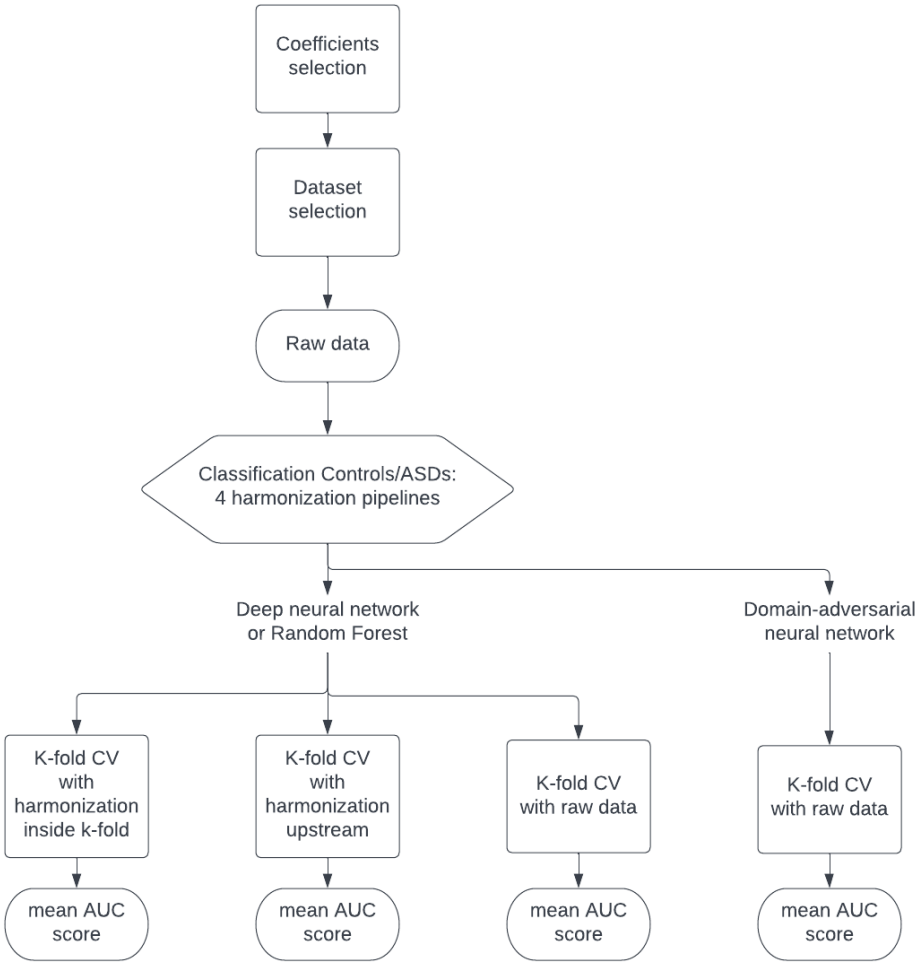


Figure 11.1: *Flowchart of the harmonization and classification pipelines carried out in this work.*

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/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/ASD using raw (not harmonized) data;

4. Classification controls/ASD using the domain adversarial neural network, giving as input raw data. Using this network, an inner 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 articles where harmonization of data is performed, evaluate model performances using a leave-one-site-out Cross Validation [11] [61]. 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 Dataset selection

With reference to the "Dataset seleciton" 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 data but only ASD patients, and without control data, we are not able to perform a proper harmonization on these sites.

We choose to run analysis considering only male subjects, aged between 5 and 40 years old. This allows to reduce variability due to sex

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/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 [11]. 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.

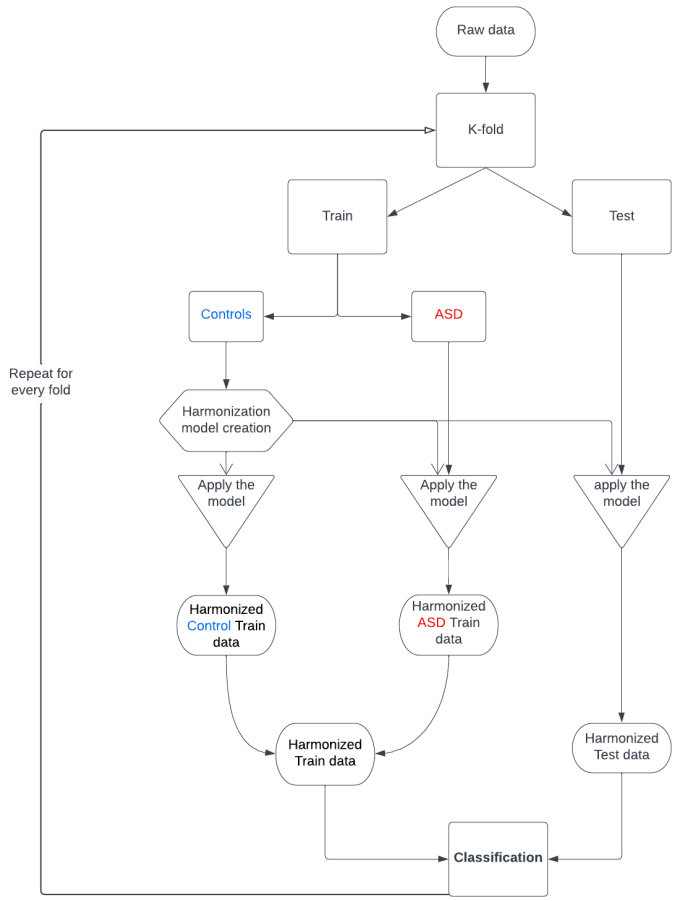


Figure 11.2: *Flowchart for the implementation of harmonization inside a k-fold CV:*

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 and thresholds.*

11.4 Dimensionality reduction of data

An important issue related to data we are working with concerns their dimensionality. We are dealing with data lying in an 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 3.6, hence reducing the dimensionality of the problem.

When applying PCA to a dataset we seek to explain as much variance as possible, without losing

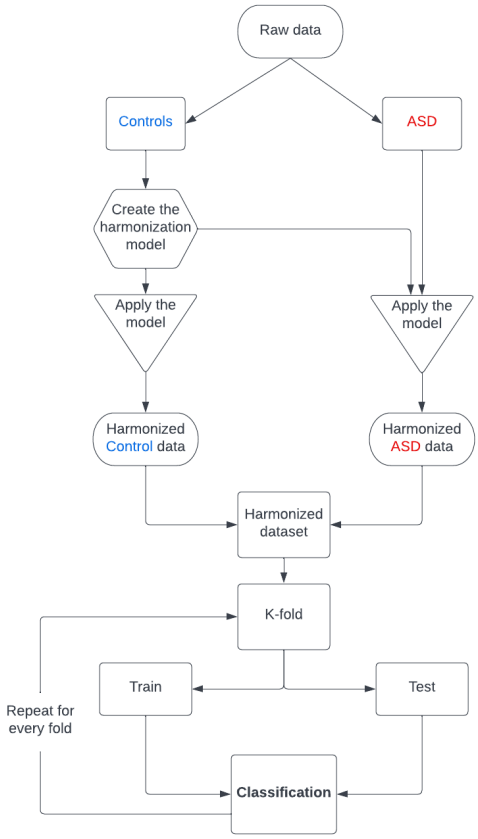


Figure 11.3: *Flowchart of the upstream implementation of harmonization, before the k-fold split*

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 3.6, the maximum number of PCs 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 just the train dataset 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 analysis 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. Results are shown in section 14.3

Chapter 12

Harmonization - results

As described in chapter 8, 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.

Harmonization procedure was tested on the dataset created by using patients from ABIDE I + ABIDE II after the cuts on sex and age discussed in section 11. From these patients we tested harmonization using Pearson correlation coefficients and wavelet coefficients of in phase time percentage (w_in). We run this analysis to have a quantitative and a visual representation of how features are modified after being processed with an harmonization pipeline.

To harmonize data we used NeuroHarmonize package[1] for Pyhton, developed by Pomponio et al. [54] 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 csv file 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 harmonization procedure on 1470 male patients coming from ABIDE I and ABIDE II of which 737 are controls and 733 cases, choosing as covariates to preserve 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 control. 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.

To test the effectiveness of this harmonization procedure, we performed a binary classification of data: site vs site, for each pair of sites. Classification is carried out using a random forest classifier and we tested its performances for site classification before and after the harmonization of data. In this analysis, we splitted the model into a train and a test subset: using only control subjects from the train dataset, we create 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. Thus, two random forest classifiers are trained: one using

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

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

raw data and the second 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 perform a proper harmonization with them, we excluded these sites from our analysis.

Figures 12.1 and 12.2 show the results of classification site vs site using two different correlation coefficients: 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 scores are reported in terms of AUC.

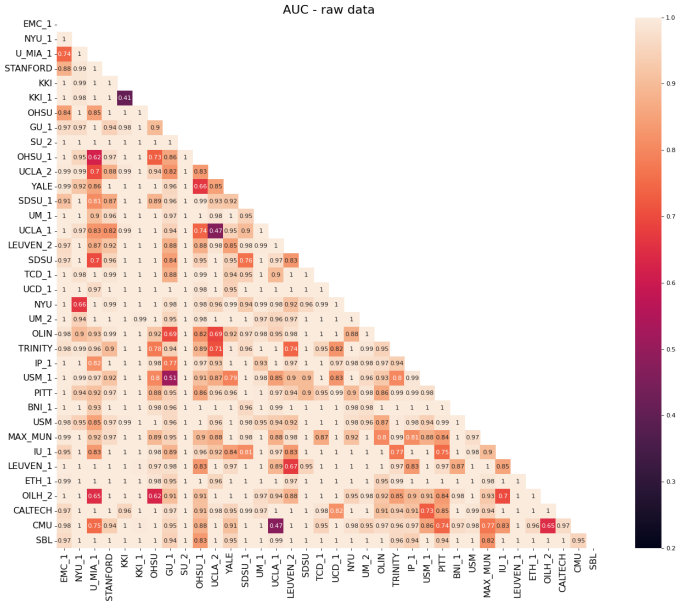
Specifically, fig 12.1a and 12.2a show the AUC score of the model trained on the raw, not harmonized dataset, while in figures 12.1b and 12.2b the results obtained with harmonized data are shown.

To have a visual representation of how much each feature is modified in respect of its sourced site, in fig 12.3 two features among the 5995 are shown as a function of sites: *feature 324* and *feature 2800*. This figure shows features from Pearson coefficients while in figure 12.4 are shown the same two features obtained with wavelet coefficients of in time percentage. To clarify the meaning of each feature: feature 324 represents the correlation between right and left inferior frontal gyrus, and feature 2800 the correlation between left precuneous cortex and the left inferior frontal gyrus. There is not a particular reason for choosing these two feature, they are just randomly drawn from the 5995 possible correlation coefficients. Their values are plotted as a box along the y axis and sites are indicated along the x-axis, it appears clear how the mean value of each feature for each site is shifted and features are stretched or shrinked to make them follow a more uniform distribution.

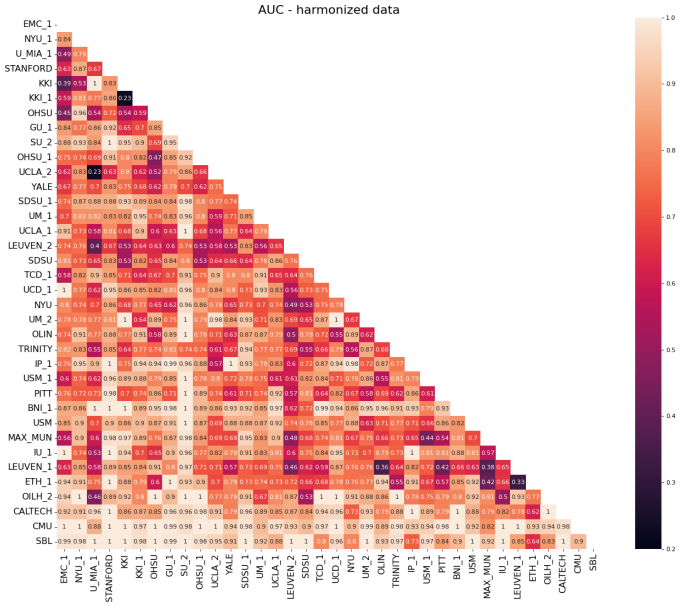
If we split these plots into control and ASD subjects to see the effect of harmonization separately on these subsets, we obtain figure 12.5 which shows feature 324 computed using Pearson coefficients and figure 12.5b with wavelet coefficients.

12.1 Comments on harmonization

Figure 12.1 and 12.2 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 (fig 12.1a and 12.2a) that the AUC value is mainly near to 1, which stands for exact ability to classify site_a vs site_b. With Pearson-based coefficients this value decreases when the classification is performed using harmonized data (figure 12.1b). This stands as a confirm that the harmonization procedure removes (or, at least, reduces) site-related features making them less recognizable among data. This same effectiveness, though, is not visible with wavelet-based coefficients. It appears clear from figure 12.2b) 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 figure 12.4 and confront it with figure 12.3 we can notice the presence of a greater number of outliers between 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 figures 12.5a and 12.5b both for Pearson and for wavelet coefficients, ASD data keep a site-related trend. The effect of both this trend and the outliers is what makes sites recognizable even after the harmonization.

