PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF COMPUTER SCIENCE
POSTGRADUATE PROGRAMME IN COMPUTER SCIENCE

IDENTIFICATION OF AUTISM DISORDER THROUGH FUNCTIONAL MRI AND DEEP LEARNING

ANIBAL SÓLON HEINSFELD

Dissertation presented as partial requirement for obtaining the degree of Masters in Computer Science at Pontifical Catholic University of Rio Grande do Sul.

Advisor: Prof. Dr. Felipe Meneguzzi
Co-Advisor: Prof. Dr. Alexandre Rosa Franco

Porto Alegre
2017

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

STATEMENT OF PRESENTATION OF THE DISSERTATION

Dissertation entitled "Identification of Autism Disorder Through Functional MRI and Deep Learning" presented by Anibal Sólon Heinsfeld as part of the requirements to achieve the degree of Master in Computer Science approved on March 28, 2016 by the Examination Committee:

Prof. Dr. Felipe Rech Meneguzzi — PPGCC/PUCRS
Advisor

Prof. Dr. Alexandre Rosa Franco — PPGCC/PUCRS
Co-Advisor

Prof. Dr. Rodrigo Coelho Barros — PPGCC/PUCRS

Prof. Dr. Richard Cameron Craddock — Child Mind Institute

Prof. Dr. Paulo Martins Engel — UFRGS

Ratified on 09/03/2017, according to Minute No. 02 by the Coordinating Committee.

Prof. Dr. Luiz Gustavo Leão Fernandes
Graduate Program Coordinator

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320–3621
E-mail: ppgcc@pucrs.br

Ficha Catalográfica

H471i Heinsfeld, Anibal Sólon

Identification of Autism Disorder through Functional MRI and Deep Learning / Anibal Sólon Heinsfeld . – 2016.
73 f.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Felipe Rech Meneguzzi.
Co-orientador: Prof. Dr. Alexandre Rosa Franco.

1. Deep learning. 2. Neuroimaging. 3. Autism spectrum disorder. I. Meneguzzi, Felipe Rech. II. Rosa Franco, Alexandre. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).

I dedicate this work to my family and my friends, for being there for me throughout the entire master program.

I dedicate this work especially for my wife, Bruna Damiana, for her lovely and encouraging spirit.

ACKNOWLEDGMENTS

I would like to acknowledge the support of the Brain Institute of Rio Grande do Sul (InsCer), the LABIMA team (Image Processing Laboratory) and the LSA team (Autonomous Systems Laboratory) from PUCRS. I would also like to thank Felipe Meneguzzi, Alexandre Franco, and Augusto Buchweitz, for their invaluable guidance.

I acknowledge the effort from ABIDE team to build and openly publish their high-quality dataset. I believe that initiatives like ABIDE bring traction to scientific research.

Finally, I acknowledge NVIDIA donating Tesla K40 GPUs to our research project. Without such hardware resources, our research would have taken substantially more time to complete and would not have yielded the results that it did. And I acknowledge the LISA lab, for open sourcing their deep learning framework Pylearn2.

IDENTIFICATION OF AUTISM DISORDER THROUGH FUNCTIONAL MRI AND DEEP LEARNING

ABSTRACT

Autism Spectrum Disorders (ASD) comprise a range of neurodevelopmental disorders, characterized by social deficits and communication difficulties, repetitive behaviors, and cognitive delays. The diagnosis of ASD is largely based on behavioral measurements, which can be time-consuming and relies on the patient cooperation and examiner expertise. In order to address this limitation, we aim to investigate neural patterns to help in the diagnosis of ASD. In this dissertation, we use deep learning techniques to extract robust characteristics from neuroimages of autistic subject brain function. Since neuroimage contains about 300,000 spatial points, with approximately 200 temporal measurements each, deep learning techniques are useful in order to extract important features to discriminate ASD subjects from non-ASD. By using denoising autoencoders, a specific deep learning technique that aims to reduce data dimensionality, we surpass the state-of-the-art by achieving 69% of accuracy, compared to 60% using the same dataset.

Keywords: Deep learning, neuroimaging, autism spectrum disorders.

IDENTIFICATION OF AUTISM DISORDER THROUGH FUNCTIONAL MRI AND DEEP LEARNING

RESUMO

O Espectro Autista (EA) compreende uma série de desordens no desenvolvimento neurológico, caracterizado por deficiências sociais e dificuldades de comunicação, comportamentos repetitivos e atrasos cognitivos. Atualmente, o diagnóstico do EA é amplamente baseado em medições comportamentais, que pode ser demorado, e depende da cooperação do paciente e da experiência do examinador. Para mitigar esta limitação, investigamos padrões neurais que ajudem no diagnóstico de desordens do EA. Nesta dissertação, usamos técnicas de *deep learning*, a fim de extrair características robustas de neuroimagens de pacientes com autismo. Neuroimagens contêm cerca de 300.000 pontos espaciais, com aproximadamente 200 medições cada. As técnicas de *deep learning* são úteis para extrair características relevantes que diferenciam autistas de não-autistas. Ao utilizar *denoising autoencoders*, uma técnica de *deep learning* específica que visa reduzir a dimensionalidade dos dados, nós superamos o estado da arte, atingindo 69% de acurácia, comparado com o melhor resultado encontrado na literatura, com 60% de acurácia.

Palavras Chave: Deep learning, neuroimagem, espectro autista.

LIST OF FIGURES

LIST OF SYMBOLS

LIST OF TABLES

CONTENTS

1. INTRODUCTION

Autism Spectrum Disorders (ASD) comprise a range of conditions classified as neurodevelopmental disorders that affect the normal development of social and communication skills. Formerly considered rare, ASD are now reported to occur in $\sim 1\%$ of children [DYL+14]. Individuals with ASD have difficulties with social interaction, display problems with verbal and nonverbal communication, and exhibit repetitive behaviors or narrow, obsessive interests [LMS09]. Since ASD lies on a spectrum, the severity of the disease may vary within a population, thus making the data highly heterogeneous [Men11].

Neurological disorder studies have found that some dysfunction in the brain is not due to the alteration of a specific brain area, but due to the failure of diffuse systems [Gre08]. Considering the need to find patterns in patients to reach reliable prognoses of these disorders, we rely on research results that have found evidence of anomalies in patients' brains with ASD [ADF+11, NZF+13, DYL+14, PBM15]. The evidence of different properties in different phenotype subgroups, defined as a biomarker [APF+13], is obtained using modern neuroimaging techniques, which collect precise images of brain characteristics, discovering unnoticed anatomical and functional anomalies. Precise biomarkers can guide studies into specific neural circuits, developmental windows and genetic factors that may result in improved treatments, such as pharmacological intervention [ST10]. The advent of open datasets for neuroimaging is a significant step towards neurological biomarker research since it enables worldwide research and their proper reproduction [DYL+14]. Due to the high variance intrinsic to the current neuroimaging techniques, caused by different phenotypes and even different scanners, the use of open datasets can lead to robust biomarkers [APF+13].

Functional Magnetic Resonance Imaging (fMRI) is currently the most used technique to search for novel biomarkers in brain function. fMRI is a neuroimaging technique that indirectly detects neural activity using magnetic properties peculiar to the brain [Bux02]. We analyse how brain regions work together (correlate) by using neuroimages from a specific type of experimental design: the resting-state. In this experimental design, the scanned subject does not perform any particular task, and the spontaneous fluctuations in the brain is captured by the MRI.

We applied deep learning techniques to create an fMRI-data driven model to differentiate accurately ASD from non-ASD subjects, in order to bring a proper diagnosis and, for ASD subjects, we envisage the referral to a qualified professional for evaluation and treatment. To create a diagnosis model, we use the Autism Brain Imaging Data Exchange (ABIDE) dataset. We extract and analyze characteristics from the deep learning model to find biomarkers for ASD. By using deep learning models, we overcome the current state-of-the-art [NZF+13]. This improvement demonstrates the potential of such new techniques for neuroscience and brings about an analysis of which brain regions are most relevant for the differentiation between mental states or neuropsychological diseases.

1.1 Contribuitions

The diagnosis of neurodevelopmental disorders is largely based on patient's behavior [WBB+15]. The problem with this type of diagnosis is that its accuracy depends upon a number of variables that are not easy to control, such as examiner skill and patient collaboration [ERRJ+10]. Thus, it is important to apply measurable and reliable diagnosis methods, based on biomarkers.

Resting-state functional neuroimaging is a specific type of neuroimaging technique that requires minimal collaboration of the patient. Resting-state fMRI can even be performed on patients that are asleep or under different levels of anesthesia [dIIVMMJ+13]. However, this method yields a massive amount of measurements, many of which do not contribute towards diagnosis.

The main goal of this work is to accurately identify a specific neurodevelopmental disorder—autism—among subjects. Based on scarce results of deep learning and fMRI usage [PHS+14, KSN+15], our main objective is to use deep learning methods to use high-level abstractions of fMRI data as discriminant to diagnose autism. In this context, we propose the following research question:

- Is it possible to use deep learning algorithms to identify neurodevelopmental disorders from fMRI data?

Thus, the main contribution of our work is to create a deep learning model used with neuroimaging techniques that can automatically filter the data to diagnose autism based on the filtered measurements. We generate a deep learning model that classifies data in a way that surpasses the current state of the art, and may eventually contribute towards a diagnosis model. By analyzing filtered measurements that most contribute to the diagnosis, we found patterns that may become biomarkers that help in ASD identification.

Therefore, the contributions of this work are:

- Create a deep learning model using resting-state fMRI data to discriminate subjects with and without ASD.
- Identify brain patterns that may lead to biomarkers. ASD is not characterized by a dysfunction in a single brain region, but in a diffuse network. Thus, we identified a potential network of regions in neuroimaging data.
- Show the applicability of deep learning methods in neuroscience research projects.