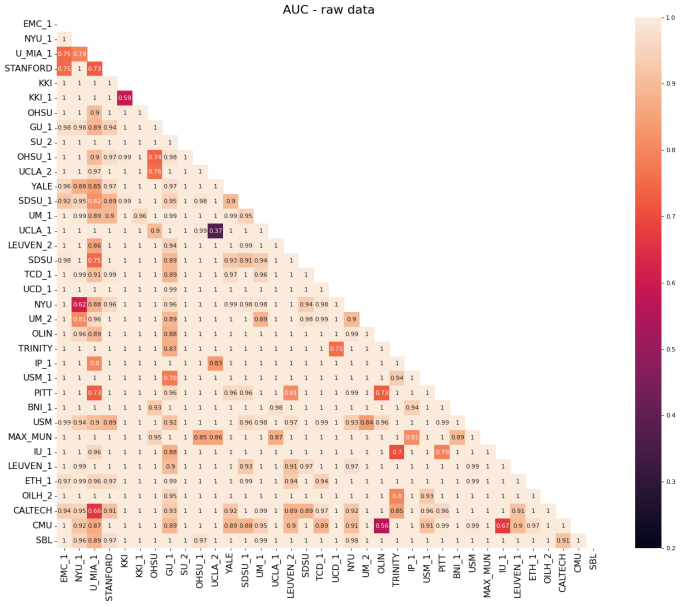


(a) *Heatmap with AUC score of a binary classification site vs site with raw data*

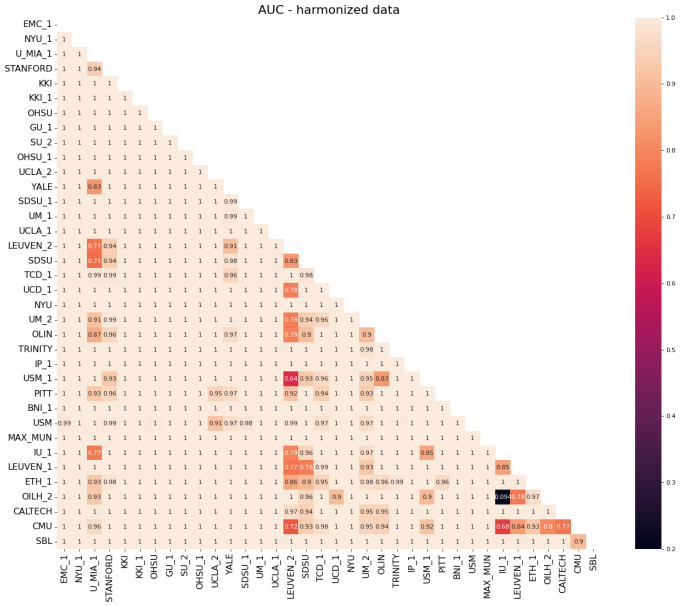


(b) *Heatmap with AUC score of a binary classification site vs site with harmonized data*

Figure 12.1: *Comparison of two heatmaps with AUC score of binary classification site vs site with* ***Pearson****-based correlation coefficients of raw (fig 12.1a) and harmonized data (fig 12.1b) classified using a Random Forest Classifier*

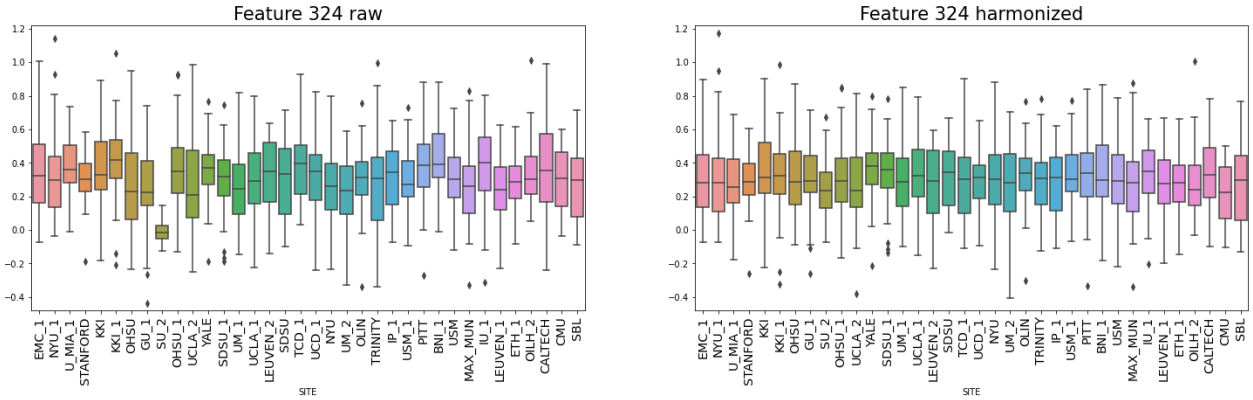


(a) *Heatmap with AUC score of a binary classification site vs site with raw data*

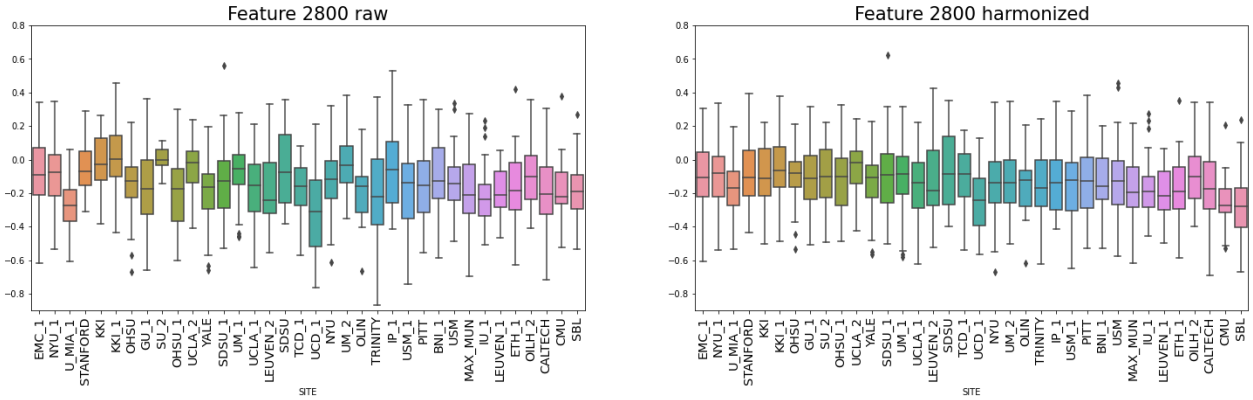


(b) *Heatmap with AUC score of a binary classification site vs site with harmonized data*

Figure 12.2: *Comparison of two heatmaps with AUC score of binary classification site vs site of* ***wavelet****-based correlation coefficients of raw (fig 12.2a) and harmonized data (fig 12.2b) classified using a Random Forest Classifier*

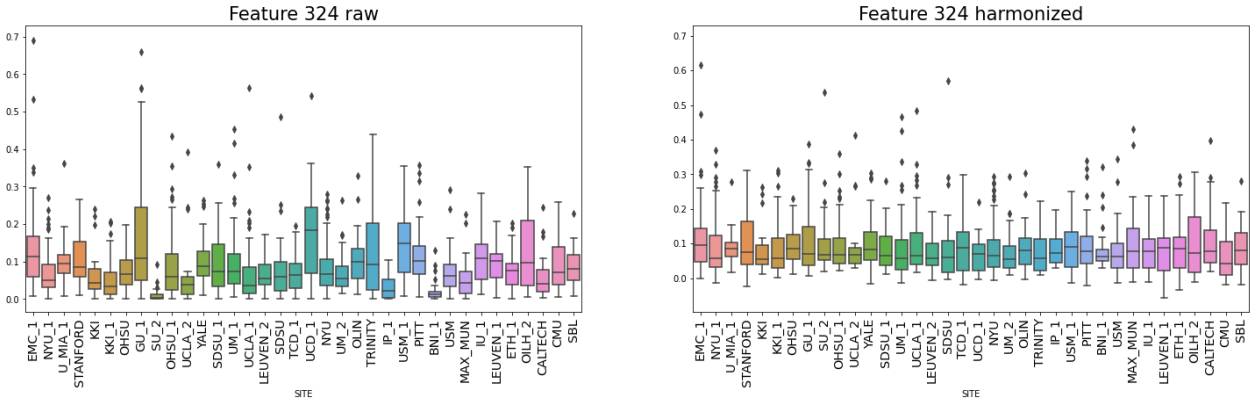


(a)

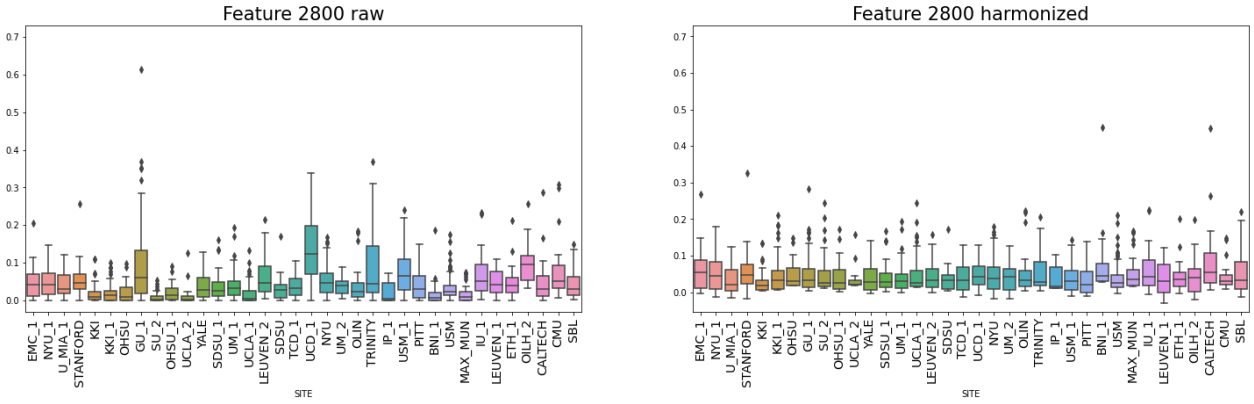


(b)

Figure 12.3: *Pearson-based connectivity coefficients per site for features 324 (fig 12.3a) and 2800 (fig 12.3b) before and after harmonization*

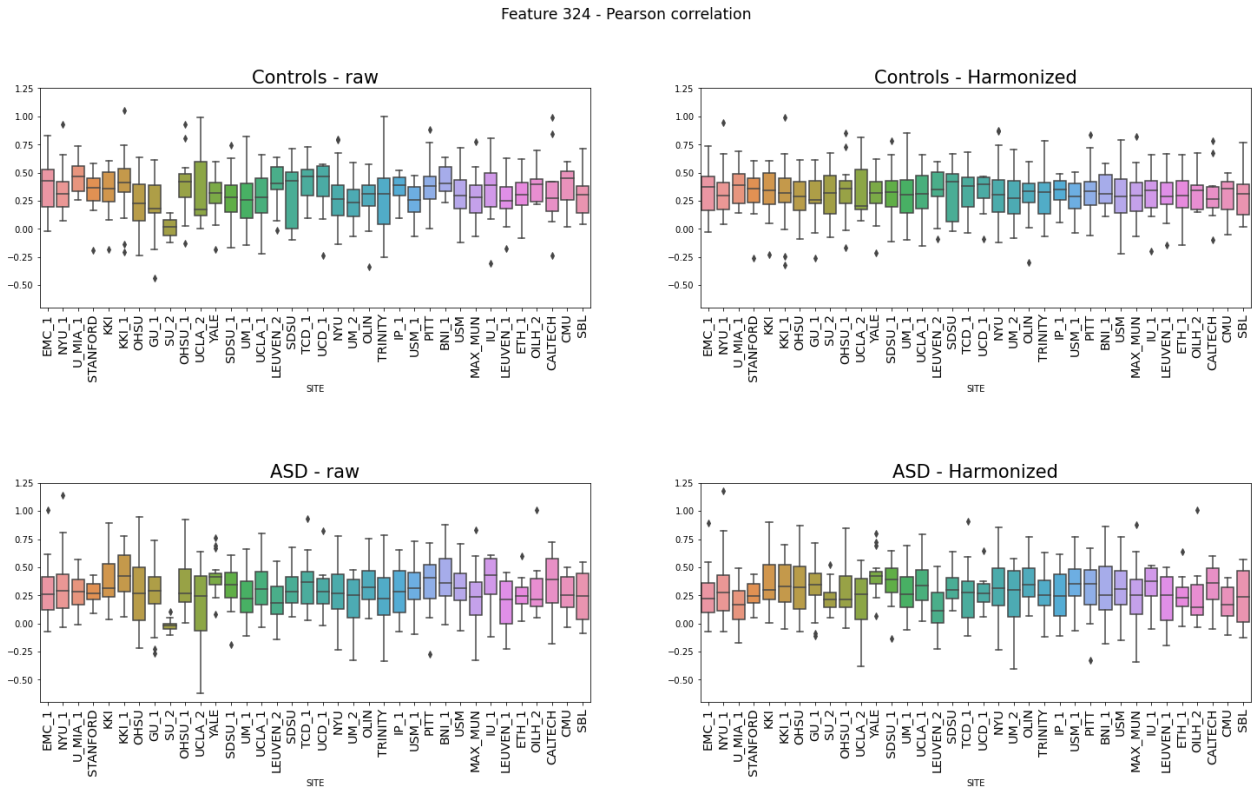


(a)

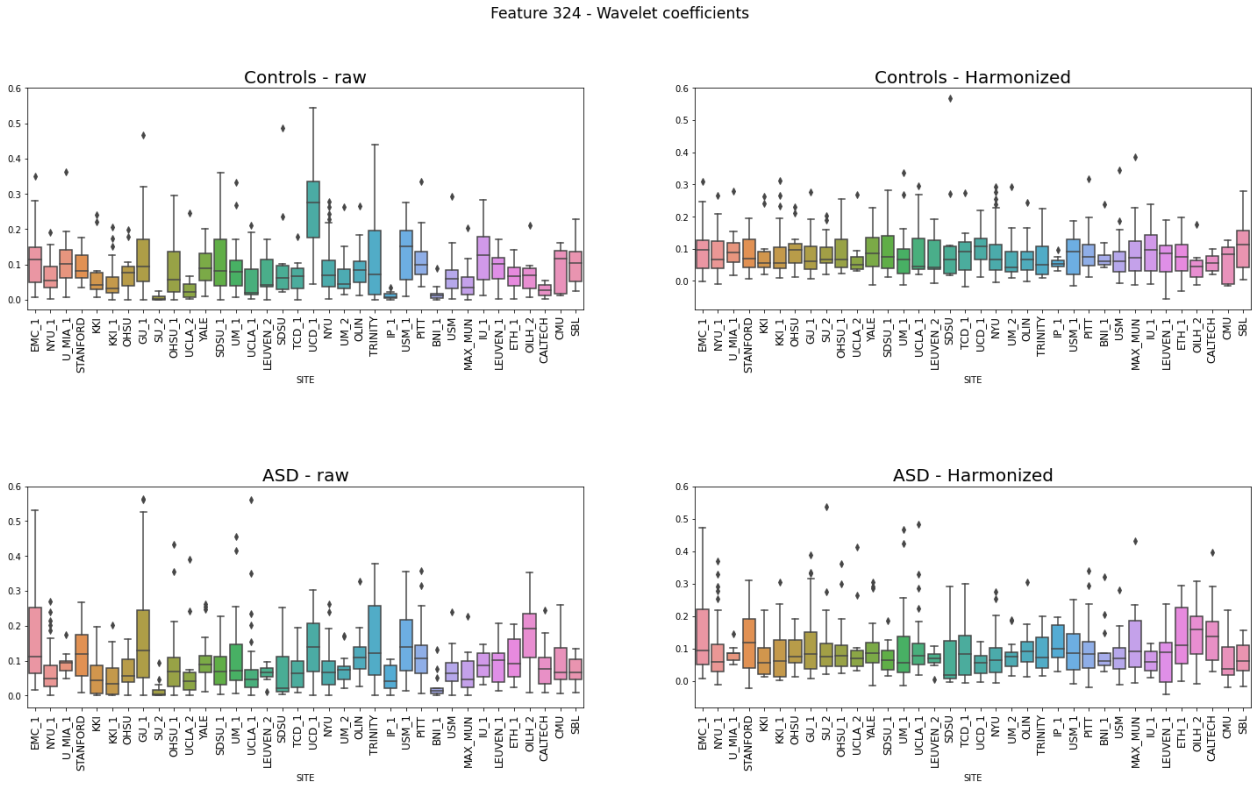


(b)

Figure 12.4: *wavelet-based connectivity coefficients of in-phase percentage, for features 324 (fig 12.4a) and 2800 (fig 12.4b) before and after harmonization*



(a) *Feature 324 with Pearson correlation coefficients*



(b) *Feature 324 with wavelet-based correlation coefficients*

Figure 12.5: *Pearson correlation and wavelet coefficients for features 324 before and after harmonization with a separated plot for controls and ASD*

Both for Pearson-based coefficients in figure 12.3 and for wavelet-based coefficients in figure 12.4 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 figures 12.3 and 12.4. However, harmonization acts differently on controls and ASD data. In figure 12.5a and 12.5b 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 little statistic in both cases. In a dataset of bigger 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 coefficients and wavelet-based coefficients assume a more regular trend. This is a visual confirm that harmonization effectively removes site-related, but if for Pearson coefficients this is effective to remove inter-site variability, for wavelet-based coefficients this method is not enough to remove site-related effect.