1.2 Publications

During the master program, we work in the following publications:

- Craddock, R Cameron; Margulies, Daniel S; Bellec, Pierre; Nichols, B. Nolan; Alcauter, Sarael; Barrios, Fernando A.; Burnod, Yves; Cannistraci, Christopher J.; Cohen-Adad, Julien; De Leener, Benjamin; Dery, Sebastien; Downar Jonathan; Dunlop, Katharine; Franco, Alexandre R.; Froehlich, Caroline Seligman; Gerber, Andrew J.; Ghosh, Satrajit S.; Grabowski, Thomas J.; Hill, Sean; Heinsfeld, Anibal Sólon; Hutchison, R. Matthew; Kundu, Prantik; Laird, Angela R.; Liew, Sook-Lei; Lurie, Daniel J.; McLaren, Donald G.; Meneguzzi, Felipe; Mennes, Maarten; Mesmoudi, Salma; O'Connor, David; Pasaye, Erick H.; Peltier, Scott; Poline, Jean-Baptiste; Prasad, Gautam; Pereira, Ramon Fraga; Quirion, Pierre-Olivier; Rokem, Ariel; Saad, Ziad S.; Shi, Yonggang; Strother, Stephen C.; Toro, Roberto; Uddin, Lucina Q.; Van Horn, John D.; Van Meter, John W.; Welsh, Robert C.; Xu, Ting. (2016). "Brainhack: a collaborative workshop for the open neuroscience community". GigaScience, 5(1), 16. http://doi.org/10.1186/s13742-016-0121-x

- Heinsfeld, Anibal Sólon; Franco, Alexandre R.; Buchweitz, Augusto; Meneguzzi, Felipe. (2016) "NeuroView: a customizable browser-base utility". Manuscript submitted for publication.

- Pereira, Ramon Fraga; Heinsfeld, Anibal Sólon; Franco, Alexandre R.; Buchweitz, Augusto; Meneguzzi, Felipe. (2016) "Detecting task-based fMRI compliance using plan abandonment techniques". Manuscript submitted for publication.

- Heinsfeld, Anibal Sólon; Franco, Alexandre R.; Buchweitz, Augusto; Meneguzzi, Felipe. (2016) "Identification of brain states using unsupervised learning algorithms: a study of patterns in autism spectrum disorder and fMRI data". 2016. Unpublished manuscript.

2. BACKGROUND

Neurodevelopmental disorders are a group of conditions that manifest in the growth and development of the brain [GR10]. These disorders are characterized by developmental deficits that vary from some limitations in learning to global impairments of intelligence or social skills and are associated with a wide variety of mental, emotional, behavioral, and physical factors. Common neurodevelopmental disorders include: communication, speech, language and executive function disorders, such as dyslexia, autism spectrum disorder, and attention deficit hyperactivity disorder. It also comprises genetic disorders, such as the Down syndrome.

The diagnosis of neurodevelopmental disorders is mostly based on subject's behavior [ERRJ+10]. Pattern recognition techniques have been applied to identify biological characteristics that could be useful for a correct diagnose [WBB+15]. These characteristics extracted by pattern recognition techniques usually yield biomarkers. Biomarkers are physiological, pharmacological or biochemical measurement indicators of the presence of some disease state, which can be measured accurately and reproducibly [SKE03]. In order to use biomarker in clinical diagnosis, they must be well characterized and repeatedly validated to predict correctly relevant clinical outcomes. Thus, it is important to continuously reevaluate biomarkers, so they remain reliable for potential variations in the disease. Understanding the relationship between measurable biological processes and clinical outcomes is not only useful for diagnosis, but it also plays a critical role in improving drug development processes. The overall research time can be drastically reduced by understanding the process of the condition since the drug can be directed to avoid or change the development of the condition.

Pattern recognition techniques are promising to extract biomarkers from a combination of metrics since neurological disorders are unlikely to stem from a single biological process but from several processes [WBB+15]. In order to compose biomarkers for neurological disorders, neuroimaging techniques, such as volumetric MRI, functional MRI (fMRI) and diffusion tensor imaging (DTI), are widely used to correlate altered cortical tissue, functional and physical characterization with subject's disease [APF+13]. These techniques are used to identify irregularities in neuroimages, which can be used to predict a diagnosis based on image features, i.e., measurable properties.

2.1 Neuroimaging

Neural activity is hard to measure directly without placing electrodes within the brain. To fulfill the need to record the brain activity, there are techniques developed to avoid surgical interventions or to expose the brain to harmful radiation. Neuroimaging is a set of non-invasive techniques for the acquisition of images of different brain structures. Magnetic Resonance Imaging (MRI) is an example of a non-invasive technique [HSM04]. As the name implies, it uses magnetic fields to create three-dimensional images of biological tissue. During the imaging process, a strong and constant magnetic field, called B0, is generated by a scanner to magnetize the patient's tissues.

Tissues magnetize because they consist of protons, and protons behave as small magnets and align with the B0 magnetic field. The field magnetization is changed throughout the scan, but tissue magnetization does not change at the same rate for all tissues. This rate difference allows the scanner to distinguish between tissues, as a consequence of the contrasting magnetization between tissues. The strength of B0 is expressed in units of Tesla and for imaging humans, it is typically between 1 to 7 Tesla. By way of comparison, the Earth's magnetic field ranges from 0.00025 to 0.00065 Tesla.

After aligning protons to the B0, the scanner emits short bursts of radio frequency waves (called pulse sequence), seeking to disturb the alignment of protons. This disturbance occurs through the transference of energy from the electromagnetic radio frequency waves to the protons. As soon as the radio frequency waves are turned off, protons tend to realign to the B0 field and to dissipate the stored energy to their neighbors. This energy is detected by the scanner, generating the activation signal. Usually, hydrogen protons are targeted in the process because human tissues are largely composed of them. The MRI scanner can detect different tissue properties depending on the pulse sequence used during the scan which generates different image contrasts. Different pulse sequences are developed to identify different health conditions, such as tumors, abnormalities in blood vessels and bone damage. The ability to examine multiple biologically interesting properties of tissue makes MRI an extraordinarily flexible and powerful clinical tool. The analysis of brain's structure provides a great understanding of neurological disorders by identifying patterns that refer to the condition. However, structural studies do not reveal physiological changes that may occur during the brain activity. A different approach, called Functional Magnetic Resonance Imaging, is used to address this limitation.

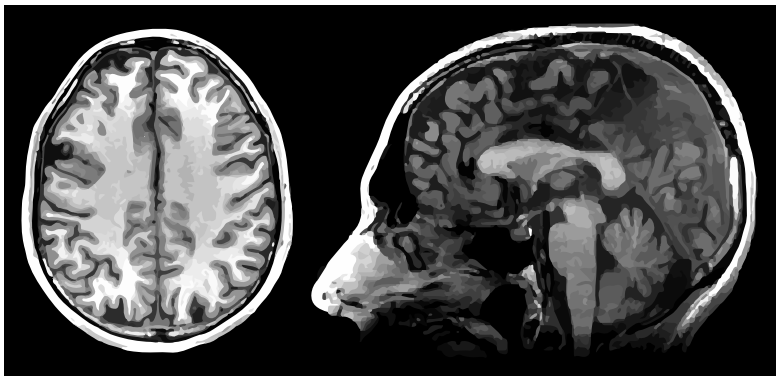


Figure 2.1: An MRI scan from Anibal. Besides the brain, it captures signals from other tissues, such as the skull.

2.1.1 Functional Magnetic Resonance Imaging

Functional Magnetic Resonance Imaging (fMRI) is an imaging technique within MRI [HSM04]. It measures brain activity based on an indirect association of neuronal activation and blood flow, in order to study how brain functioning changes across time. As neurons do

not have reserves of oxygen and glucose within themselves, they require these resources from the body through a rapid process of glucose and oxygen distribution via the blood flow. Haemodynamic Response (HR) allows the rapid deliver of nutrients to active neuronal tissues. Ogawa et al. [OL90] observed that hemoglobin has different magnetic properties when they assume different levels of oxygenation, discretized as oxygenated and deoxygenated hemoglobin, allowing the distinction between both states. By detecting these levels of magnetization, it is possible to see the brain regions that require oxygen through the HR, which allows us to infer an activation in the region based on its metabolic requirements. Thus, the magnetic signal variation from oxygen concentration is referred to as the blood oxygen level–dependent (BOLD) signal. fMRI generates a four-dimensional functional neuroimage: a time sequence of three-dimensional images of the brain. This sequence of successive three-dimensional images is made over a time interval, thus being defined as a time series. Each three-dimensional point of the image is called a voxel, or a 3D pixel. So, for each voxel, the fMRI contains a time series of BOLD signals that occurs at voxel's position.

2.1.2 Image Preprocessing

The BOLD signal arises from multiple sources. Generally, the scientific interest is in the part of the signal that arises from neural activity. The separation of signal from noise is an important step for scientific experiments. To properly capture the desired BOLD signal, data must pass through a preprocessing phase to remove such interference. It is an important step because analysis can lead to erroneous results if data does not represent what an experiment designer intends to model, e.g., the expected brain activation for a given task.

There are several origins of interference, and many ways to deal with them [HSM04]. Firstly, the intrinsic thermal disturbance that consists of heat-related motion of electrons contained in the scanner, in the patient and in the environment that surrounds them. Secondly, the motion noise caused by involuntary movements of the patient's head during scan. As our body continues to work, we are not absolutely immobile even if we are lying down. Our heart still beats through muscle contractions, blood pulses through arteries and veins, and our lungs expand on breathing. These movements can displace data points from their original places, deforming the resulting time series. Finally, noise can arise from neural variability. Activation that is not related to the task is common during scanning. For example, the scanner produces a loud sound, that activates the auditory receptors of patients. If we are trying to measure motor responses to visual stimulus, the activation of brain areas related to sound recognition does not provide relevant information.

To measure brain signals, such interference must be removed. Three techniques are usually applied. First, the data must go through a process called slice timing correction. This technique aims to correct different slice scanning times from pulse sequence through interpolation since the whole brain is not scanned once but with a series of successive slices. The interpolation occurs between slices of same spatial position, but from a different time.

fMRI data contains information from other tissues, such as data generated from the skull or from the fat that envelops the brain. Because these surrounding tissues have magnetic properties, the generated signal also includes data from them. So, as the second technique, parts that do not generate neural activity are removed through signal processing methods.

The third technique is head motion correction. This step aims to adjust the images, so the brain is always in the same position to maintain activation data with their respective voxels. A common approach is to align functional volumes with a high-resolution structural image of the subject's brain. The process used to align brain images is called coregistration. Alternatively, the motion correction can be mitigated by using motion parameters: head translation and rotation. So the volumes are aligned based on these parameters.

In order to compare brain activations within a group, each fMRI is coregistred to a specific brain template. This technique is called spatial normalization. Human brains are variable in their size and shape. Thus, this removes the variability associated to subjects different brain sizes, and it becomes possible to determine region correspondence from brain to brain.

These are some basic technique toward data quality used in neuroimaging. The interpretation of functional measures is difficult unless the interference is removed [HSM04, Chapter 8]. Although it is not currently possible to entirely remove all interference source, application of noise removal techniques increases confidence in fMRI results and interpretations.

2.1.3 Experimental Design

In fMRI studies, a paradigm, or experimental design, is the organization of an experiment to allow effective testing of the study hypothesis [Bux02]. The goal of a paradigm is to induce mental states of a subject through specific stimulus modalities. For psychopathologies, paradigms are designed to stimulate what may bring their manifestation in the brain (e.g. simulation of social interactions for autism), so neural activity may appear in brain regions associated with the pathology. In general, studies address two types of paradigms: task-based fMRI and resting-state fMRI. Task-based fMRI is used to detect neural activity in response to certain experimental conditions i.e. tasks or stimuli relevant to the research hypothesis. For example, in order to activate language networks of the brain, a paradigm might contain reading tasks during the scan. If the hypothesis is about brain networks for foreign languages, a paradigm can interleaves reading of foreign and native words, in order to see the differences between mental states caused by each stimulus.

Two different types of presentation design are typically employed for conditions in task-based fMRI: block and event-related.

Block design defines the ordering of events, in which events are grouped by their conditions, as illustrated in the left column of Figure 2.2. The first row represents the moment that the subject will receive the stimuli. Each dot is a stimulus, and its color represents its type. In the language example, foreign and native words will not be interleaved in block design: they will be shown to the

subject in groups (or blocks). So, the blue stimuli might be foreign words, and red stimuli might be native words. The second and third row represent the expected measurement from a specific voxel given the stimulus type. The last row represents the measured activation. Block design has the property of insensitivity to the shape and timing of HR through the BOLD signal. Inasmuch as the stimulus is presented continuously, BOLD signal deviates from a normal HR since continuous signals of HR overlap. Thus, the BOLD signal rises rapidly at the onset of the task, then remains at a plateau until the end of the block. In this form, the BOLD signal is simple to analyze, since it presents a trapezoidal shape. For example, in a simple analysis, the BOLD signal can be converted to a binary wave to represent if a condition affects a voxel or not. However, since HR are averaged in an entire block, it is not possible to distinguish the BOLD signal curve for each stimulus, which may be required for some experiment hypotheses, such as in event-related design.

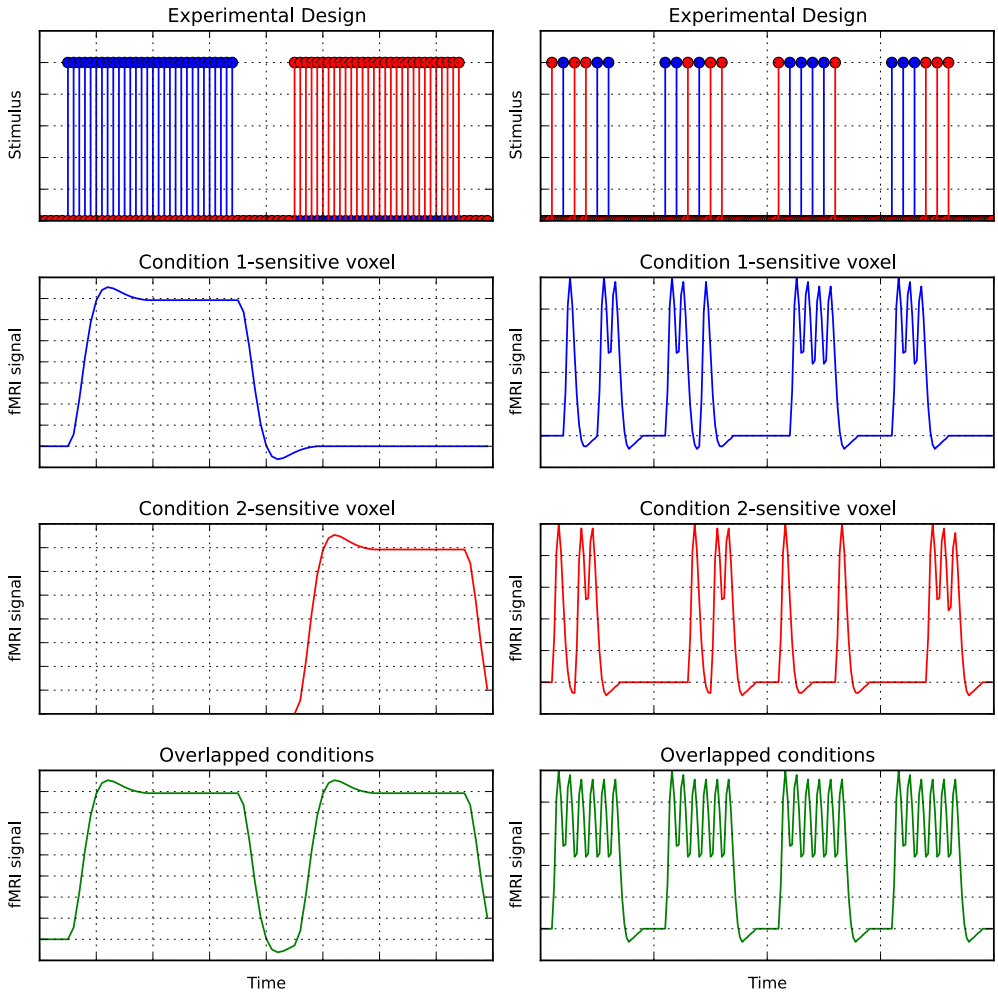


Figure 2.2: Experimental designs with two conditions. Left column: Block design. The overlap of signal creates a trapezoidal shape of BOLD signal. Right column: Event-related design. It assumes random presentation of conditions, and it is possible to estimate the BOLD signal for each stimulus.

In event-related design, unlike block designs, the different conditions are usually presented in a random order rather than an alternating pattern, as illustrated in the right column of Figure 2.2. This design interleaves the conditions and presents them during a short period (two to ten

seconds), such as in the first row of Figure 2.2. Since the brain never ceases activity, paradigms must contain a baseline event, in which the subject does not execute tasks nor receive stimuli. Commonly, the baseline signal is subtracted from stimulus signal, in order to accentuate activations derived from tasks. A difficulty of task-based paradigm is its analysis. Due to the closely spaced events, the captured neural activity overlaps substantially. In the last row of Figure 2.2, it is not possible to distinguish which stimulus type generates each peak. To properly analyze it, it is necessary to separate the activation model for each stimulus by using techniques, such as general linear models [Bux02, Chapter 15].

In contrast to task-based fMRI, resting-state fMRI is a paradigm that can be used to analyze brain function when a subject is not performing any specific task: the subject is asked to stare at a fixation cross (illustrated by Figure 2.3) for at least 6 minutes and not to think about anything in particular during this time. Biswal [BYHH95] detected that brain regions that are simultaneously active during a given task are correlated in the absence of a task. Until then, low-frequency BOLD fluctuations were discarded as noise in task-based fMRI scans. Since Biswal's observations, new fMRI studies use the resting-state approach, and their analysis relies on the relation between spontaneous low-frequency BOLD fluctuations and the intrinsic activity of the brain. The application of this approach to the study of certain disorders led to the conclusion that certain areas of the brain are correlated with specific higher executive function, such as language and memory, over large-scale brain networks.

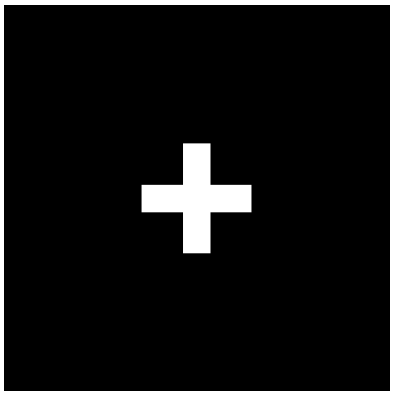


Figure 2.3: Example of fixation cross that the subject must stare during resting-state fMRI.

The resting state approach eliminates limitations and difficulties during scan and analysis since it has no cognitive demands. It enables the analysis of patients that are incapable of performing a functional task. Such advantage allows previously excluded subjects to perform fMRI mappings, such as children, patients with cognitive impairments or other difficulties. Since spontaneous fluctuations in the BOLD signal are still present under conditions of sleep and different levels of anesthesia, this technique can be performed in unquiet subjects and young children under sedation [dIIVMMJ$^+$13]. A second advantage is that, since this technique does not focus on a specific region of interest, a single acquisition can be used to study several different brain networks. In contrast, task activations require specific acquisitions for each studied function. However, since subjects are

frequently stimulated in task-based paradigm, it may be less tedious in comparison to resting-state fMRI.

An approach to analyze resting-state fMRI data is the functional connectivity [vdHH10]. Functional connectivity is the measurement of brain activation patterns of anatomically separated regions. Since the brain is a large and complex network, the measurement of how regions work together (or apart) provides important insights of the core organization of the brain. In order to identify how functional brain areas are connected, we correlate every time series, resulting in the connectivity matrix, as shown in Figure 2.4. Each cell in the connectivity matrix contains the Pearson correlation coefficient that measures how much two time-series correlate, ranging from 1 to -1. Values close to 1 indicate that time series correlate, and values close to -1 indicates that time series are anti-correlated. Thus, brain regions that correlate are over-connected, and brain regions that anti-correlate are under-connected.

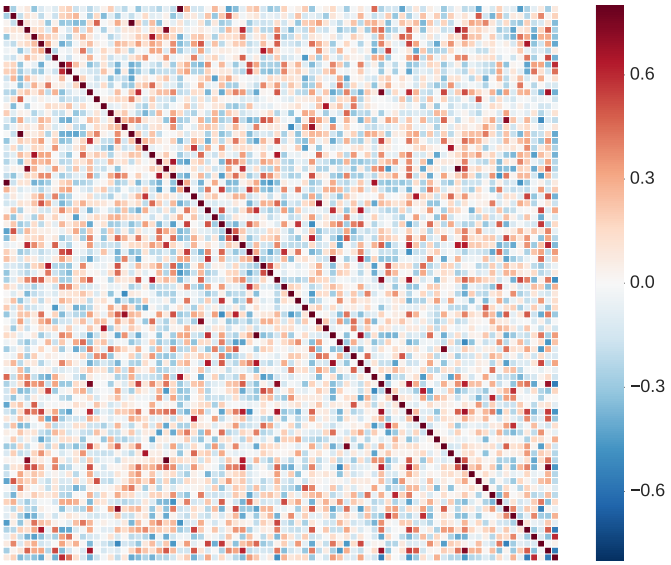


Figure 2.4: Connectivity matrix of a resting-state neuroimage. Colorbar represents Pearson correlation coefficient, that ranges from -1 to 1.