Chapter 13

Deep neural models

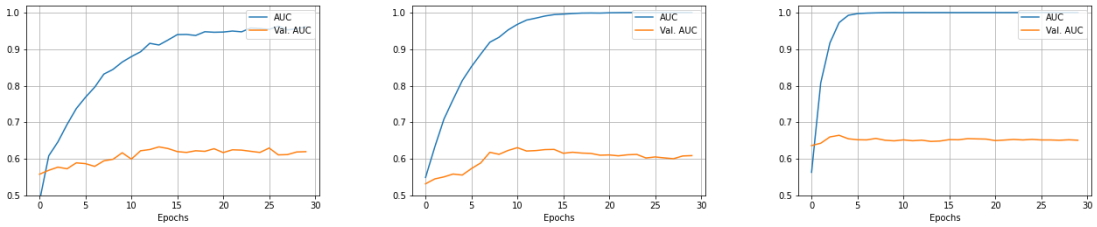
13.1 Deep neural network

In this work we performed a classification task, using a shallow neural network 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 coefficients to perform a control/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 figure 13.1 which shows the training and validation AUC curves for three simple models: the first one (fig 13.1a) is the trivial structure just mentioned and 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 increases. We continued adding neurons to each layer and tried different configurations. With a configuration of 8-8-1 (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 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 network reached a stable performance value. Different attempts and the related classification scores are reported in table 1 in Appendix. We decided to pick the configuration comprising the minimum number of neurons, beyond which 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 figure 13.2. This network consists of 2 hidden layers made up of 264, and 8 neurons respectively. Both layers are activated by a ReLU function and separated by a Batch-Normalization and a Dropout layer with a dropout chance of 30%. After the 8-neurons 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

[1] https://keras.io/



(a) *Train and validation AUC curve for a network with structure 3-2-1.*

(b) *Train and validation AUC curve for a network with structure 8-8-1.*

(c) *Train and validation AUC curve for a network with structure 64-8-1.*

Figure 13.1: *AUC learning curve corresponding to deep neural networks with different structures trained on the entire non-harmonized dataset consisting on ABIDE I + ABIDE II.*

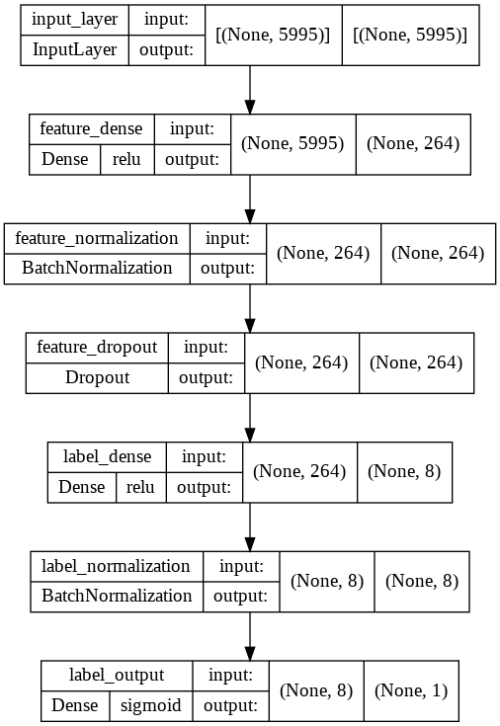


Figure 13.2: *Schematic structure of the deep neural network employed in this work for control subjects vs ASD patients classification. It follows a structure 264-8-1. Each box contains the layer name, layer task (Input, Dense, BatchNormalization etc..), activation function, input shape (number of neurons)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

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 Domain-adversarial neural network

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 losses composition
Our first attempt was to implement a model able to predict category label without any influence of site information. It consisted on two different branches, one for the control/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 in the article [62]. This combined loss is in the form:

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

Where L_1 is in our case the loss for the binary control/ASD classification, that we want to minimize (binary crossentropy), while L_2 is the loss for site classification: a categorical crossentropy we aim to maximize in order to avoid to learn any information related to sites. λ is a parameter empirically set, to control the contribution of the sites loss L_2 to the overall loss. This way, the 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 one created with different constraints on patients, the best score is obtained for a slightly different value of λ.

Adversarial model with 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/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 controls/ASD. 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/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 backpropagation. We empirically set this value to 0.3. The structure of this network is illustrated in figure 13.3

The final layer of the site-classification branch is activated by a softmax function, and the loss function employed is the categorical crossentropy, commonly used for multi-class classification tasks as explained in section 3.4.

To work with this loss function it is necessary to perfrom 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 non optimal 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 the 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 gradient reversal) and we noticed that they had more or less similar performances. However, we choose to carry out all the subsequent analysis 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 backpropagation, weights related to the site-classifier branch are updated in order to minimize the site loss. Conversely, as shown in equation 9.6, 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 [63], that compared the performances obtained with different implementation of adversarial network, 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.

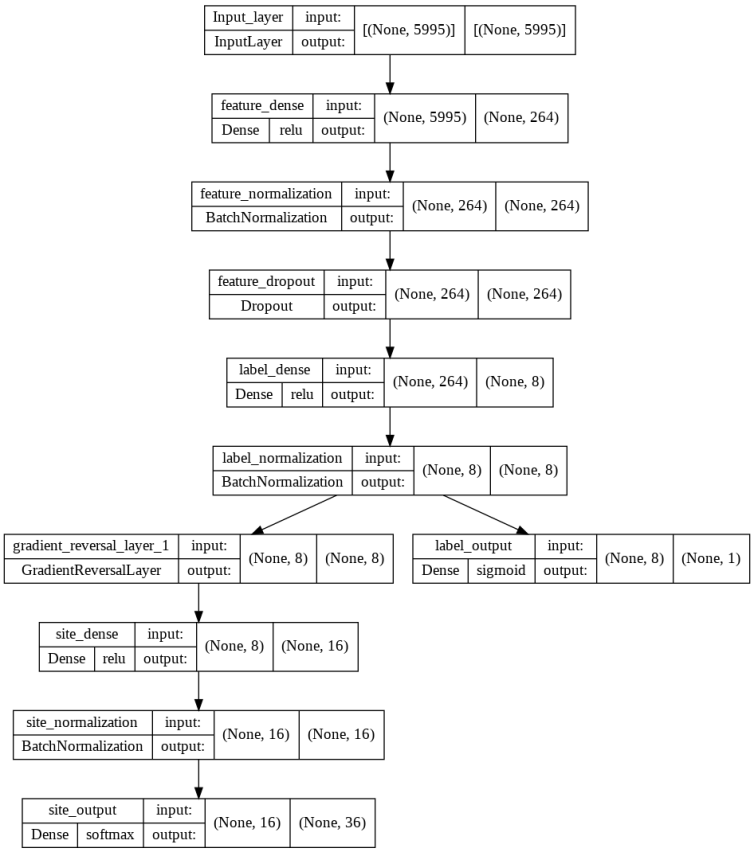


Figure 13.3: *Structure of the domain-adversarial neural network with two outputs: a single-neuron output for binary classification controls/ASD and a multi-class classification for site classification. Each box contains the layer name, and the related task (Input, Dense, BatchNormalization etc..), activation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

Chapter 14

Results of classification

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.

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

- **Harmon. k-fold** the pipeline where harmonization is implemented inside the k-fold CV and harmonization model is created only on control train data, and then applied to the rest of the dataset. The process is outlined in figure 11.2.
- **Harmon. upstream** the pipeline where 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, harmonization model is created using data of all controls as outlined in figure 11.3.
- **No harmon.** the pipeline where classification is carried out using raw data and no harmonization is applied.
- **Adversarial** 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 datasets created with patients belonging to the two different release of ABIDE dataset, as described in chapter 5. The cuts on covariates such as sex and ages as discussed in chapter 11. When from these subsets we select only patients who kept their eyes open, we expressly specify it as **eye = open**.

We eventually compared performances of the deep neural network, with a random forest classifier. We limited these comparison only on two dataset partition, following the three main harmonization pipelines of Harmon. k-fold, Harmon. upstream and No harmon.

14.1 Results with Pearson coefficients

The first classification analysis was made on Pearson correlation coefficients.

Results of classification are reported in terms of mean AUC across all the 5 folds. We listed in table 14.1 results obtained with our DNN model 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 is appropriate to use a deep neural model instead of a conventional machine learning algorithm. Comparison with random forest were performed on the whole dataset ABIDE I+II, and on ABIDE I with open eyes, 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-related coefficients 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 correlation coefficients, for dataset: 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 Results with wavelet coefficients

Following the same workflow as with Pearson-based coefficients, we run the classification and harmonization pipelines using wavelet-based coefficients. We used different wavelet coefficients as described in section 11.1. 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-related coefficients, we compared results obtained using a DNN with 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 dataset: 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 coefficients 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 elementwise subtraction in order to create w_diff.

w_diff is an effective way to create a coefficients that carry information in a similar way as Pearson coefficient do. They compress the information held by w_in and w_out into a single coefficient without the further increasing in dimensionality as we obtain with the creation of w_stack.

The best classification performances are obtained using the dataset ABIDE I with only open eyes. While using just ABIDE II dataset leads to the lowest classification scores. As we noticed from the results obtained with Pearson coefficients, 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 achieve on average similar performances.

14.3 Results with PCA

As mentioned in section 11 we chose to run PCA analysis using coefficients that gave the best classification results in the previous analysis. Comparing average scores obtained with Pearson-based and with wavelet-based coefficients, we can notice that with Pearson we achieved a better separability between controls/ASD. Furthermore, to not weight down this table, we limited our dataset choice to the whole ABIDE I+II and ABIDE I with open eyes. The latter is chosen because on it, we obtained slightly hiher AUC scores.

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

PCA explained variance [%]		
N 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

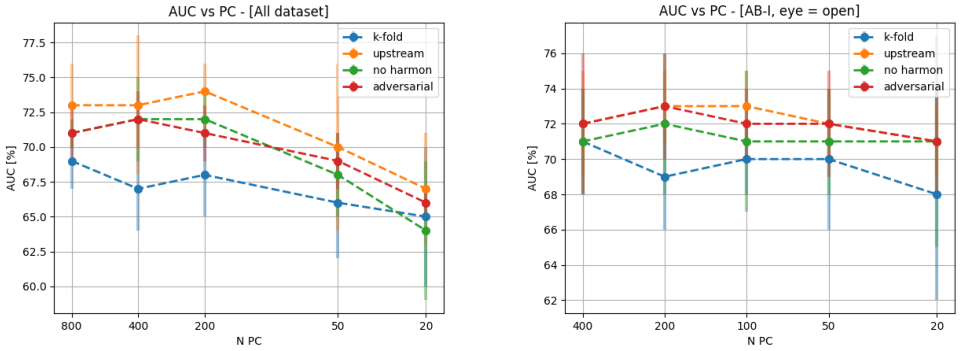
Table 14.5: *Explained variance [%] for different numbers of principal components, and for the whole dataset of ABIDE I+II and ita subset of ABIDE I with eye open.*

As explained in section 3.6 the number of pc 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 open eyes. Using this dataset, we have in fact, an amout of training data (from which PC are extracted) lower than 800 samples.

We followed the same pipelines as we did for Pearson and wavelet coefficients, following the four different harmonization procedures. Results obtained from this analysis are reported in table 14.6 and shown in figure 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 score obtained with the deep neural networks, using a decreasing number of principal components, following different harmonization procedures*



(a) *Classification on the entire dataset ABIDE I+II*

(b) *Classification on dataset ABIDE I - only open eyes*

Figure 14.1: *AUC scores obtained from control/ASD classification of data using decreasing numbers of principal components for Pearson-based coefficients. Results obtained with different analysis are represented with different colors: AUC scores with data harmonized inside k.fold are marked* ***blue****, AUC score with data harmonized upstream* ***orange****, AUC scores with raw data (not harmonized)* ***green*** *and AUC scores classification with the adversarial network* ***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, expecially with an higher number of principal components, classification performances are not affected by this reduction of dimensionality. The higher is the percentage of variance explained with PCA the higher classification performances are achieved for the both dataset ABIDE I + II and ABIDE I with open eyes.

At a first glance we can notice from figure 14.1 that reducing the number of PCs leads to a gradual reduction of classification performances. This trend is more evident using the whole datataset, while with ABIDE I only open eyes 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 an high classification AUC comparable to that obtained without PCA. In this analysis as well, following pipeline of upstream harmization, leads to an improvement of performances the suspected reason for this common trend are explained in section 14.4.

14.4 Discussion on classification results

Comparison between Pearson and wavelets

From results reported in section 14.1 and 14.2 we can observe that average scores obtained with Pearson correlation coefficients are systematically higher than wavelet-based ones. We can therefore assert that working with wavelet-based correlation coefficient obtained through processes described in section 7.2 does not bring many advantages to this type of analysis. A possible reason for this outcome is the loss of information that occurs during the extraction of these coefficients. To this regard an important role is played by the introduction of redundancy and randomness into this

analysis. Starting from two timeseries wavelet transform computes a 2D matrix for each timeseries and uses them 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 timeseries pair, 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 correlation.

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 a more direct and linear correlation calculus like with Pearson coefficients.

This lower scores is found both on wavelet coefficients of in-phase and counter-phase time percentage (w_in and w_out), and not even the combination of this two 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 through all the pipelines and over all the dataset is lower that the ones obtaines with Pearson.

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

Effects of a smaller and more regular dataset

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 every dataset (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 cortex and gyri areas, expecially those related to the processing of visual stimuli. An other variability introduced by patients with eyes closed, 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 patterns between controls and ASDs.

We can also notice that results on the dataset made by ABIDE I + II are mostly driven to the data belonging to ABIDE I. This is true either if we restrict our analysis on open eyes patients only or we do not. In fact, putting together the two dataset does not lead to a great improvement of scores than limiting just on ABIDE I. It is possible that this trend is due to two main factors: the greater number of ABIDE I patient with respect to those in ABIDE II left after all the cuts on covariates, and the presence of a greater biological variability of subjects in ABIDE II dataset.

Comparison between a deep neural network and a random forest classifier

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 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 [64]. This is true especially 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 structure and the best number of layers and neurons of the deep neural network.

Comparing the results obtained by the deep model with those obtained by a random forest, (table 14.2 and 14.4) it is possible to deduce that a deep model is more adapt to find characteristic patterns between controls and 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 paragraph below, this procedure leads to biased data and as a consequence results obtained with this pipeline are not reliable. We are still going to use Random Forest classifier in some of the remaining anslysis just as a comparison, even though we assert that a deep model is more suitable to understand differences between the two classes of interest.