2.2 Autism

Autism spectrum disorders (ASD) are a range of neurodevelopmental disorder that affects the normal development of social and communication skills. ASD are highly genetic, which comes more from heredity than other factors such as the environment [Mil11]. Like many developmental disorders, ASD is diagnosed mostly by behavioral criteria [ERRJ+10]. The behavioral diagnosis of ASD is however often time-consuming and can be problematic since it relies on the patient cooperation. Gold standard diagnostics for ASD, such as the *Autism Diagnostic Observation Schedule* (ADOS) [LRG+89] and the *Autism Diagnostic Interview* (ADI-R) [LRL94] depend on the skill of the examiner and further information provided by relatives.

To help ASD diagnosis and further intervention, several studies describe genotypes and neurological characteristics [MW07]. In general, most of the neurodevelopmental disorder studies were limited to univariate models of measures of global change involving metrics such as overall brain volume [Men11]. Since ASD neuroimaging studies have increasingly converged to abnormalities in connectivity among brain regions, rather than local functional or structural abnormalities [WBB+15], we focus on methods for connectivity analysis, specifically functional connectivity. The heterogeneity of ASD presents great challenges for researchers, but evidence across neuroimaging modalities is converging to aberrant functional connectivity patterns [MSK+11]. When compared to non-ASD subjects, functional connectivity in task-based fMRI is consistently under-connected for ASD, between regions involved in several cognitive tasks including visual attention, language, and memory [CKM13]. However, studies of resting-state functional connectivity show results of both under and over-connectivity between brain regions [DYL+14].

3. MACHINE LEARNING

Machine learning is a subfield of computer science and statistics that aims to build algorithms that improve performance from data. Each machine learning technique uses specific mathematical models (i.e. a mathematical representation of systems behavior) to store and apply what it has learned. These techniques can improve models performance by receiving more and more data to learn from, in order to achieve a particular task. Primarily, the set of techniques of machine learning is used in problem domains in which developing a solution is infeasible with explicit programming or even manually [Sam59]. Since patterns may be hard to find by humans when there is a large quantity of data to analyze, machine learning algorithms can be useful to look for rules and exceptions, and even replace rule-based programs.

3.1 Machine Learning Basics

Mitchel provides one possible formal definition of machine learning [Mit97]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". For example, in a machine learning program that identifies fraudulent transactions from an account holder, "identify fraudulent transactions" is the task T, transactions are the experience E, and the identification accuracy is the performance measure P. In other words, the program will identify fraudulent transactions and improve itself based on how many transactions (fraudulent or not) it analyzes. However, this is a simple and abstract machine learning problem. To proper understand machine learning, it is necessary to understand the components T, P, and E.

First, the tasks: T. The machine learning task is not to learn, but to execute the task previously (or even simultaneously) learned by using a learning process. For example, in our fraud detection example, the task T does not learn how to accurately detect fraud, but the action of detect fraud. Usually, machine learning tasks define which technique should be used to process the experiences. Examples are experiences originated from the task domain. By using examples, the machine learning techniques can learn using quantitatively measured characteristics of an example. These characteristics are called features. In the fraud detection problem, the account holder's transactions can be examples for the learning process, and the transaction value and date can be used as example features. Many types of tasks can be solved with machine learning. Common machine learning tasks include:

- classification tasks, in which the machine learning model is expected to return which category some example belongs to e.g. identify if an e-mail is a spam or not based on the presence of certain words.

- regression tasks, in which the machine learning model is expected to generate a real value for some example e.g. informing how much a person can borrow from a bank loan given its financial situation.

- clustering tasks, in which the machine learning model tries to group examples into groups by their feature similarities.

Secondly, the performance measure: P. For a machine learning technique to know how well it is performing a given task T, it must have a well-defined measure of its performance. The measure is strictly related to the task. For example, a common performance measure for classification tasks is accuracy. Accuracy measures the proportion of examples that the model correctly identifies. Otherwise, for regression tasks, the accuracy does not bring useful information to regression techniques, since it only informs if the model is correct or not. For regression tasks, a better performance measure is the RMSE (root mean square error). RMSE is the square root of the average squared error between the value predicted by a model and the value actually observed for an example. It not only informs if the model is wrong but also informs how distant the output value is from the expected value. Usually, the performance of a machine learning algorithm measures how well it performs on new experiences, in order to evaluate the generalization of the model. The better capacity of generalization means that the model still can achieve its task with good performance even in the presence of some variation in new unseen examples.

Finally, the experience E (e.g. a set of examples), which is used for the model learning. Based on Mitchel's definition, the more experience is available for learning, the more the model will learn and achieve better performance. Machine learning assumes three broad categories of learning process [RN10, Chapter 18]: supervised learning, unsupervised learning, and reinforcement learning. This proposed division is based on the experience that is available during learning, but each learning process has its particularities. Based on the problem domain and available data of this work, we focus on supervised and unsupervised learning.

Supervised learning algorithms attempt to adjust a model from a given training data and its known response. Each example is a pair consisting of a set of features and a corresponding output value. By this means, examples for supervised learning can be described in terms of ($\mathbf{x}$, y) pairs, where $\mathbf{x}$ is a feature vector and an associated value y, that is the corresponding value that the model should output given $\mathbf{x}$. Thus, supervised learning involves observing several examples ($\mathbf{x}$, y) and adjusting the model to predict y from x, usually by estimating $p(y|\mathbf{x})$ (commonly represented as $\hat{y}$). This type of learning comprises classification and regression.

Unsupervised learning algorithms intend to recognize patterns from data without an expected response. In contrast to supervised learning, there are no mapped outputs y to the collection of inputs $\mathbf{x}$. Instead, unsupervised learning attempts to find hidden structure in data, either in an intra-example or extra-example way. Intra-example learning looks for hidden features in examples, based on their original features. These hidden features are defined as latent features. So, this learning method generates a model to compute latent features from data. An example of an intra-sample

unsupervised technique is the Principal Component Analysis (PCA). Usually, extracted latent features are used in subsequent tasks, such as classification, since they may help to discriminate between categories. Extra-example algorithms attempt to group examples based on the similarity of their features. These algorithms generate a model that defines in which group the example $\mathbf{x}$ belongs. For extra-sample unsupervised learning, there are clustering algorithms such as k-Means [HW79].

3.2 Learning and Validating Models

The challenge in machine learning is that the model must perform well on unseen examples. To evaluate a model, the available dataset must be split into two subsets: training and test sets, as illustrated by Figure 3.1.

Figure 3.1: Split of available data into training and test set.

The training set is used during learning: the training phase. During the training of a machine learning model, it is possible to compute the error of the training set over the model and minimize this error. This error is called training error. However, machine learning tasks must generalize to be useful. To evaluate the generalization of a model, the test set must be used to measure the test error (or generalization error): the error of the test set over the model. By this means, models can fall into two unwanted scenarios: underfitting and overfitting. Underfitting is a scenario where training error is not sufficiently low. Overfitting is a scenario where the training error decreases at the expense of model generalization. Generally, both scenarios occur because of the model's complexity. If a model has too many parameters, much more that the number of examples in the training dataset, the model can memorize the training set (even its fluctuations): this leads to overfitting. Typically, models in this scenario fail drastically when evaluated with the test set. Otherwise, in underfitting scenarios, if a model has a low number of parameters, it may struggle to fit the training set and becomes unable to solve complex tasks. Thus, it is important to control the number of model's parameters, in order to maintain a good trade-off of how much the model will learn from the training set.

To guide the learning algorithm to reduce both training and generalization error, a common data-driven technique is the holdout [HTF01]. By splitting the training set into two subsets: training and validation set, the holdout technique gives an estimate of how accurately a model will perform in practice by using the validation set during the training phase.

It mitigates overfitting scenarios and, by evaluating the validation set, the learning technique can adjust its parameters in order to get a better approximation of unseen data. The technique

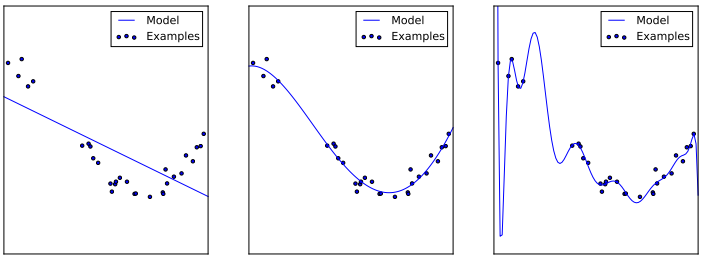


Figure 3.2: Underfitting and overfitting for a classification task.

Figure 3.3: Split of training set into training and validation set.

parameters are called hyperparameters, that control how the technique learn. However, there is still a risk of overfitting on the test set because the hyperparameters can be tuned until the model performs optimally even for the test set. Thus, cross-validation techniques are used to avoid overfitting. For the k-fold technique, the training set is divided into k smaller sets called folds, and these folds are rotated as train and validation set.

Figure 3.4: k-folds, where $k = 4$.

For each fold, the model is trained using the remaining folds and validated with this separate fold. Thus, the performance of models by using cross-validation is the average of the performance values computed in the rotation.

3.3 Machine Learning in Neuroscience

Several research fields actively apply machine learning techniques, such as: face recognition from digital images, product recommendation, and stock trading. In neuroscience, a common application of machine learning techniques is the identification of disorders (or diseases) by comparing fMRI metrics of subjects with and without the disorder. This classifier can be used in expert systems to diagnose a patient with a particular disorder. However, beyond a possible disorder that the patient might have, these techniques can also be used to answer the question of how disorders manifest themselves in the brain. This information is useful for research, to study the evolution of a disorder, to improve its understanding and develop new treatments.

Neuroimaging data is highly dimensional i.e. it has a large number of features in each image. Some studies use whole feature set from brain images, but a common practice is to reduce the number of features since many features may not contribute for the condition prediction. In order to reduce the number of features, feature selection methods take place. Usually, feature selection is performed by an expert, that pre-selects features based on his domain knowledge [WBB+15]. However, a variety of automatic feature selection methods are used, to discover other patterns and yield more relevant features.

In neuroscience, a commonly used technique is the support vector machine (SVM) [CHHM09]. SVM [Bis06] is a supervised learning technique developed by Vapnik and Cortes [Vap95]. It attempts to determine the separation of examples into categories, choosing a margin that produces the largest distance between categories of examples. In a supervised problem of binary classification, SVM chooses from among the infinite number of divisions, the one that maximizes the margin between the two categories. Thus, the chosen division is the one that has the greatest margin between the two categories, where the margin is defined as the sum of the distances from a hyperplane to the closest points of the two categories. Intuitively, as illustrated by Figure 3.5, the blue margin provides a higher confident classification, since examples are more distant from this decision boundary.

The technique attempts to find a hyperplane of the form

$$\hat{y}(\mathbf{x}) = W^T\mathbf{x} + b \tag{3.1}$$

where W and b define the hyperplane, and $\hat{y} \in \{-1, 1\}$ is the x class. SVM finds the hyperplane parameters W and b that yield the greatest margin by solving the problem

$$\min \frac{1}{2}\|W\|^2 \tag{3.2}$$

[1] Plot based on scikit-learn documentation, available at: http://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html

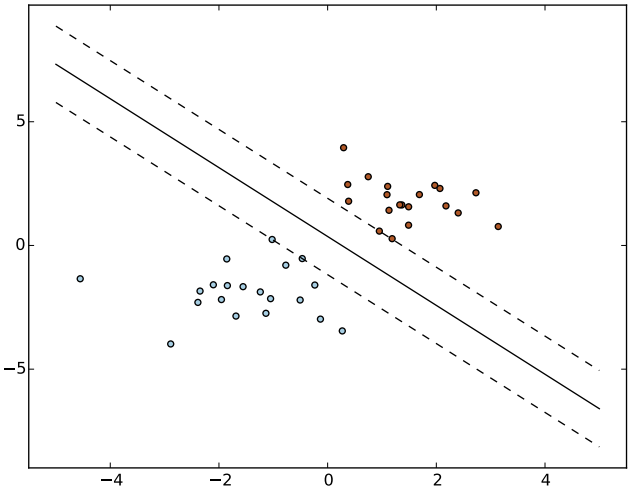


Figure 3.5: SVM maximizes the margin between classes.[1]

subject to

$$y^{(i)}(W^T\mathbf{x}^{(i)} + b) \geq 1, \forall i = 1...n. \tag{3.3}$$

where n is the number of examples in the training set. Equation 3.2 can be solved with the Lagrange function [MMR+01], that results in the problem

$$\max \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=j=1}^{n} a_i a_j y_i y_j \left\langle \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \right\rangle \tag{3.4}$$

subject to

$$\begin{gathered} a_i \geq 0, \forall i = 1...n \\ \textstyle\sum_{i=1}^{n} a_i y_i = 0 \end{gathered} \tag{3.5}$$

where a are Lagrangian multipliers. When the optimal margin is found, a becomes zero for examples distant from the margin, and, for examples that lie on the margin, a becomes non-zero. Thus, to make a prediction, the Equation 3.1 becomes

$$\hat{y} = \sum_{i=1}^{n} a_i y^{(i)} \left\langle \mathbf{x}^{(i)}, \mathbf{x} \right\rangle + b \tag{3.6}$$

Hence, if SVM found the a_i, in order to make a prediction, it is necessary to compute dot products between x and the training set. Since a_i is non-zero only for the closest examples to the margin, many terms in the sum of Equation 3.6 will be zero, and it only needs to find the dot products between x and the examples that lie on the margin. These examples are defined as support vectors.

Common problems are defined in a finite dimensional space, and their examples may not be linearly separated. Then, in order to make it easily separable, examples can be mapped into a higher-dimensional space using a function ϕ, as illustrated by Figure 3.6.

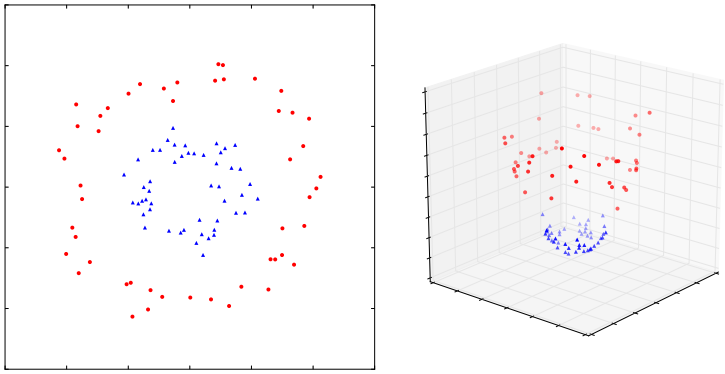


Figure 3.6: Mapping of examples in a higher dimensional space turns the problem into a linearly separable problem.

For example, a function that maps the two-dimensional data into three dimensions, illustrated by Figure 3.6, is

$$\phi(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_1^2 + \mathbf{x}_2^2 \end{bmatrix} \tag{3.7}$$

In Equation 3.4, the mapping of examples replaces the examples in the dot product

$$\max \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=j=1}^{n} a_i a_j y_i y_j \left\langle \phi(\mathbf{x}^{(i)}), \phi(\mathbf{x}^{(j)}) \right\rangle \tag{3.8}$$

However, the mapping of examples into a higher-dimensional space may be computationally expensive, specially when the original dimension of data is already large. The dot product between mappings can be replaced by similarity functions, called kernels.

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \tag{3.9}$$

By using an efficient way to calculate $K(\mathbf{x}, \mathbf{z})$, SVMs can learn in the high dimensional feature space given by ϕ, but without having to explicitly represent the mapping $\phi(x)$. It is an attractive characteristic for neuroscience since neuroimaging data has high dimensionality and, usually, few examples.

For example, the Gaussian kernel is a function that yields 1 when $\mathbf{x}$ and $\mathbf{z}$ are similar, and 0 when $\mathbf{x}$ and $\mathbf{z}$ are different. It is defined by the formula

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right) \tag{3.10}$$

and it corresponds to an infinite dimensional feature mapping ϕ, but efficiently computed. The hyperparameter σ controls how distant example $\mathbf{x}$ can be from $\mathbf{z}$, and it must be adjusted so that

K yields a similarity metric that allows some difference between examples (that possibly generalizes better), but that does not compromise the solution.

The algorithm is susceptible to outliers since they can alter the hyperplane in such way that it become non-optimal, as illustrated by Figure 3.7 In order to allow some misclassification in exchange of a better margin, two parameters are introduced to the Equation 3.2: C and ξ.

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{3.11}$$

subject to

$$\begin{gathered} y_i(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \forall i = 1...n, \\ \xi_i \geq 0, \forall i = 1...n. \end{gathered} \tag{3.12}$$

where ξ_i is the distance of the i-th misclassified example from its correct side, and C is a hyper-parameter that controls the degree to which the misclassified examples affect the solution. These parameters allow some examples to be misclassified. The value for the hyperparameter C is usually found by using cross-validation with a set of possible values. If $C = 0$, there is no penalization for misclassification. Greater values for C penalize more misclassification, so SVM becomes more susceptible to outliers.

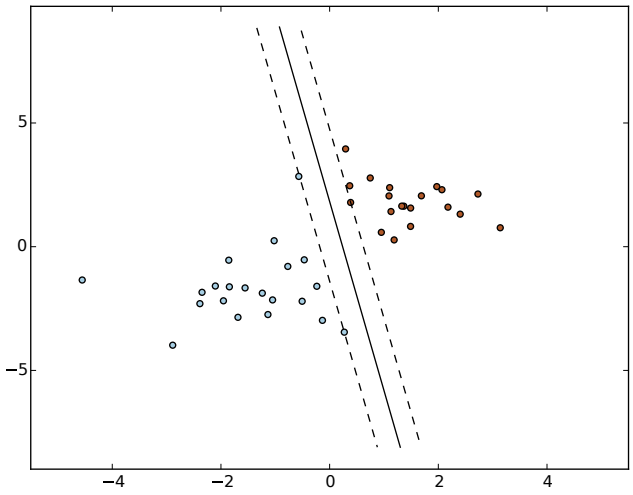


Figure 3.7: Outliers can alter the margin, resulting in a smaller margin.

In order to find the maximum margin hyperplane, we can compute its weights w by:

$$w = \sum_i a_i y_i x_i \tag{3.13}$$

In linear SVM, where $K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$, the hyperplane's i-th component correlates with the i-th feature relevance, since in the linear combination of Equation 3.1, a $\mathbf{x}$'s feature influence depends on the absolute value of w components combination [CRC+11]. Since it is possible to

analyze the relevance of a feature in a classification task, it becomes a useful technique to extract biomarkers from a set of features.

3.4 Neural Models

Studies and applications of artificial neural networks have been growing in the recent years [Ben13]. Artificial neural networks (or only neural networks) are mathematical models that are used to fit a specific function [Hay98, Chapter 1]. Neural networks are inspired by organic neurons in many ways, from its basic structure, i.e. the neuron, to its architecture and learning techniques. From a machine learning angle, a task to map a set of examples X with m features to their single valued outputs Y can be started with a single building block of neural networks: a neuron. The neuron functioning is defined by the following formula:

$$\begin{aligned} v(\mathbf{x}) &= \sum_{k=1}^{m} w_k x_k + b \\ \hat{y} &= o(\mathbf{x}) = \varphi(v(\mathbf{x})) \end{aligned} \tag{3.14}$$

where m is the number of features from $\mathbf{x}$ and $\mathbf{w}$ is the set of network weights. There is one weight for each feature from the input. A second parameter is introduced as b: the bias. As Equation 3.14 shows, the bias acts globally on the output, and it is not influenced by the given input. Thus, $\mathbf{w}$ and b are the model parameters. The weighted sum of the input and bias is defined as the activation potential of the neuron [Hay98, Chapter 1]. The function φ, called activation function, is a pre-defined function that affects the network behavior. In our example, $\mathbf{x}^{(i)}$ is the i-th example of X and, by adjusting $\mathbf{w}$ and b, we pursuit (in some cases) that $\hat{y}$ be equal to $y_{(i)}$, the i-th expected output. By adjusting its parameters $\mathbf{w}$ and b, it attempts to minimize the error between the actual output y of example $\mathbf{x}$ and the neural network output $\hat{y}(x)$. The neural network with a single neuron is represented as illustrated at Figure 3.8.