Effect of different harmonization pipelines

An other common trend that stands out from all the analysis is the systematic improvement of AUC score when we use data harmonized with the pipeline of upstream harmonization (explained in section 11 and outlined on flowchart in figure 11.3). This improvement disappears when we implement harmonization inside the k-fold cross validation procedure (sketched in 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, with harmonization implemented inside the cross validation procedure, we run harmonization only after the division of the dataset into a train and test subsets. We suspect that these higher results are driven by a misleading way to proceed. Classification results are affected by a bias due to data leakage which occur if we implement harmonization upstream. It is also possible that some good results on other similar studies, obtained with harmonization implemented in this same way such as [61] may be due to a similar data leakage as we have in this analysis.

In this case we have data leakage when we harmonize the entire dataset before splitting it into train and test, becaus, when we use train data to train the model, they already contain information about 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 assert that the right way to implement harmonization 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 don't notice any particular improvements. As we seen in chapter 12 harmonization is an effective strategy to reduce site-related features, but it has no particular effect on classification controls/ASDs.

This is likely due to the weak information contained within our data. Distinction between healthy and ASD subjects based only on functional connectivity data is not a straigthforward task, and it is possible that these results are the best it is possible to achieve with these data. However harmonization procedure can become a useful tool to remove some noise from data and obtain a cleaner assessment of what feature are the most discriminative between controls and ASD. This theme is entirely addressed in chapter 15 talking about feature importance.

Results with adversarial

Looking at tables 14.1 and 14.3 we notice that the adversarial model does not bring significative advantages in classification scores. AUC score obtained with this network are on average the same as obtained with other harmonization pipelines (except for the upstream harmonization pipeline that as we stated before, 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 different papers [55], [63] or [62], adversarial networks are employed with only two domains, which can be considered as the equivalent of our sites. In our data we deal with 36 sites when we consider ABIDE I+II, or, best case scenario, about 15 sites when we reduce to ABIDE I or II with just open eyes. This way even if the site-predictor branch is encouraged to learn site-distinctive traits, it is not always able to distinguish between 36 sites, so this can cause confusion during backpropagation 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.

Effect of dimensionality reduction

The same alertness we paid for the implementation of harmonization was applied when extracting PCA. As mentioned in the overview chapter 11, it is possible to make the mistake of computing principal components on the whole dataset, before its separation into train and test sets. For this reason we implemented PCA extracting PCs from the training dataset and applying the transformation to both the train and test datasets.

An other aspect that leaps out comparing results with PCA with results without it, is a light improvement of performances of the adversarial learning with respect to the other harmonization pipelines. An explanation to this is the reduced source of noise and, consequently, of error 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/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 that just don't significantly change their activity between healthy and ASD people. Following this explaination, all the coefficients representing a link between them are just similar between the two groups and don't bring any benefit in this discrimination. 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 any information about the feature and the biological connection they represent. Each array containing 5995 feature is projected into a subspace of 800 or lower dimension. This leads to the loss of biological information of each feature which is not possible to retrieve if we aim to study the altered connection identified by a machine learning model, as we did in the following chapter. For this reason, to extract relevant features, linked to relevant altered functional connection, we need to work on the higher dimensional space of the whole set of features.

Chapter 15

SHAP - Implementation and results

In this chapter we present and discuss the results obtained by implementing a feature explainatory model using the DeepSHAP algorithm described in chapter 10. We also compared them with a more conventional random forest feature importance assessment.

Here we present a brief summary of all the analysis we performed using feature importance information.

We choose to run this analysis only on Pearson correlation coefficients since they gave the best classification results, which means achieving the best separability between controls subjects and ASD patients. We used the whole dataset consisting of ABIDE I + ABIDE II patients with the same characteristics as we choose for classification: only males and with an age range of 5-40. In addition we put in appendix some results obtained on the dataset ABIDE I only open eye.

In this section we are looking for 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. From the study of altered connections we are able to identify biological areas relevant for the distinction between healthy and ASD subjects.

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.

As a first, introductory analysis, in section 15.1.1 we start analyzing the most important feature that lead the classification of the DNNs models, using DeepSHAP. We extract 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 15.1.2, following the same harmonization pipelines, we extract 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 provided by sklearn.

This is, however, just a first, preliminary inspection of important features, and for this reason we limit our focus to the first 20 important feature for each pipeline. We choose to limit to this number because we can easily visualize them and have a visual comparison to identify differences across the different analysis.

A more meaningful analysis, is carried out afterwards. We select just 5 pipelines among all the one considered earlier, because we regard at these 5 as the most significant ways to proceed. In section 15.2 we will explain more in details what these analysis are and why we excluded the others and we will present results on common important features between these 5 analysis. From them we examine the biological meaning of these feature and the brain areas they involve, to create an

histogram of the most involved areas in discrimination of healthy and ASD subjects.

Operate with the DeepSHAP algorithm

To instantiate the desired class of SHAP called *DeepExplainer*, we require 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 will be. In particular using a big amount of background data, 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 choose 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 inputted a batch of test data, 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 choose to input all the test datasets to obtain a more statistically 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 get a final and general result, we computed the Shapley values 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.

Different types of SHAP plots

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 with 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.

If we want look at the overall result on all the test data, it would be inconvenient to create a plot like 15.1 for each data. 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 figure 15.2a: this plot

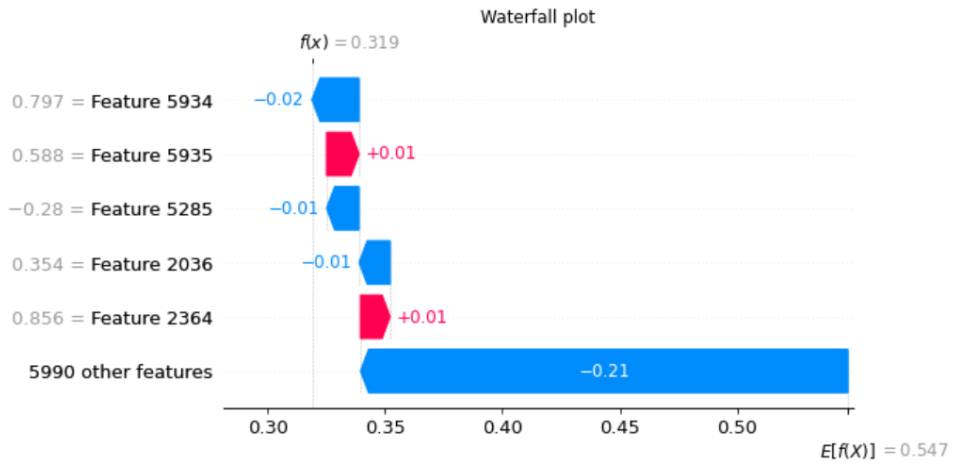


Figure 15.1: *Example of a waterfall plot obtained with a single test data. For each row is displayed the feature name and value and its contribution to the final output: negative contribution are colored blue and positive ones red. It is also marked the reference output in the bottom right corner, and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

shows for each instance (for each sample of the test data) [1] the contribution of one particular feature to the output computed with that test data. In other words, for each test data, 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 a value of zero. A zero value indicates that for a certain test data, that corresponding feature did not contributed to the output. This 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. Just to be clear, positive and negative adjectives are to be intended in a mathematical sense, as greater or smaller than zero, furthermore, pushing the output towards greater values (closer to 1 than to 0), means that the input data is being classified as ASD if ASD is associated to a label 1. This process of placing Shapley values for each feature in a plot, is repeated for each test data to obtain a scatter plot of Shapley values, where the related features are ordered by importance. Each spot is also colored red or blue. The color indicates, for each sample, the magnitude of the feature calculated with respect to the mean value of the feature across all the test dataset. With this additional information we are able to understand if a great or a small value of a feature pushes the model towards one output or towards the opposite (0 or 1). As a practical example to understand the concept of colors, with our data, they are useful if we ask "is a strong or a weak correlation between two areas, that contributed the most to the prediction of my model towards high values (and then more close to the label associated with ASD)?".

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, in 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 most clear to assess how much a feature is important in an absolute sense for a model.

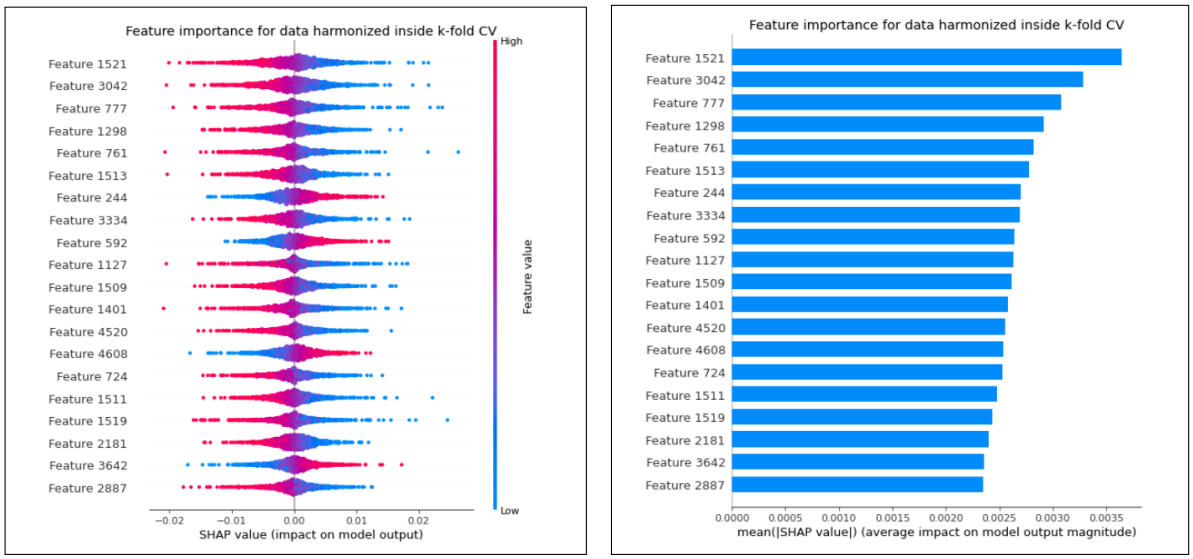
Alternatively, we can visualize the total amount of the contribution of each feature, by considering the absolute value of the Shapley values. In figure 15.2b a bar plot representing of all the

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

first 20 important features is reported. Each bar represents a feature importance, computed from equation 10.7.

For a better readability, we prefer reporting the results in terms of absolute Shapley value, where we just consider their module, regardless its positive or negative contribution to the output. This is a clearer way just to understand the absolute contribution of a single feature to a given analysis.

From this plot, we already have a more immediate understanding of what the most important features are. As we noticed from the example in figure 15.1, there is not a standing out feature, or group of features that have 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 regarding the first 20 most important features obtained from data where harmonization procedure is implemented inside the k-fold CV scheme. In plot 15.2a, for each feature each instance (each test sample) is represented by a dot, the 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. In figure 15.2b for each feature is represented the mean absolute value of all the Shapley values computed for each instance.*

15.1 Plots of the most relevant features for DNNs and random forest

15.1.1 Feature importance for DNNs

In figure 15.3 we show the first twenty 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 the four pipelines.

We find that: feature 592 and 1521 appear, even if in different order of importance, in all our four analysis. If we consider only common features between harmonization in k-fold, harmonization upstream and raw data, two more features appear: 1513 and 3334. We run this latter analysis excluding the adversarial network to allow comparison with results of the next section 15.1.2, extracted from random forest.

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

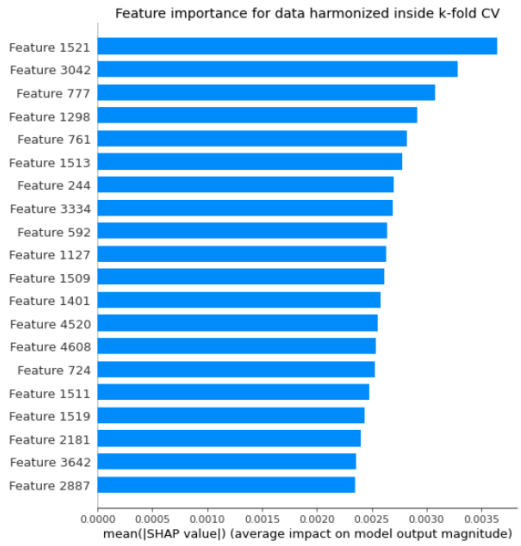
- Feature 592: correlation between Right Middle Temporal Gyrus (anterior division) and Left Superior Temporal Gyrus (anterior division)
- Feature 1521: correlation between Left Angular Gyrus and Right Middle Temporal Gyrus (posterior division)
- Feature 1513: correlation between Left Angular Gyrus and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

15.1.2 Feature importance for random forest

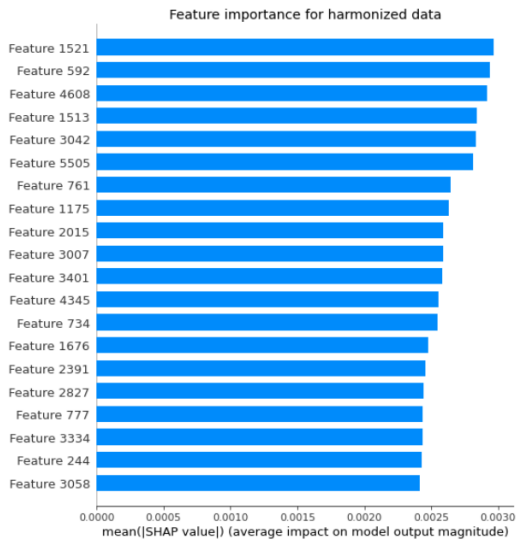
In this section we repeat the same analysis we obtained with a DNN in figure 15.3, with a random forest classifier. We plot the first 20 most important features obtained from the three main pipelines of harmonization in k-fold, harmonization upstream, and no harmonization. Features are extracted using the function *feature_importance_* provided by scikit-learn library. Results are shown in figure 15.4.