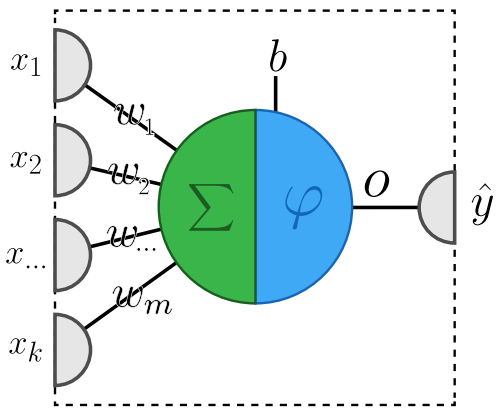


Figure 3.8: A neural network with a single neuron. In this configuration, the neuron's output is the neural network's overall output.

If $y_{(i)}$ is a continuous value and we expect to model a regressor (for a regression task), φ can be a linear function: $\varphi(v) = v$. If our task is to classify the given examples between two classes, i.e. we expect to model a classifier, a non-linear function can be used to squash the linear combination v in a specific range. The example will be classified as Class 1 when the activation gives a value close to the lower bound of function's range or as Class 2 when the value is close to the upper bound. Commom activation functions are illustrated by Figure 3.9.

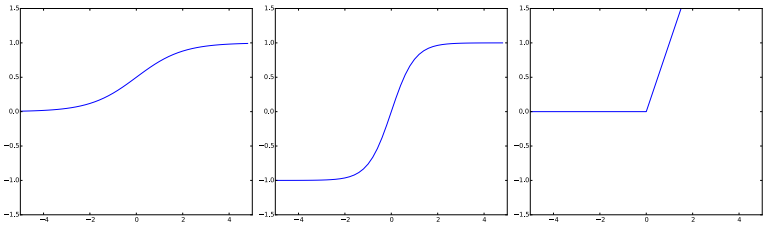


Figure 3.9: Commom activation functions: sigmoid, hyperbolic tangent and the rectifier

For a model with a single neuron, regardless of φ, the output is single-valued. In multi-neuron neural networks, neurons output become the inputs to other neurons and the neural network might have multiple values as output, depending on its architecture. There are several architectures of neural networks, and they are named according to their characteristics and functionalities. A regular architecture is the feedforward multi-layered neural network, in which neurons are disposed of side-by-side in layers, and the layer outputs are inputs to the next layer, as illustrated by Figure 3.10. The inner layers are denominated as hidden layers since there is no direct interaction with these layers, but just through input and output layers. In fully-connected architectures, neurons from a layer l are connected with each neuron from the successor layer $l + 1$. The connections between neurons represent the previously seen weights and regulate the influence of a neuron to their successors. The structure of connections can be manually changed, but it is difficult to measure how these changes impact the network performance.

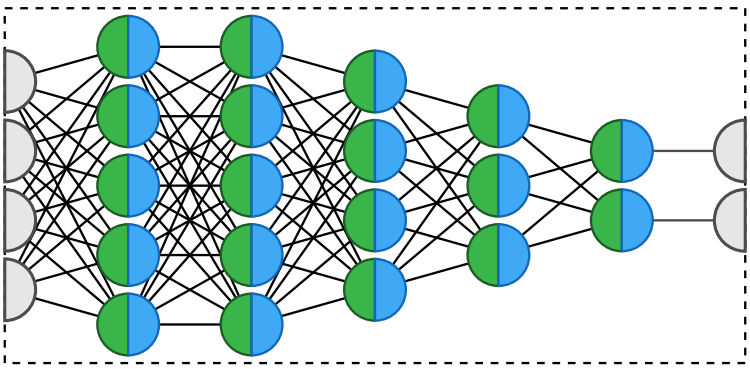


Figure 3.10: A neural network organized by layers. In this example, the network output are ruled by two neurons.

A specific activation function is preferred for the output of a neural network when the machine learning task is for classification between a set of classes C: the softmax function, shown in Formula 3.15. It assumes a different behavior, since it is not applied to a single neuron, but rather

is applied to the whole output layer. For each class c, the softmax function yields a probability of x being classified as c. Since the denominator adds up the exponential of each class, it guarantees that the probabilities sum up to 1.

$$p(y = c|\mathbf{x}) = \frac{exp(\mathbf{x}^T\mathbf{w}_c)_c}{\sum_{k=1}^{C} exp(\mathbf{x}^T\mathbf{w}_k)_k} \tag{3.15}$$

3.4.1 Neural Networks Efficiency

The capacity of a single neuron, i.e. the complexity of computation that it can perform, can be analyzed by the output it produces given an input. As in Figure 3.11, for a given neuron with two inputs and a sigmoidal output, it produces two flat planes separated by a ridge. Examples that fit in the upper plane are classified as one class, and examples that fit in the lower plane are classified as another class. The plane's format indicates that a single neuron can produce a linear decision boundary. In other words, a neuron capacity is to act as a binary classifier, and it can be used in situations where the given problem is separable by a hyperplane (consequently, for two dimensions, separable by a line).

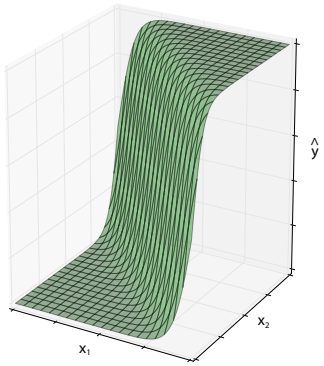


Figure 3.11: Plot of a single neuron output $\hat{y}$ given two features inputs: x_1 and x_2.

The following example shows how an artificial neuron can be modeled to simulate logic gates. The neuron has two input features (x_1 and x_2), and tries to simulate the logic gates: $AND(x_1, x_2)$, $OR(x_1, x_2)$ and $XOR(x_1, x_2)$ (exclusive OR). These logic gates are implementations of their respective logical operators and work in a binary space of values (0 and 1). The AND gate outputs 1 if both inputs are 1. Otherwise, it will output 0. The OR gate outputs 1 if at least one input is 1. If both inputs are 0, it outputs 0. The XOR gate outputs 1 if one, and only one, of the inputs to the gate, is 1. Their plots are illustrated in the Figure 3.12.

Given proper parameters (weights and bias), a single neuron can simulate the gates AND and OR, but it can not simulate an XOR gate. This occurs due the XOR behavior, in which its

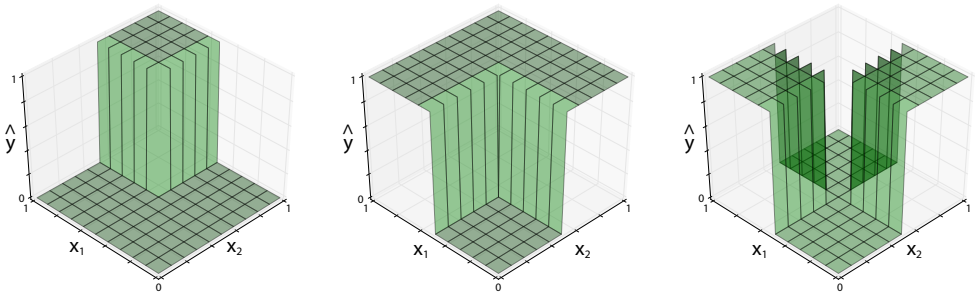


Figure 3.12: Plots of how AND, OR and XOR gates ideally behave.

output is not linearly separable. Intuitively, due to the simplicity of an XOR gate, this limitation suggests that a single neuron may not be sufficient for many problems of binary classification.

However, if the input is mapped to an alternate representation, and then used in XOR gate, it becomes linearly separable and is possible to simulate the gate with a single neuron. This is possible by projecting the inputs into $AND(NOT(x_1), x_2)$ and $AND(x_1, NOT(x_2))$ The different representation is easily seen for XOR gate, since it is a simple function. For complex functions, with lots of features, these relations are difficult to observe. So, for these cases, it would be useful if there is a way to automatically map inputs into better representations.

Since XOR gate is reproducible by mapping the input into $AND(NOT(x_1), x_2)$ and $AND(x_1, NOT(x_2))$, and single neurons can simulate the AND gate, it suggests that by connecting neurons into a network, this neural network can simulate more complex functions. As illustrated by Figure 3.13, a neural network with three neurons can simulate the XOR gate. So, based on multi-layer neural networks, it is possible to model XOR gate by adding the output from neurons in the same layer.

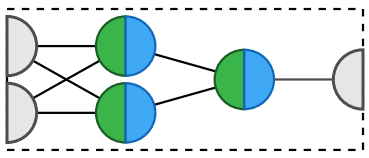


Figure 3.13: Neural network architecture that can simulate an XOR gate.

Regardless of toy problems with boolean operators, extensively analyzed by neural network literature (such as in [Ant01] and [RN10]), several mathematical analyses of the potential of neural networks brings about an interesting property of them: the universal approximation. The universal approximation theorem [Cyb89] states that, given a neuron network with a single hidden layer, its neurons can approximate continuous functions: each neuron models subsets of the function. This configuration is only possible when using an activation function that is non-constant, bounded and monotonically-increasing continuous, such as sigmoidal functions. Thus, for example, the function plotted in the Figure 3.14 can be modeled by a set of neurons.

Each hidden neuron models a piece (or a subset) of the function. The resulting sum of neurons' outputs by the output neuron gives an approximation of the expected function. Thus,

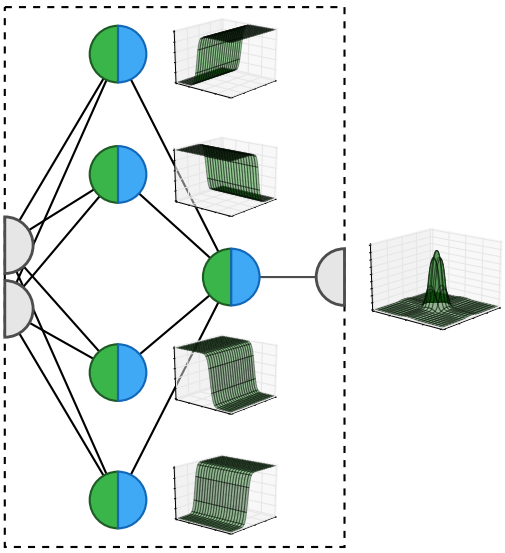


Figure 3.14: Capability of a neural network to approximate a mathematical function.

since simple neural networks can represent a wide variety of functions, it is expected that complex networks can represent highly varying data and complex functions. The ability to represent complex functions is regulated by the parameters of the network (weights and biases) and their relations within the model. In order to learn functions, optimization techniques are used to adjust the neural network parameters.