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

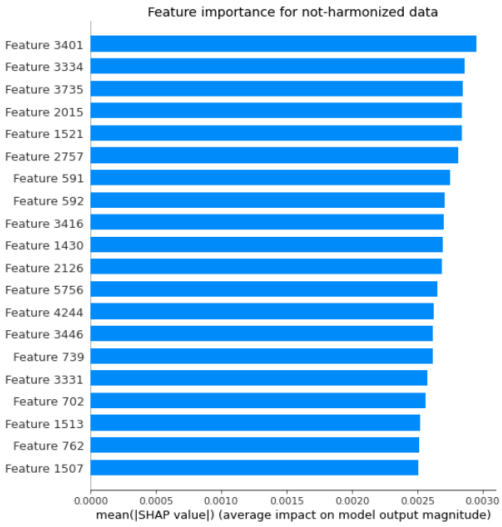
- Feature 710: correlation between Right Middle Temporal Gyrus (temporooccipital part) and Right Thalamus
- Feature 1127: correlation between Left Postcentral Gyrus and Right Postcentral Gyrus
- Feature 1703: correlation between Right Lateral Occipital Cortex (inferior division) and Right Supramarginal Gyrus (anterior division)
- Feature 748: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Right Thalamus
- Feature 2735: correlation between Right Precuneous Cortex and Right Middle Temporal Gyrus (anterior division)



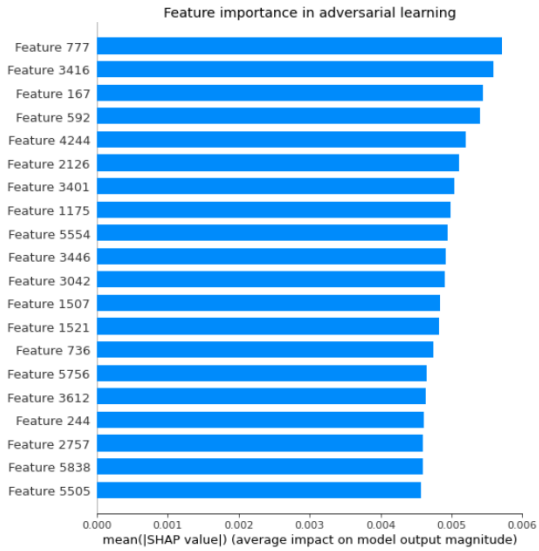
(a) *Important features from harmon. k-fold*



(b) *Important features from harmon. upstream*



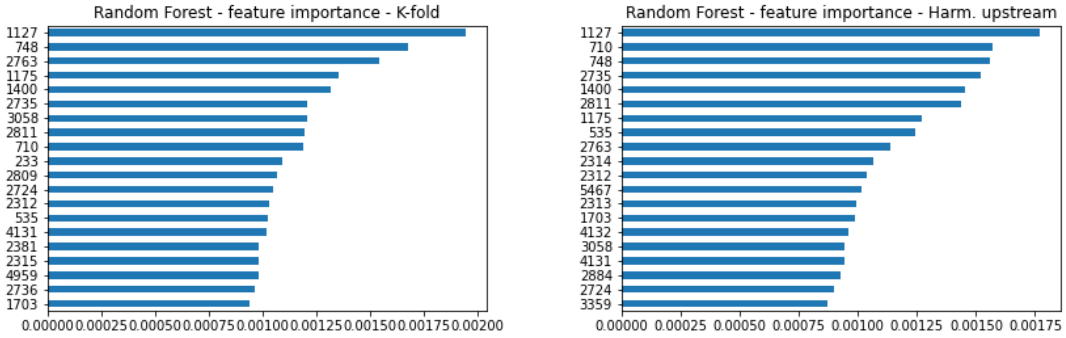
(c) *Important features from no harmon.*



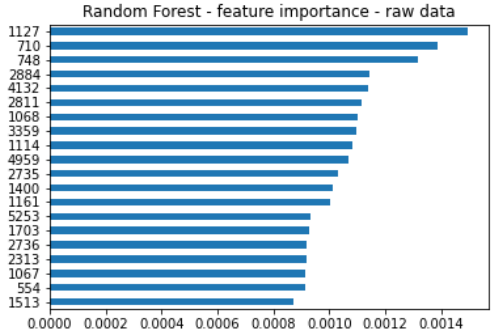
(d) *Important features from adversarial*

Figure 15.3: *First twenty most important features plotted with a bar plot of mean absolute Shapley values. Extracted from each of the four harmonization pipeline with DNN and adversarial network using DeepSHAP. Figure 15.3a shows the important features extracted from harmonized data, with harmonization procedure implemented inside* ***k-fold*** *CV. Figure 15.3bshows the important features extracted from harmonized data, with harmonization procedure implemented* ***upstream****, before the k-fold. Figure 15.3c shows important features extracted from* ***raw****, not-harmonized data. Figure 15.3d shows important features extracted from raw data using the* ***Adversarial*** *neural network*

- 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)



(a) *Important features from harmon. k-fold* ***(b)*** *Important features from harmon. upstream*



(c) *Important features from no harmon.*

Figure 15.4: *Feature importance obtained with a Random Forest classifier trained on the entire (ABIDE I + ABIDE II) dataset. Figure 15.4a shows the important features extracted from harmonized data, with harmonization procedure implemented inside* ***k-fold*** *CV. Figure 15.4bshows the important features extracted from harmonized data, with harmonization procedure implemented* ***upstream****, before the k-fold. Figure 15.4c shows important features extracted from* ***raw****, not-harmonized data.*

15.1.3 Comments on feature importance

Comparing results obtained using different classifiers: a DNN and a random forest classifier, we notice from plots 15.3 and 15.4 that there are no common features among the first 20. This is most likely due to the difference in the inner algorithm that guides the decisional process. A DNN and a random forest have very different modes of operation. This can lead the two of 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 achieved. However, as we can notice from plots, random forest feature importance is able to determine some feature that stand out over the others, resulting in an abrupt descending trend of importances. The same characteristic is not visible with features extracted from the DNNs except for the pipeline of harmonization inside k-fold where it seems that some feature have a greater contribution than others and the descending in 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 does not bring up new information, helps in data cleaning, and remove some noise due to site-distinctive patterns. Harmonization thus helps to obtain a better and a more defined assessment of what the most important features are, in discrimination of controls/ASD.

Even if there are no overlapping features between analysis with DNN and with random forest, if we take a look at the lists of the brain areas involved in these correlations, we can notice that certain brain areas, such as the temporal gyri, are in common between these two results. This can suggest that even if the feature itself does not determine in the same way the output of these two models, maybe some brain areas present in a feature, are involved more than others. 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 Cross analysis of important features

So far, we discussed results linked to images just to have a visual comparison of feature importance between DNN and random forest. Now we go deeper to fulfill a more meaningful analysis of what are the important features and what they represent.

Since, as explained in section 14.4, the pipeline of upstream harmonization is a misleading way to proceed because it creates a bias in results, from now on, we exclude it from our analysis. In this section, when we refer to harmonized data, we are referring to data with harmonization procedure implemented inside k-fold.

To assess what are the most recurrent features between these different types of analysis, we choose to limit our analysis to the 1% important features among 5995. To this end we collected the first 60 most important features and we limited our analysis to just 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

Firstly we checked if and how many common features we find among the 60 most important features between these five classification. However, there are no common features between all these 5 pipelines taking into account only the first 60.

Now, we carry out the analysis on subgroups of classification methods and we report the number of common important features among the 60 most relevant one as a number and as percentage. This kind of analysis is useful for example to assess whether the harmonization procedure significantly changes important features. It is also aimed to quantify the common features between a DNN and a random forest classifier.

- Focusing just on **DNN**, we compared pipelines 1 and 3 finding that 22 features (37 %) out of 60 are common between harmonized and raw data.
- Focusing just on **Random Forest** classifier, we compared pipelines 2 and 4 finding that 33 features out of 60 are common (55%) between harmonized and raw data.
- Comparing results on **harmonized data** obtained with **DNN** (pipeline 2) and **random forest** (1) we obtain 13 common features (22 %)
- Comparing results on **raw data** obtained with **DNN** (pipeline 3) and **random forest** (4) we obtain 7 common features (12%)
- Comparing results of the **Adversarial network** (5) with the **harmonized data** with DNN (1) we obtain 21 common features (35%)
- Comparing results of the **Adversarial network** (5) with DNN classification of **raw data** (3) we obtain 28 common features (47%)

15.3 Brain areas related to 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 in healthy/ASD discrimination.

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

Figure 15.5 shows for each pipeline, what are the brain areas recurrent among the 60 most important feature. In figure 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 analysis. In figure 15.6 the overall histogram of the most recurrent brain areas between these common feature is reported.

15.4 Discussion on common features and important brain areas

Common important features

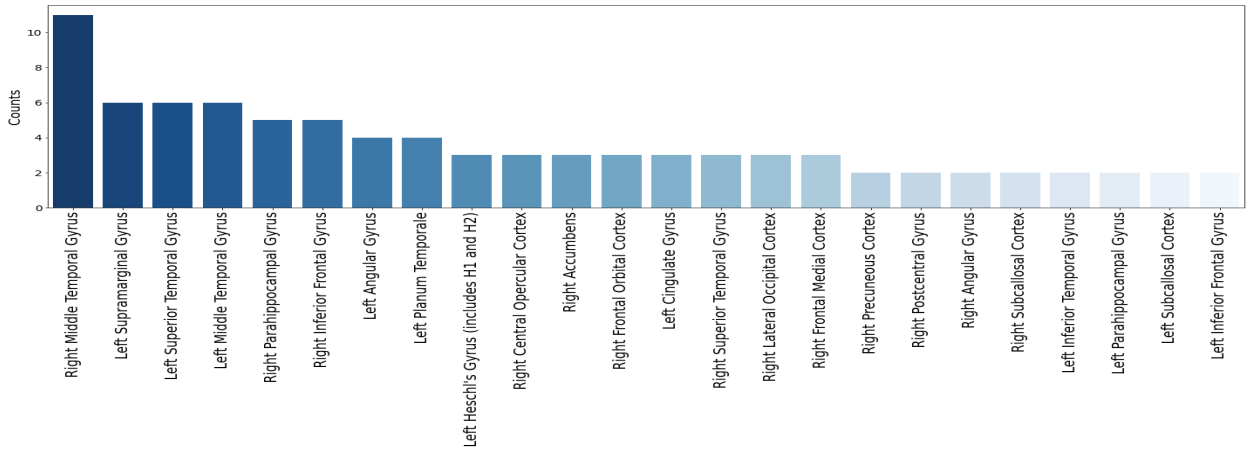
By checking the common features between different machine learning model and harmonization procedure, we can have a better comparison on how much impact these factor have on the assessment of feature importance.

Looking at the results listed in section 15.2 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.

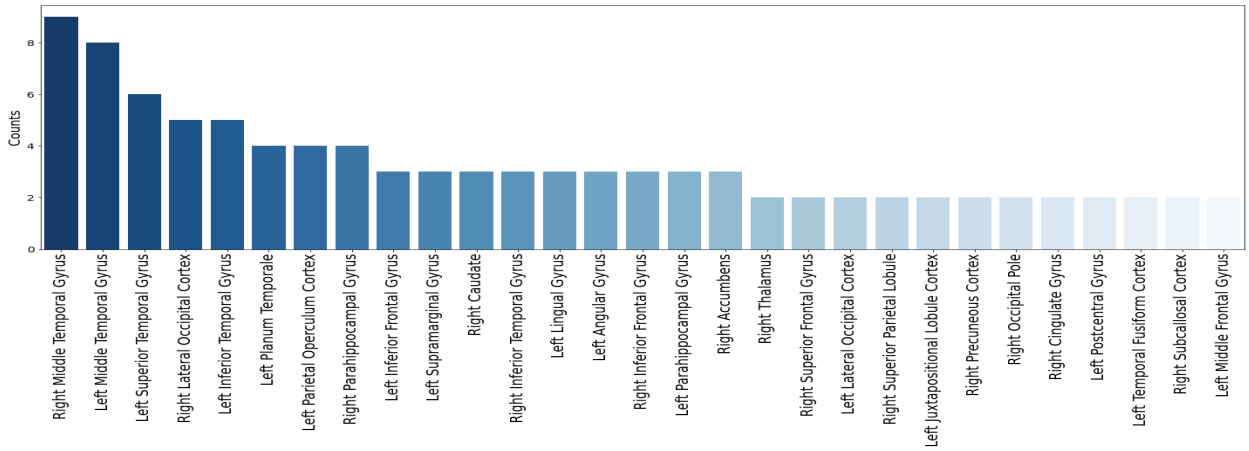
We can also state that harmonization procedure changes, especially in a DNN, the most important features. As we pointed out when discussing the plots of feature importance, harmonization is able to remove noise from data and reveals in a cleaner way what features are really important for the network, 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.

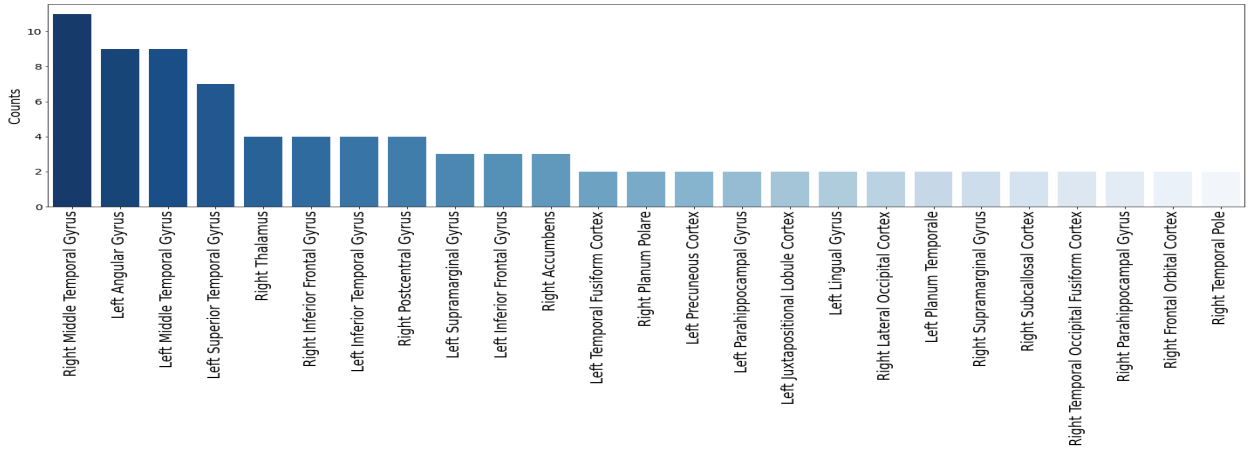
Between raw data and adversarial network a similarity of feature importance $\approx 47\%$ that is greater than what we obtained with raw data and harmonized data $\approx 37\%$. This means that with the adversarial network data are modified and harmonized according to some inner processes, which occur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



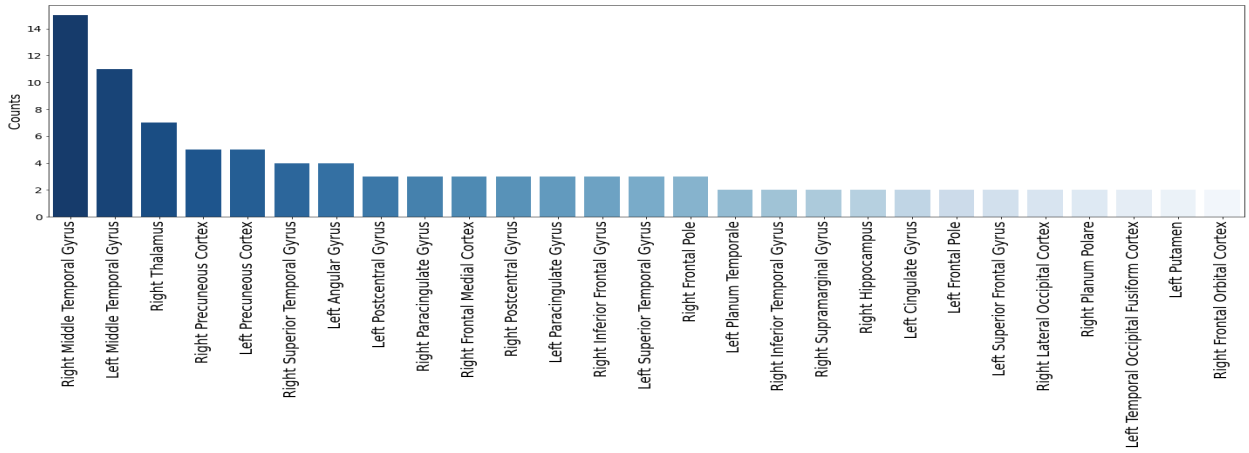
(a) *Important areas from Raw data classified with DNN*