3.4.2 Learning as an Optimization Problem

If the learning process of a machine learning task is considered as an mathematical optimization problem, it is possible to apply methods of the Optimization field to look for a valid set of parameters for the machine learning model i.e. that have a better performance for the task.

An important algorithm in the optimization field is the Gradient Descent (GD) method [MOM12]. This method attempts to adjust the parameters of a model to minimize the model's error E, based on a predefined error function (also called loss function and cost function). In order to adjust these parameters, it computes the gradient vector that points to the error function's greatest rate of increase. Since GD aims to minimize the error, the adjustment is pointed to the negative of the gradient. The gradient is composed of partial derivatives of the error function with respect to each parameter, that are used to update parameter's value by:

$$\begin{aligned} \Delta w_k &= -\alpha \frac{\partial E}{\partial w_k} \\ w_k &= w_k + \Delta w_k \end{aligned} \tag{3.16}$$

where α is a chosen learning rate. Hyperparameters, such as the learning rate, are the parameters that will tell how an optimization algorithm will behave. The GD method can be used in several parametrized models, such as Support Vector Machines, k-Means and linear regression [MOM12].

For neural networks optimization, Backpropagation is the most common algorithm [MOM12], especially because of its simplicity and computational efficiency. This algorithm, along with gradient descent, is designed to propagate the error of a neural network backward, to optimize its parameters (weights and biases). For a given expected output y, and an error function E, the algorithm executes the following steps:

- Propagate the input through the network to generate the output $\hat{y}$.
- Propagate the error between y and $\hat{y}$ ($E(y, \hat{y})$) backwards to update weights.

where in the second step, the algorithm computes the partial derivatives of the error w.r.t network's parameters and adjust these parameters, as described in Equation 3.16.

To compute the gradient w.r.t a weight, the error must be backpropagated proportionally to the weight's influence for this error. Since the output is an entangled composition of weights and activation functions, the chain rule is used to compute the derivative of the error w.r.t. each weight. The chain rule states that, for a formula $f(g(x))$:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} \tag{3.17}$$

if f is differentiable at $g(x)$ and g is differentiable at x.

Consider a layered neural network with two inputs (x_1 and x_2), one hidden layer (with neurons 1 and 2) and one output neuron (3), as illustrated by Figure 3.15.

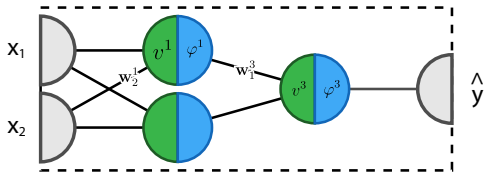


Figure 3.15: A neural network.

The network output is the result of the following formulae:

$$\begin{aligned} o^1(\mathbf{x}) &= h^1(s_1^1 x_1 + \mathbf{w}_2^1 x_2 + b^1) \\ o^2(\mathbf{x}) &= h^2(\mathbf{w}_1^2 x_1 + \mathbf{w}_2^2 x_2 + b^2) \\ \hat{y}(\mathbf{x}) &= h^3(\mathbf{w}_1^3 o^1(\mathbf{x}) + \mathbf{w}_2^3 o^2(\mathbf{x}) + b^3) \end{aligned} \tag{3.18}$$

where o^n, h^n, s^n, $\mathbf{w}^n$ and b^n represent the output, activation function, linear combination, weights and bias of neuron n, respectively. For a better comprehension, Figure 3.15 demonstrates the ordering of each variable across the neural network. In order to compute the derivative of error E w.r.t $\mathbf{w}_2^1$, the chain rule of this derivative is:

$$\frac{\partial E}{\partial \mathbf{w}_2^1} = \frac{\partial E}{\partial \varphi^3}\frac{\partial \varphi^3}{\partial v^3}\frac{\partial v^3}{\partial \mathbf{w}_1^3}\frac{\partial \mathbf{w}_1^3}{\partial \varphi^1}\frac{\partial \varphi^1}{\partial v^1}\frac{\partial v^1}{\partial \mathbf{w}_2^1} \tag{3.19}$$

This chaining is illustrated by Figure 3.16. Since the neural networks is a combination of functions, derivatives depend on the sequence of functions.

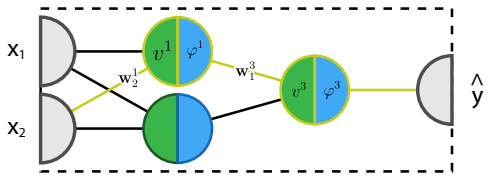


Figure 3.16: Chain rule in a neural network.

Each iteration of backpropagation steps is called a training epoch. Optimization algorithms that take an excessive number of training epochs are usually defined as slow learners. Commonly, the proper tuning of hyper-parameters of an optimization algorithm leads to faster learning. For example, the learning rate α dictates how much of the gradient will affect the weights. Large learning rates can bring an unstable optimization when it comes close to a local (or global) minimum. However, small learning rates limit the learning process of a neural network by decreasing the gradients and are more sensitive to local minimums.

For each epoch, gradient descent iterates over all the training set to do a single parameter update. If the number of examples in the training dataset is large, then the gradient descent may take a considerable amount of time to compute the gradients. This version of gradient descent is usually called batch gradient descent (BGD), given the methods to iterate the dataset. An alternative to batch gradient descent is the stochastic gradient descent (SGD) [MOM12]. SGD, that is also called on-line gradient descent, computes the gradient for each training example. It is defined as stochastic because the computed gradient is an approximation of the gradient from the whole training set. SGD computes the gradient for each example from the training set, and updates the network parameters, as defined in Equation 3.16. Since it is an approximation, SGD may slowly converge to a minimum, or may not converge. Usually, it presents a zigzag pattern, caused by fluctuations in the gradient. A good trade off between BGD and SGD is the mini-batch gradient descent (MBGD) [MOM12]. By splitting the training set into mini-batches (e.g. sets of 10 examples), the approximation of the BGD's gradient can be less noisy, and the convergence becomes faster.

To minimize the zigzag pattern from SGD, and accelerate the learning process, the momentum method is used [SMDH13]. It works in the parameters update, by applying Equation 3.20. In the first epoch, it does not have any effect, since velocity r has no value (i.e. zero). For the subsequent epochs, the momentum reduces the velocity of gradient if it goes in an opposite direction of the previous gradient, regulated by $\mu \in [0, 1]$.

$$\begin{aligned} r_{t+1} &= \mu r_t - \alpha \frac{\partial E}{\partial \mathbf{w}} \\ \mathbf{w}_{(t+1)} &= \mathbf{w}_{(t)} + r_{t+1} \end{aligned} \tag{3.20}$$

By using momentum method, the gradient becomes more stable, and convergence is faster. Its behavior is illustrated by Figure 3.17. The weight updates use the previous momentum to focus the gradient towards the minimum, rather than going to surrounding solutions.

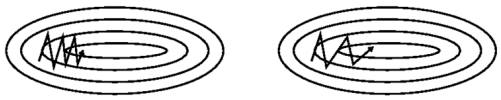


Figure 3.17: Behavior of gradient descent with and without momentum method.

Backpropagation has some limitations, such as the vanishing (or exploding) gradient problem [PMB13]. This problem is described as a lack of proper propagation of training errors to first layers, which do not modify first layers of networks, so they do not contain useful information. This problem occurs for deep neural networks that use sigmoidal activation function, with more than one hidden layer, in which the network layers exponentially decay to zero or become very large during back-propagation. In their boundaries, sigmoidal function leads to infinity (or negative infinity). When the weights assume values that, multiplied by the input, reach these boundaries, the gradient becomes unstable. The problem is more severe for recurrent neural networks since the "unfolded in time" version of these networks are very deep architecture. This limitation was the inspiration to researchers for proposing new methods aiming to overcome it [HS06].

3.4.3 Regularization

Since the learning problem is treated as an optimization problem, the applied optimization algorithms try to adjust the model parameters in order to minimize the error of training data. Overfitting can occur since models might contain a large number of parameters. If overfitting is not controlled, these parameters can be adjusted to precisely model training data with its particularities e.g. noise. This may not be the desired behavior since the model might not perform well on new data. To avoid this scenario, regularization techniques prune model's complexity and mitigate overfitting. Thus, regularization is any modification we make to the learning algorithm that is intended to reduce its generalization error but not its training error.

A use for the validation set, from cross-validation technique (detailed at Section 3.1), is to stop early the learning process [MOM12]. This regularization method is data-oriented, since it only works by regularizing the model performance when subjected to datasets, and does not directly change neural network parameters. Iterative learning algorithms, such as gradient descent, make the model better fit the training dataset at each epoch. At a certain epoch, the model can start to get worse generalization error and overfit to the training set. In order to avoid this scenario, the learning algorithm uses a validation set to stop the process when the performance measure for validation set becomes worse. However, if the criterion to stop the training is the increase of generalization error based on a validation set, a peak in the generalization error will stop the training, and a better model parametrization may have been lost. For this cases, usually the learning algorithm is not stopped

when the criterion match, but the model's parameters are stored when it reached the smaller value for generalization error, and the learning process continues. If the model does not perform well in the subsequent epochs, the stored parametrization is used as the final model.

A regularization technique that works based on the neural network parameters is the Dropout [Hin14]. During training, for each mini-batch, the technique sample a thinned version of the neural network, as illustrated by Figure 3.18. However, rather than preventing features dependency, it works over units dependency (called co-adaptation, based on its biological inspiration). Units are trained to behave in the presence of other units at the same network layer. By using this technique, the co-adaptation is prevented by making the presence of other units unreliable. Formally, units from a layer are sampled during training from a Bernoulli distribution with probability p. So, the propagation of a unit's output is silenced to the next layer in the case of a failure probability (i.e. $Bernoulli(p) = 0$). When dropout is used in the fine-tuning process, weights information are trained to behave with co-adaptive units, and dropout might wipe out this information. To guarantee that the weights' information are not lost, the weights from pre-training must be scaled up by a factor of $1/p$.