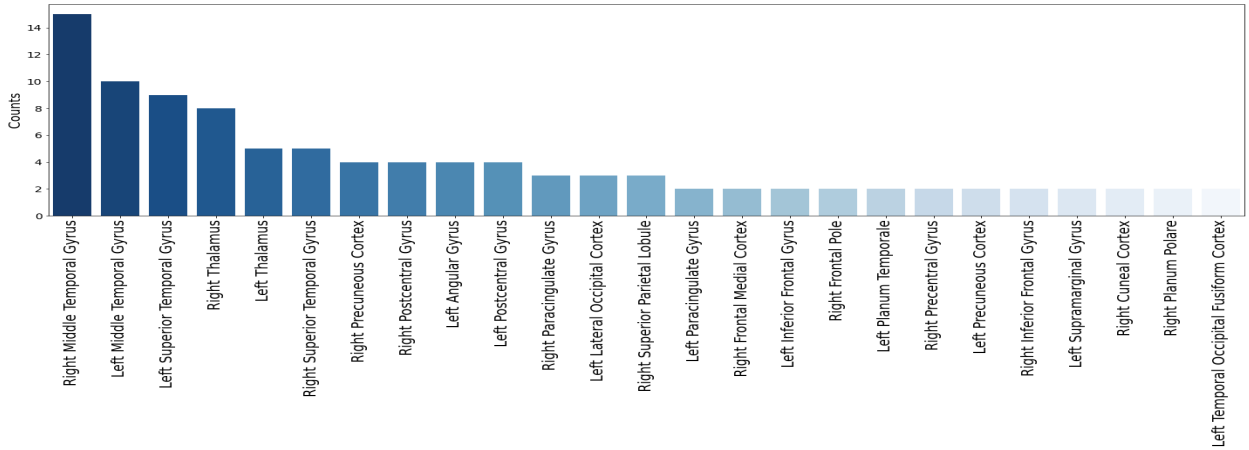
(b) *Important areas from Raw data classified with Adversarial network*



(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 of different classification procedures*

Important areas

From the analysis of important brain areas, especially looking at figure 15.6 we can look at ≈ 30 among 110 areas that contributed the most in discrimination between controls and ASD. We compared the brain areas extracted from these analysis with the brain areas obtained by Saponaro et al. [65]. They studied with a random forest classifier the most discriminative features extracted from structural images of patients from the ABIDE dataset. For parcellation they used the atlas Desikan-Killiany-Tourville implemented in Freesurfer [2] consisting on 62 total ROIs. This number differs from the number of ROIs implemented in Harvard-Oxford, equal to 110. For this reason for some feature, 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 paper. We also provide a brief explanation about the role of different brain regions and how they could be linked to the development of ASD. Information about brain areas are taken from the Wikipedia page of the areas we focus on. The scope of this short investigation is not to achieve a medical accurate analysis but just to give some glimpses of the function of anatomical regions found in common from the structural and the functional analysis. 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 r. 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 orbitofrontal 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 plays and important role on the analysis of logical conditions as well. For other regions like the precuneous cortex is still quite unclear what processes they are mainly involved in. All these regions, and many others, 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 is too simplistic.

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

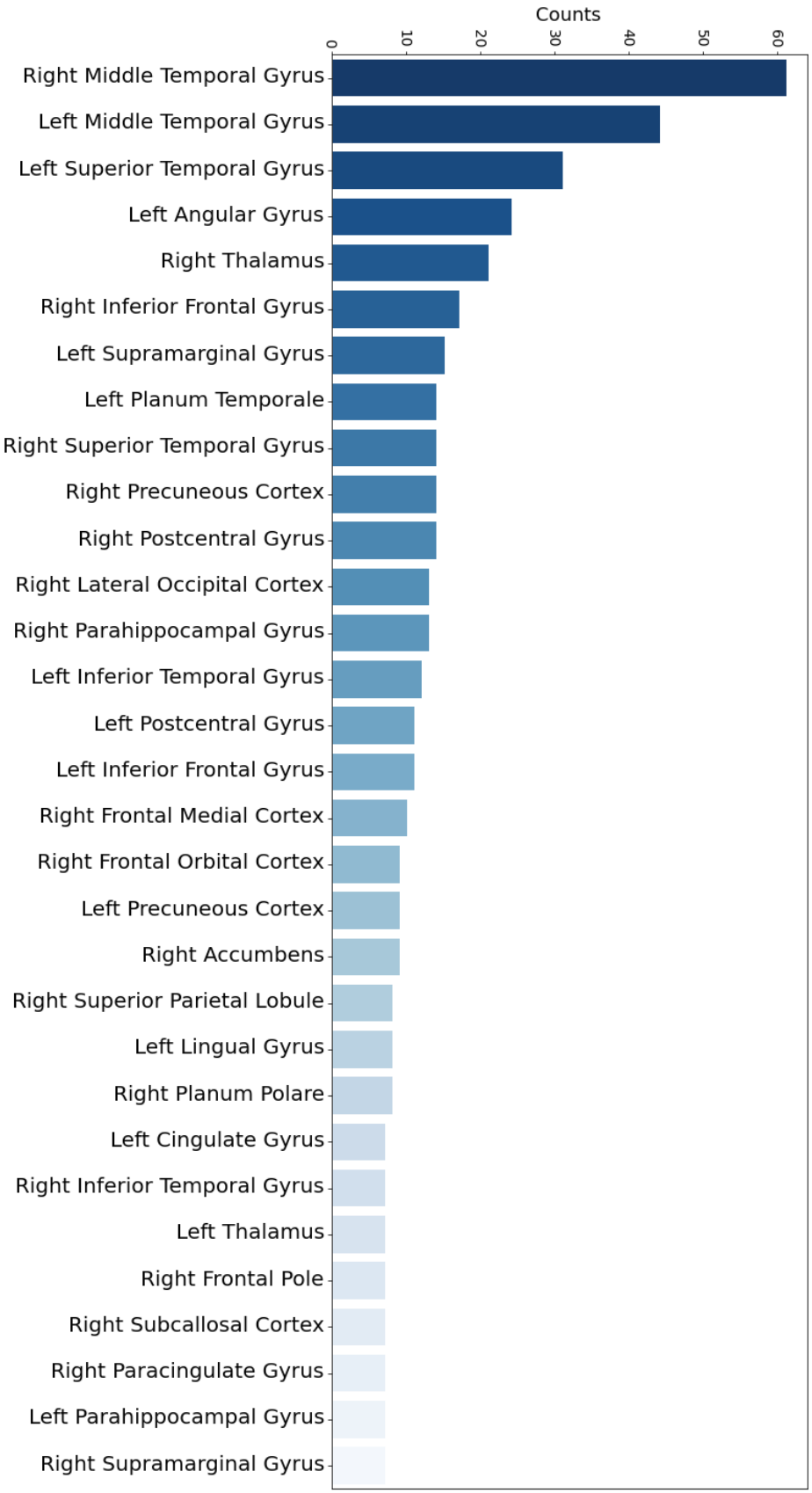


Figure 15.6: *Histogram of the most important brain areas putting together results obtained from all the five analysis listed above.*

Chapter 16

Conclusions

In this thesis we 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. ASD has a strong social impact on the quality of life of affected individuals and of their families. For its diagnosis and characterization 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 ABIDE initiative. Brain functional connectivity alterations in ASD have already been reported. However, the choice of each element in the data processing and analysis pipeline and of a number of parameters for analysis module, can have a strong impact on the identification of possible differences between subjects ASD and healthy controls.

In this work we present the data analysis pipeline we followed to extract a measure of functional connectivity from rs-fMRI raw data. To preprocess raw data, CPAC allowed to assemble a robust and reproducible pipeline which allowed us to extract timeseries for each patient. To investigate the functional connectivity and seek for the best strategy to identify differences between healthy controls and ASD patients, we compared two different approaches. One is based on Pearson correlation while the other employs a time-frequency analysis using wavelet transforms. Using machine learning techniques, Pearson-based correlation coefficients allowed to achieve a better separation between controls and ASD.

An other topic we covered in this thesis is related to the harmonization of multicentric data. We compared two different procedures with the intent of removing site-related patterns from data. One consists of an analytical harmonization of data using the NeuroHarmonize pagkage. Following this approach, we also assessed the best way to implement it during a classification task with a machine learning model. The second harmonization strategy employs an originally developed neural network specifically designed to perform an adversarial learning and remove biases toward sites during the classification of control data vs ASD data. We noticed that the two different implementation of harmonization allow, on average, to obtain similar discrimination performances.

Working with machine learning algorithms, especially when dealing with clinical data, it is of great importance to assess what feature are decisive for the prediction of a model. 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 works with. Applying this to functional connectivity data it is possible to identify important alteration between healthy and ASD subjects. All the procedures: analytical harmonization + deep learning, and adversarial learning lead to a common set of important features. Similarity becomes even greater when from altered connections we trace back to the sources of each one of them, identifying what

are the brain areas mainly involved. We determined a set of brain areas and compared them with other studies, observing that a great number of the areas we found relevant are often identified as related to ASD.

Raw data are provided by ABIDE which is a multicenter dataset created by pooling together data acquired by different American and European centers over the years. Dealing with multicenter data can be misleading because of the possible presence of the so-called *batch effect*. Data affected by batch effect exhibit characteristic features or distributions related to the acquisition center/procedures. For this reason, before working with multicenter data to perform statistical analysis, they should be modified using a harmonization procedure. We addressed this issue by employing two different harmonization techniques. One is analytical, based on ComBat technique implemented with the open source Python package NeuroHarmonize. The other was implemented inside a deep neural network through an originally developed adversarial learning model.

Machine learning was employed to perform classification of control subjects vs ASD patients. We compared classification performances using Pearson and wavelet coefficients. We sought to find which one allowed the best discrimination between controls and ASD, and found out that Pearson coefficients performed better than wavelet ones.

We also assessed the best way to implement harmonization during the training of a machine learning algorithm.

An other issue that arises working on this data is the problem of dimensionality. We deal with a dataset consisting on less than 1500 datapoints and each one consisting on 5995 features. Using a dataset like this, machine learning models work in a condition of overfitting. We implemented a Principal Component Analysis (PCA) to reduce the number of features for each subject. PCA allows to effectively reduce the dimensionality of the problem of one order of magnitude without affecting classification performances.

Furthermore, we covered the topic of ML explainability by determining the features that contributed the most to the prediction of a deep neural network. Since each feature is representative of a correlation between two brain areas, we look for altered connection between controls and ASD, and from a study of these connections, we assessed the brain areas mainly involved. To this end we implemented SHAP, an algorithm based on game theory to assess the contribution of a feature in the prediction of a ML model. The algorithm we used is called DeepSHAP and it is specifically designed for deep neural networks. It allows to determine an importance coefficient for each feature of each subject. By collecting coefficients over all the subjects assigned to a test set, we were are able to determine the features, and consequently the altered functional connections that influenced the most the controls vs ASD classification. We compared the results obtained with SHAP, with the results of the feature importance extracted from a conventional random forest classifier. Being a random forest a more easily interpretable classifier, we aimed to discover whether or not important features are the same in the deep neural network and in the random forest approaches.

The identification of these altered connections allowed us to assess which brain areas are more involved in the ASD condition.

Even if this work employed ASD data, its pipeline are generals and can potentially be applied to all the mental disease and disorders where an altered functional connectivity is suspected. The use of explainable AI methods represents an effective tool to interpret machine learning model predictions and allows humans to trust more ML methods. Using explainable AI, it is possible to identify the main brain areas involved in functional connectivity alterations. Using these information, we could tackle the problem of dimensionality during classification. An idea to tackle the problem of dimensionality, could be to focus only on those areas and calculate only the connectivity related to them and use it as input to the ML model. This would remove a great amount of noise or uninformative data and help a ML model in making more more decisive predictions.

With regard to ASD data, the heterogeneity of conditions this disorder manifests into, makes it hard to define and subsequently, to identify univocal biomarkers.

During the years, diagnostic parameters change, including the broadening of the range of conditions associated to this disorder.

This is one of the main aspects that restrain from achieving high classification performances. The other aspect involves the dimension of the dataset. The unprecedented project carried out by the ABIDE initiative for the investigation of ASD is of great importance, but it cannot be considered done yet. The collection of an increasing amount of data is a crucial point to increase the accuracy of statistical and machine learning models. So far, it is not possible to state that the investigation methods employed in this thesis can also represent a diagnostic tool. Also, all the studies carried out until now, are mainly focused on males because available data are in a greater number. In the future, the expansion of the ABIDE dataset may lead to achieving greater classification performances both on males and females subjects.

In conclusion, a promising approach may be the combination of different investigation tools such as fMRI and structural MRI data. Hopefully, in a not-too-far future, the combination of the two of them, could bring to the development of a non-invasive diagnostic tool for challenging issues such as ASD, schizophrenia and other mental disorders whose etiology still remain unclear.

Appendix

.1 Classification results with different neural networks

In table 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 choose to employ a network with a configuration 264-8-1, even if with a structure 128-8-1 we obtained similar results. We choose then a 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 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.*

.2 Feature importance results on ABIDE I open eye dataset

The same tests we did to assess feature importance on ABIDE I + II dataset, we carried out on ABIDE I with only patient with open eye. In figure 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

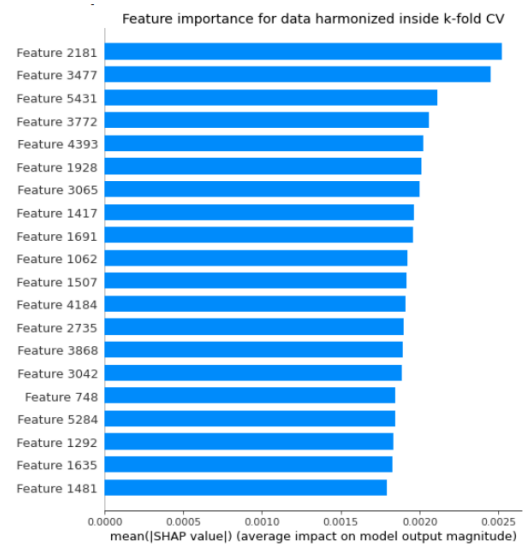
- Feature 762: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Left Middle Frontal Gyrus
- Feature 1417: correlation between Left Supramarginal Gyrus (posterior division) and Left Middle Temporal Gyrus (temporooccipital part)
- Feature 748: correlation between Left Middle Temporal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

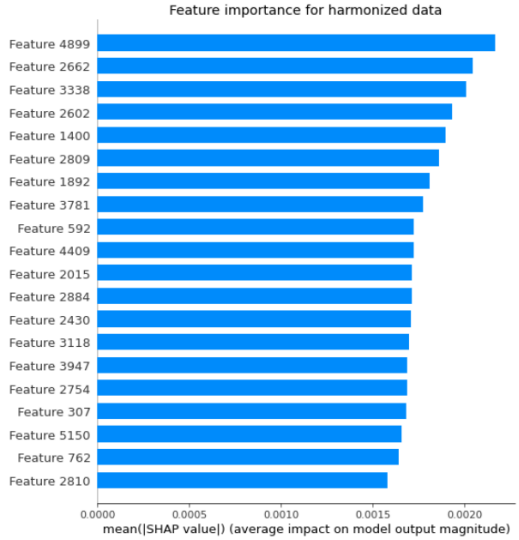
- Feature 2690: correlation between Left Cingulate Gyrus (posterior division) and Right Frontal Medial Cortex
- Feature 1127: correlation between Left Postcentral Gyrus and Right Postcentral Gyrus
- Feature 2313: correlation between Right Paracingulate Gyrus and Left Middle Temporal Gyrus (anterior division)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

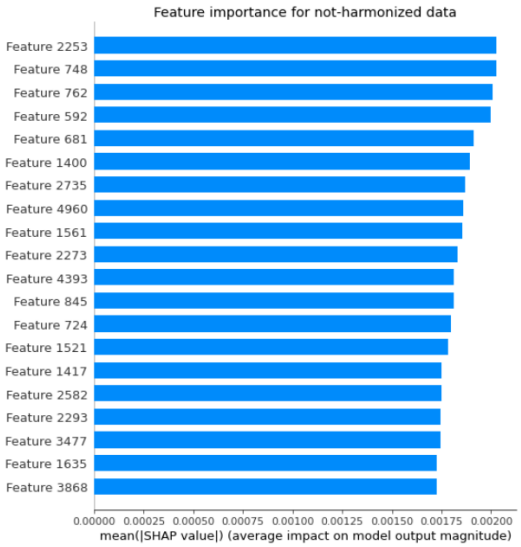
- Focusing on random forest classifier, we compared important features between 2 and 4 and found that 34 features out of 60 are common (57%).
- Focusing on DNN we compared pipelines 1 and 3 finding that 19 features (32 %) out of 60 are common.
- Comparing the same analysis pipeline with DNN and Random 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%)



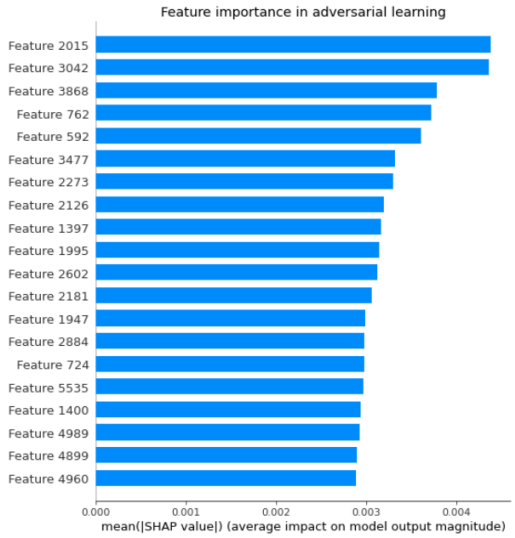
*(a) Bar plot of important features extracted from harmonized data, with harmonization procedure implemented inside **k-fold** CV*



*(b) Bar plot of important features extracted from harmonized data, with harmonization procedure implemented **upstream**, before the k-fold*

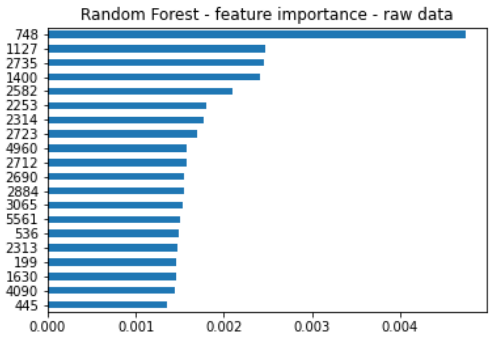


*(c) Bar plot of important features extracted from **raw**, not-harmonized data*

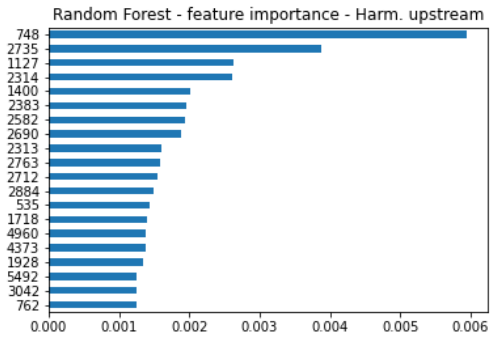


*(d) Bar plot of important features extracted from raw data using the **Adversarial** neural network,*

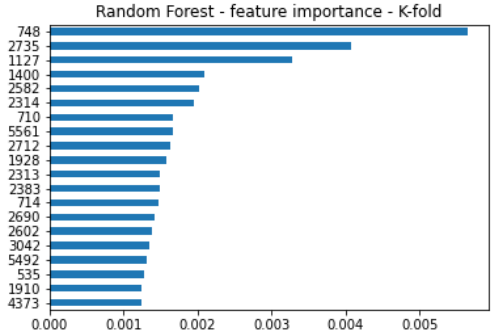
Figure 1: *Results obtained on ABIDE I only open eye dataset. First twenty most important features plotted with a bar plot of mean absolute shap values. Extracted from each of the four analysis we run with DNNs and Adversarial network.*



(a) Important features from no harmon.

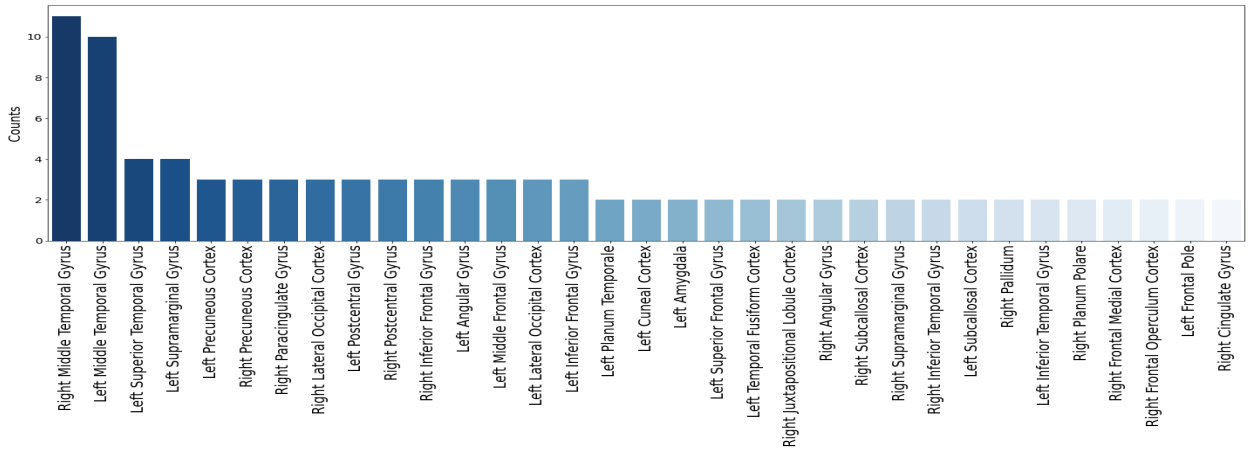


(b) Important features from harmon. upstream

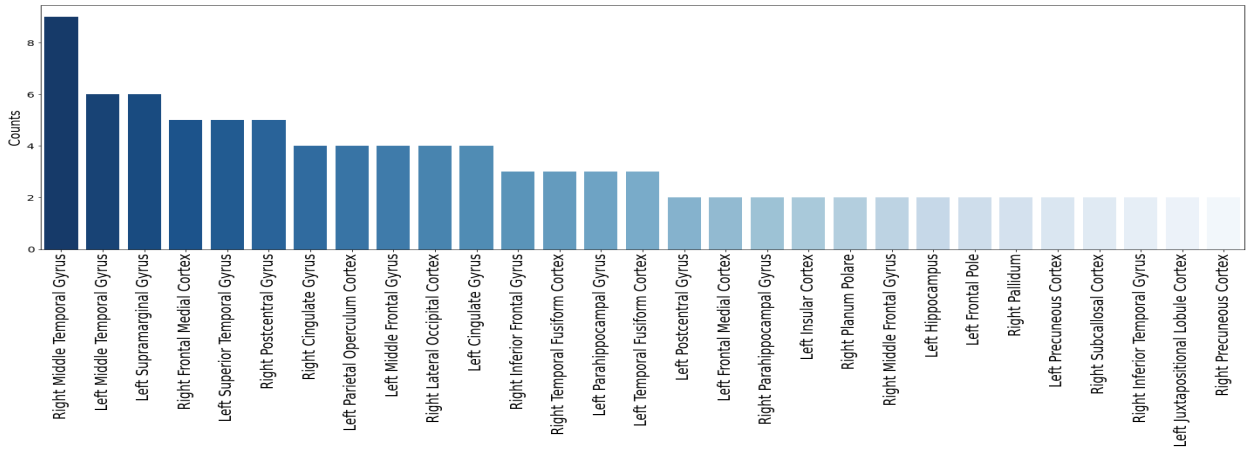


(c) Important features from harmon. k-fold

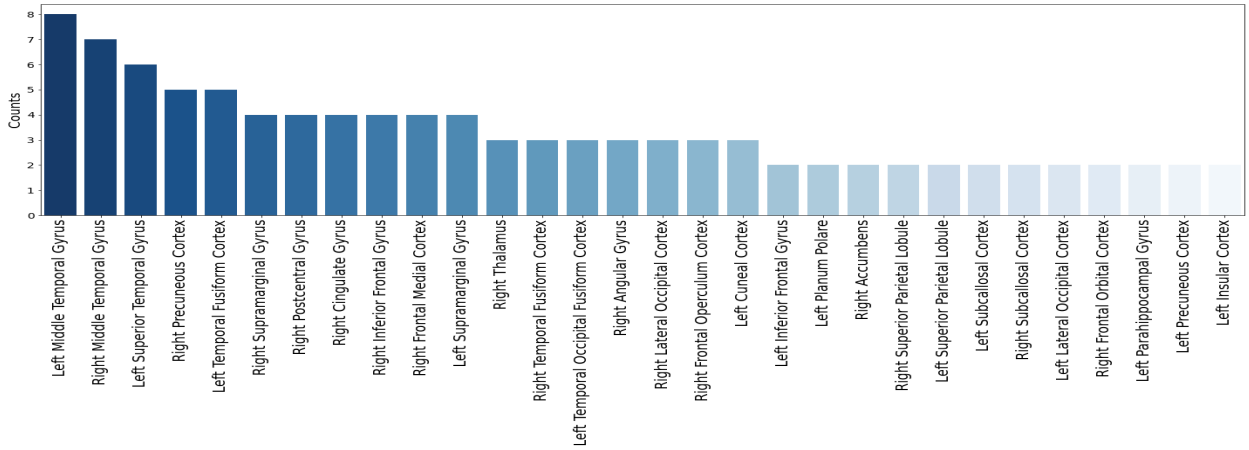
Figure 2: *Feature importance obtained from a Random Forest classifier employed in different harmonization pipelines. Results on ABIDE I only open eyes.*



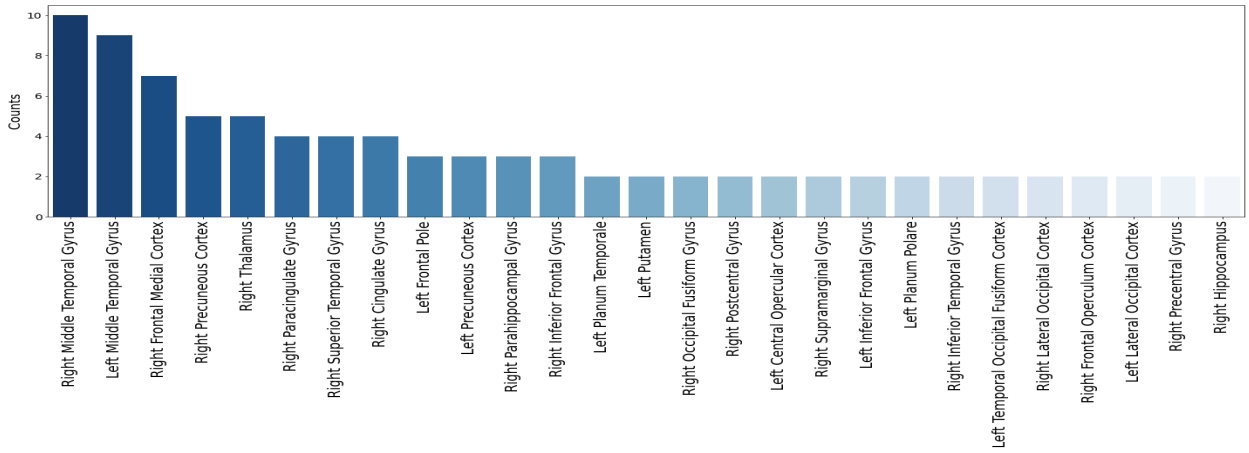
(a) *Important areas from Raw data classified with DNN*



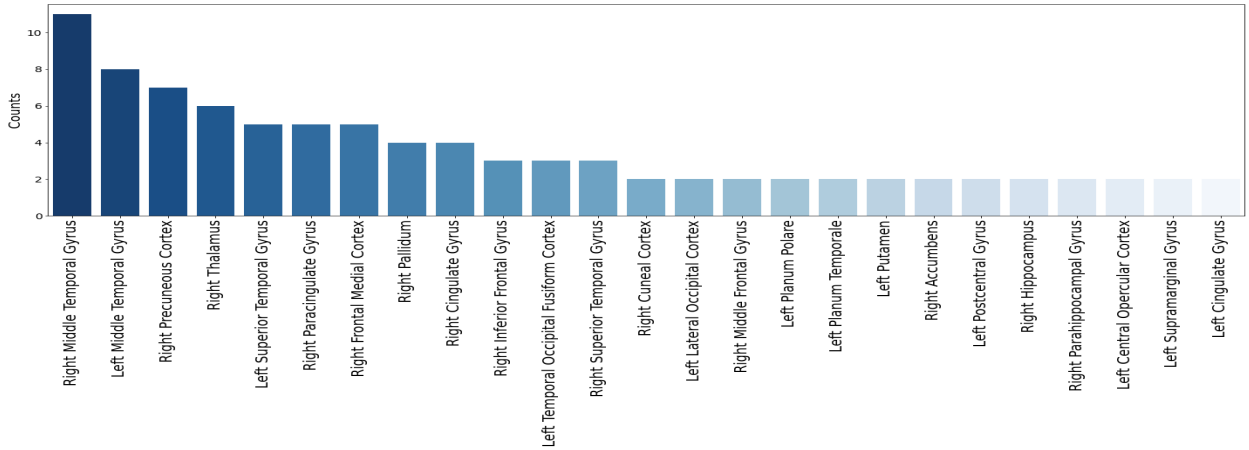
(b) *Important areas from Raw data classified with Adversarial network*



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

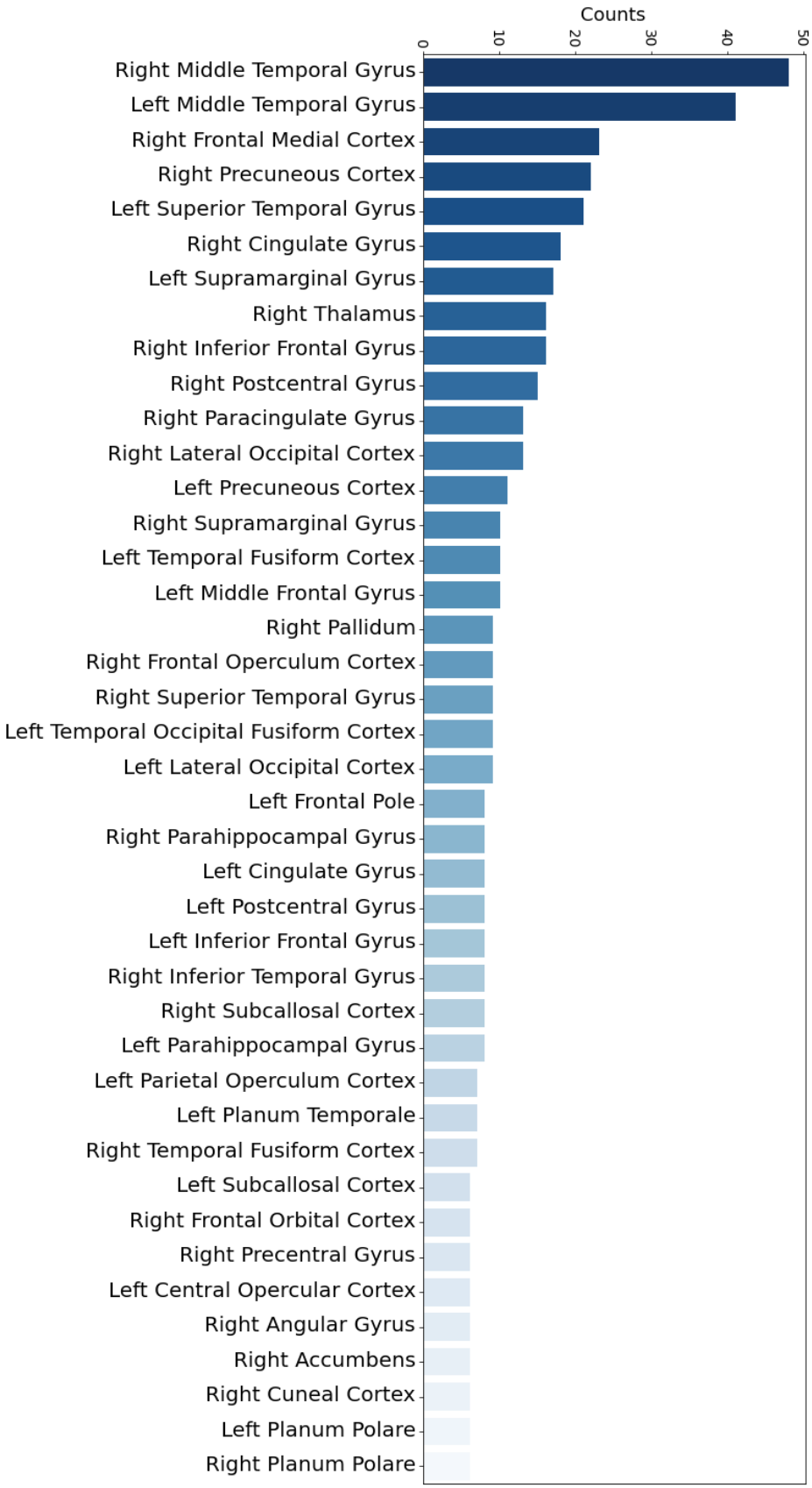


Figure 4: *Histogram of the most important brain areas putting together results obtained from all the five analysis listed above.*

Bibliography

[1] Matthew J. Maenner, Kelly A. Shaw, Amanda V. Bakian, Deborah A. Bilder, Maureen S. Durkin, Amy Esler, Sarah M. Furnier, Libby Hallas, Jennifer Hall-Lande, Allison Hudson, Michelle M. Hughes, Mary Patrick, Karen Pierce, Jenny N. Poynter, Angelica Salinas, Josephine Shenouda, Alison Vehorn, Zachary Warren, John N. Constantino, Monica DiRienzo, Robert T. Fitzgerald, Andrea Grzybowski, Margaret H. Spivey, Sydney Pettygrove, Walter Zahorodny, Akilah Ali, Jennifer G. Andrews, Thaer Baroud, Johanna Gutierrez, Amy Hewitt, Li-Ching Lee, Maya Lopez, Kristen Clancy Mancilla, Dedria McArthur, Yvette D. Schwenk, Anita Washington, Susan Williams, and Mary E. Cogswell. 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*, 70:1–16, 12 2021.

[2] SAMUEL B. GUZE. Diagnostic and statistical manual of mental disorders, 4th ed. (dsm-iv). *American Journal of Psychiatry*, 152:1228–1228, 8 1995.

[3] World Health Organization. The icd-10 classification of mental and behavioural disorders : diagnostic criteria for research, 1993.

[4] Isabelle Rapin and Roberto F. Tuchman. Autism: Definition, neurobiology, screening, diagnosis. *Pediatric Clinics of North America*, 55:1129–1146, 10 2008.

[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, 2005.

[7] Ann Le Couteur, Gyles Haden, Donna Hammal, and Helen McConachie. 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*, 38:362–372, 2 2008.

[8] C. M. Freitag. The genetics of autistic disorders and its clinical relevance: A review of the literature, 1 2007.

[9] 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*, 11:541–551, 4 2017.

[10] Kaustubh Supekar, Lucina Q. Uddin, Amirah Khouzam, Jennifer Phillips, William D. Gaillard, Lauren E. Kenworthy, Benjamin E. Yerys, Chandan J. Vaidya, and Vinod Menon. Brain hyperconnectivity in children with autism and its links to social deficits. *Cell Reports*, 5:738–747, 11 2013.

[11] Giovanna Spera, Alessandra Retico, Paolo Bosco, Elisa Ferrari, Letizia Palumbo, Piernicola Oliva, Filippo Muratori, and Sara Calderoni. Evaluation of altered functional connections in male children with autism spectrum disorders on multiple-site data optimized with machine learning. *Frontiers in Psychiatry*, 10, 9 2019.

[12] Chao Zhang, Nathan D. Cahill, Mohammad R. Arbabshirani, Tonya White, Stefi A. Baum, and Andrew M. Michael. Sex and age effects of functional connectivity in early adulthood. *Brain Connectivity*, 6:700–713, 11 2016.

[13] C. Edward Coffey, Joseph F. Lucke, Judith A. Saxton, Graham Ratcliff, Lori Jo Unitas, Brenda Billig, and R. Nick Bryan. Sex differences in brain aging. *Archives of Neurology*, 55:169, 2 1998.

[14] Víctor Costumero, Elisenda Bueichekú, Jesús Adrián-Ventura, and César Ávila. Opening or closing eyes at rest modulates the functional connectivity of v1 with default and salience networks. *Scientific Reports*, 10:9137, 12 2020.

[15] Robert W. Brown, E. Mark Haacke, Yu-Chung N. Cheng, Michael R. Thompson, and Ramesh Venkatesan. *Magnetic Resonance Imaging*. John Wiley and Sons Ltd, 4 2014.

[16] Scott A. Huettel, Allen W. Song, and Gregory McCarthy. *Functional Magnetic Resonance Imaging*. 2009.

[17] Mark Jenkinson and Michael Chappell. Introduction to neuroimaging analysis, 2018.

[18] Vahid Mirjalili Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 2019.

[19] F Pedregosa, G Varoquaux, A Gramfort, B Michel V., Thirion, O Grisel, M Blondel, R Prettenhofer P., Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] Sarah Guido Andreas C. Müller. *Introduction to Machine Learning with Python*. O'Reilly Media, Inc.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2 2015.

[22] Luca Baldini, Niccolò Di Lalla, Massimiliano Razzano, and Carmelo Sgrò. Introduzione all'analisi dei dati per il laboratorio di fisica.

[23] Don Ross. Game theory, 2021.

[24] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1 1950.

[25] *Notes on the N-Person Game amp;mdash; I: Characteristic-Point Solutions of the Four-Person Game*. RAND Corporation, 1951.

[26] L. S. Shapley. 17. a value for n-person games, 12 1953.

[27] Encarnación Algaba, Vito Fragnelli, and Joaquín Sánchez-Soriano. Handbook of the shapley value.

[28] Eric Fombonne. Epidemiology of pervasive developmental disorders. *Pediatric Research*, 65:591–598, 6 2009.

[29] Janine Bijsterbosch, Stephen Smith, and Christian Beckmann. Introduction to resting state fmri functional connectivity oxford neuroimaging primers.

[30] Matthew Brett, Ingrid S. Johnsrude, and Adrian M. Owen. The problem of functional localization in the human brain. *Nature Reviews Neuroscience*, 3:243–249, 3 2002.

[31] Matthew Brett. The mni brain and the talairach atlas, 2002.

[32] Nikos Makris, Jill M. Goldstein, David Kennedy, Steven M. Hodge, Verne S. Caviness, Stephen V. Faraone, Ming T. Tsuang, and Larry J. Seidman. Decreased volume of left and total anterior insular lobule in schizophrenia. *Schizophrenia Research*, 83:155–171, 4 2006.

[33] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *NeuroImage*, 15:273–289, 1 2002.

[34] R. Cameron Craddock, G.Andrew James, Paul E. Holtzheimer, Xiaoping P. Hu, and Helen S. Mayberg. A whole brain fmri atlas generated via spatially constrained spectral clustering. *Human Brain Mapping*, 33:1914–1928, 8 2012.

[35] Rahul S. Desikan, Florent Ségonne, Bruce Fischl, Brian T. Quinn, Bradford C. Dickerson, Deborah Blacker, Randy L. Buckner, Anders M. Dale, R. Paul Maguire, Bradley T. Hyman, Marilyn S. Albert, and Ronald J. Killiany. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *NeuroImage*, 31:968–980, 7 2006.

[36] Craddock Cameron, Benhajali Yassine, Chu Carlton, Chouinard Francois, Evans Alan, Jakab András, Khundrakpam Budhachandra, Lewis John, Li Qingyang, Milham Michael, Yan Chaogan, and Bellec Pierre. The neuro bureau preprocessing initiative: open sharing of preprocessed neuroimaging data and derivatives. *Frontiers in Neuroinformatics*, 7, 2013.

[37] 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, 2020.

[38] Rick Wicklin. Fisher's transformation of the correlation coefficient, 2017.

[39] Isidoro Ferrante. *Elaborazione dei segnali per la fisica*. Pisa university press, 2015.

[40] J.C. van den Berg. *Wavelets in Physics*. Cambridge University Press, 8 1999.

[41] Donald B. Percival and Andrew T. Walden. *Wavelet Methods for Time SeriesAnalysis*. Cambridge University Press, 2000.

[42] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1 1992.

[43] Karsten Müller, Gabriele Lohmann, Jane Neumann, Maren Grigutsch, Toralf Mildner, and D. Yves Von Cramon. Investigating the wavelet coherence phase of the bold signal. *Journal of Magnetic Resonance Imaging*, 20:145–152, 7 2004.

[44] 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 application of the cross wavelet transform and wavelet coherence to geophysical time series, 2004.

[45] Paul C. Liu. Wavelet spectrum analysis and ocean wind waves, 1994.

[46] Christopher Torrence and Peter J. Webster. Interdecadal changes in the enso–monsoon system. *Journal of Climate*, 12:2679–2690, 8 1999.

[47] Christopher Torrence and Gilbert P. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79:61–78, 1 1998.

[48] 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*, 154:143–151, 2 2018.

[49] Hartmann Atm. Lecture 11: White and red noise.

[50] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8:118–127, 1 2007.

[51] Jean-Philippe Fortin, Drew Parker, Birkan Tunç, Takanori Watanabe, Mark A. Elliott, Kosha Ruparel, David R. Roalf, Theodore D. Satterthwaite, Ruben C. Gur, Raquel E. Gur, Robert T. Schultz, Ragini Verma, and Russell T. Shinohara. Harmonization of multi-site diffusion tensor imaging data. *NeuroImage*, 161:149–170, 11 2017.

[52] Angela Lombardi, Nicola Amoroso, Domenico Diacono, Alfonso Monaco, Sabina Tangaro, and Roberto Bellotti. Extensive evaluation of morphological statistical harmonization for brain age prediction. *Brain Sciences*, 10:364, 6 2020.

[53] Jean-Philippe Fortin, Nicholas Cullen, Yvette I. Sheline, Warren D. Taylor, Irem Aselcioglu, Philip A. Cook, Phil Adams, Crystal Cooper, Maurizio Fava, Patrick J. McGrath, Melvin McInnis, Mary L. Phillips, Madhukar H. Trivedi, Myrna M. Weissman, and Russell T. Shinohara. Harmonization of cortical thickness measurements across scanners and sites. *NeuroImage*, 167:104–120, 2 2018.

[54] Raymond Pomponio, Guray Erus, Mohamad Habes, Jimit Doshi, Dhivya Srinivasan, Elizabeth Mamourian, Vishnu Bashyam, Ilya M. Nasrallah, Theodore D. Satterthwaite, Yong Fan, Lenore J. Launer, Colin L. Masters, Paul Maruff, Chuanjun Zhuo, Henry Völzke, Sterling C. Johnson, Jurgen Fripp, Nikolaos Koutsouleris, Daniel H. Wolf, Raquel Gur, Ruben Gur, John Morris, Marilyn S. Albert, Hans J. Grabe, Susan M. Resnick, R. Nick Bryan, David A. Wolk, Russell T. Shinohara, Haochang Shou, and Christos Davatzikos. Harmonization of large mri datasets for the analysis of brain imaging patterns throughout the lifespan. *NeuroImage*, 208, 3 2020.

[55] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. 2015.

[56] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. volume 13-17-August-2016, pages 1135–1144. Association for Computing Machinery, 8 2016.

[57] Scott M Lundberg, Paul G Allen, and Su-In Lee. A unified approach to interpreting model predictions.

[58] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. 4 2017.

[59] H. P. Young. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14:65–72, 6 1985.

[60] Pol Ferrando. Understanding how lime explains predictions, 2018.

[61] Madhura Ingalhalikar, Sumeet Shinde, Arnav Karmarkar, Archith Rajan, D. Rangaprakash, and Gopikrishna Deshpande. Functional connectivity-based prediction of autism on site harmonized abide dataset. *IEEE Transactions on Biomedical Engineering*, 68:3628–3637, 12 2021.

[62] Hao Guan, Yunbi Liu, Erkun Yang, Pew-Thian Yap, Dinggang Shen, and Mingxia Liu. Multi-site mri harmonization via attention-guided deep domain adaptation for brain disorder identification. *Medical Image Analysis*, 71:102076, 7 2021.

[63] Anusha Kamath, Sparsh Gupta, and Vitor Carvalho. Reversing gradients in adversarial domain adaptation for question deduplication and textual entailment tasks, 2019.

[64] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.

[65] Sara Saponaro, Alessia Giuliano, Roberto Bellotti, Angela Lombardi, Sabina Tangaro, Piernicola Oliva, Sara Calderoni, and Alessandra Retico. Multi-site harmonization of mri data uncovers machine-learning discrimination capability in barely separable populations: An example from the abide dataset. *NeuroImage: Clinical*, 35:103082, 2022.