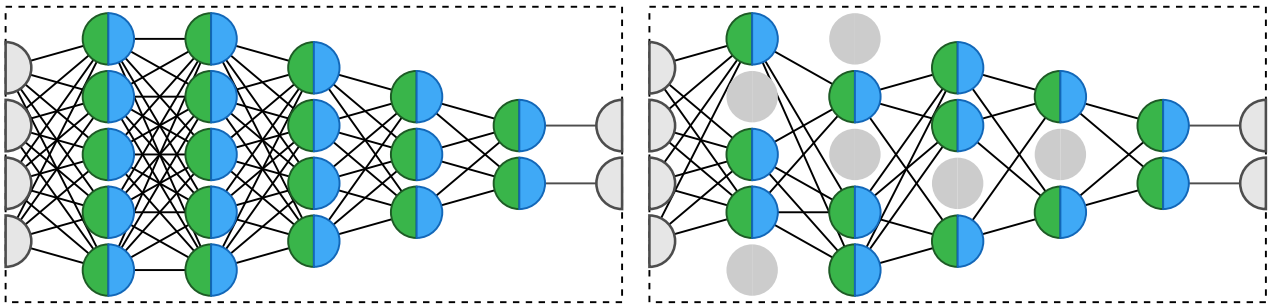


Figure 3.18: Dropout technique.

3.5 Deep Learning

The performance of machine learning algorithms depends on features extracted from data i.e. the way in which data is represented to use in the learning processes [Ben13]. Therefore, great effort (including human labor) is required to analyze data sets, in its preprocessing and transformation, seeking better representations to be used in the algorithm. Deep learning, also known as Deep Structural Learning or Hierarchical Learning, is one such effort. It consists of a set of techniques and architectures that attempt to find models with higher levels of feature abstractions, which hopefully make it easier to separate from each other the various explanatory factors extent in the data. Learning features at several abstraction levels allows a system to autonomously learn complex functions, without depending completely on human-crafted features.

Bengio [Ben09] suggests that this feature extraction is the current weakness of machine learning algorithms, showing the inability to extract and organize information that helps to learn a

machine learning task. The ability to learn useful features, called representation learning, becomes more important as the volume of data and the need to learn from data continues to grow. A known example of a linear feature extraction algorithm is the Principal Components Analysis (PCA). It looks for global linear directions of maximum variance and transform them into orthogonal components. This corresponds to a projection in a coordinate system spanned by PCA eigenvectors. The variation in the directions of the principal components are precisely captured by PCA, but variation in the orthogonal directions are lost. Pruning low-variance directions might not achieve a good solution since it is not always clear what is signal and what is noise in data. Otherwise, non-linear feature extraction algorithms assume that the relation between features is not linear, and usually achieve better results than linear processes.

In 2006, deep learning field started to yield results. Researchers have attempted to develop successful deep multilayer neural networks for decades, but they only reached networks with two hidden layers, obtaining poorer results with greater networks due to the vanishing gradient problem. After, Deep Belief Networks [HS06] brings about the notion of stacking learning units with an external learning algorithm that trains one layer at a time.

Following the same idea of learning units stacks, autoencoders were arranged in a network to work with data representations at different levels of abstraction. An autoencoder is a neural model that has a particular task: reconstruct its input at the output. It is considered an unsupervised learning algorithm since it tries to find hidden structures in data by trying to reconstruct it and does not require data to be labeled.

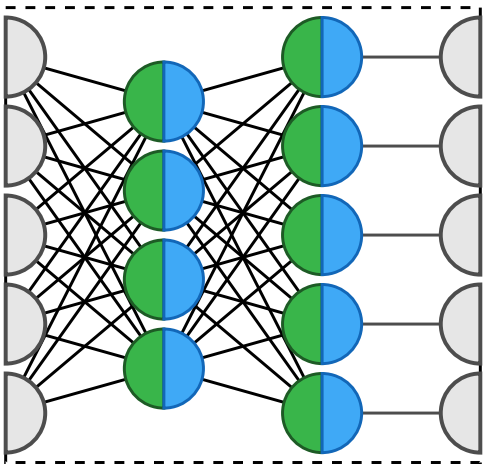


Figure 3.19: An autoencoder architecture.

For a given input x with m dimensions, it first attempts to encode to a n-dimension version c of x, and second attempts to decode c to x. By this means, the whole function is expected to produce x for a given x. In other words, the whole autoencoder aims to model an approximation to the identity function. The autoencoder is a neural network with one non-linear hidden layer with n units, and the input and output layers have the same number of units i.e. m. However, the identity function seems like a trivial function to model. But, by adding a bottleneck in the network architecture and constraints during training, the network can yield hidden structures from data. In

order to create a bottleneck, the number of units in the hidden layer n must be less than m. The first part of the network is defined as the encoder, since by inputting data in this part, the encoder will compress the input into a *code* c. Using the same logic, the second part of the network is defined as the decoder since it will transform the *code* c into the original input.

By compressing an m-dimensional data to a n-dimensional version of data, where $n < m$, the representation is characterized as undercomplete. There is a specific network configuration in which the hidden layer contains more units than the visible layers (i.e. $n > m$). This configuration is said to produce overcomplete data representations. Since this structure itself does not guarantee that it will extract meaningful structures, we only deal with undercomplete autoencoders.

By training stacked autoencoder, and after using its yielded *code* as input to the subsequent autoencoder, it is possible to achieve lower-dimensional hidden structures of data derived from several non-linear relationships of the original m-dimensional data. Each inner autoencoder compresses the code from a previous autoencoder to a lower dimension, being trained on layer per time, in a schema called greedy layer-wise training. This schema was proposed by Bengio et al. [BLPL07], that avoid the training of a deep neural network by breaking the network into shallow autoencoders (i.e. with a single hidden layer). At this point, by using autoencoders, we are interested in a paradigm called transfer learning [Ben11, Ben13]. The encoders from stacked autoencoders can be transposed to another neural network that uses the supervised learning schema. The encoders are trained to yield a lower-dimensional representation of data without much loss. The usage of encoders in a supervised task gives a proper initialization of the whole neural network parameters. However, since the previous task is not the same of the new neural network's task, the transposed encoders must pass through a phase called fine-tuning, in which the whole network is trained to this new supervised task. Thereby, encoders' parameters become more discriminant e.g. for a classification task. As pointed by Vincent et al. [VLL$^+$10], this supervised fine-tuning process can perform substantially better than without fine-tuning.

To avoid overfitting in autoencoders, Vincent at al. [VLBM08] proposed the denoising autoencoders. Denoising autoencoder (DAE) [VLBM08] is an extension of autoencoders that aims an specific characteristic: robustness under the partial corruption of the input. DAEs are expected to output the same representation of an input even if it is corrupted. The DAE hypothesis is based on that autoencoders do not deal with the presence of redundant features, so the encoding process depends on each redundant features. By training autoencoders with partially corrupted data, the encoding process becomes less dependant of each feature and less susceptible to data variability. So, for an input x, the DAE corrupts x by randomly assigning zeros to some of the features with a predefined probability p. The new corrupted input is defined as $\widetilde{x}$. However, the error function of the learning process will compare the output $\hat{x}$ to the original version x, and not $\widetilde{x}$. By doing this, the DAE is forced to use other features to create a better reconstruction of x, since by corrupting some features, they become unreliable and can not be used solely in the reconstruction. Vincent et al. [VLBM08] gives a good intuition of how denoising autoencoders work. Figure 3.20 illustrates the intuition.

For a given set of uncorrupted examples (thin black markers) and a learned manifold near which the data concentrate, it is likely that corrupted examples will be outside and farther from the manifold than the uncorrupted ones. The denoising autoencoder can thus be seen as a way to define and learn a manifold that comprise corrupted examples by projecting them over (or near to) the manifold. By doing this, DAE creates a representation map that captures the main factors of variation in the data.

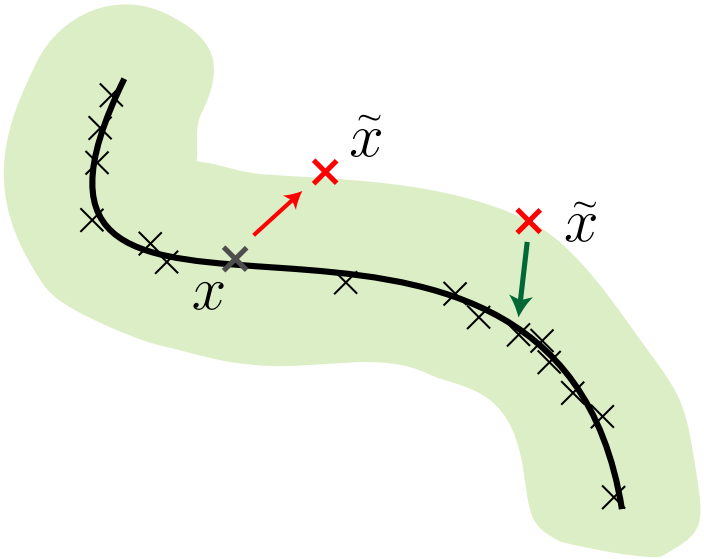


Figure 3.20: Denoising autoencoder intuition of how corruption leads to generalization. By learning a manifold that comprises the corruption process, DAE captures the main factors of variation in the data.

4. RELATED WORK

Recently, the combination of machine-learning methods with noninvasive brain imaging data has shown promising results for neurodevelopmental disorders. Most studies approach disorders such as schizophrenia, mood disorder, bipolar disorder, major depressive disorder, anxiety disorders, obsessive compulsive disorder, social anxiety disorder, post-traumatic stress disorder, attention deficit hyperactivity disorder, autism spectrum disorder, among others. However, most of these studies are based on a limited sample, with a few dozen subjects. To produce more reliable results, machine learning algorithms must be tested in a larger population samples. In general, most studies use supervised learning algorithms, such as linear discriminant classifier, support vector machines, and Gaussian process classifier, with leave-one-out cross validation.

ABIDE data have previously been used by Nielsen et al. [NZF+13] to classify autism vs. control subjects based on functional brain activation. The authors reproduced an approach reported in Anderson [ADF+11] with some modifications to include datasets from multiple sites. From the 964 subjects, they computed brain activation from non-overlapping, gray matter ROIs (SPM8 mask gray.nii) formed by seed voxels separated by at least 5mm. Voxels Euclidean-close to a specific ROI's seed voxel were included in the specific ROI. Based on data from the 7266 generated ROIs, they computed a connectivity matrix with 7266 x 7266, by using the Fisher-transformed r-to-z scores. Using a leave-one-out approach, they fit a general linear model for each condition (ASD and control) to associate the connectivity matrix with the subject covariates: age, gender and handedness. For each connection, its value was estimated for the left out subject based on the covariates and adjusted by using the difference between the site's mean value for that connection and the other site's mean values. This procedure aimed to mitigate between-site differences that could bias results, such as different scanners, and variations in scanning parameters and protocols. For the left-out subject, the actual value for each connection was then subtracted from the estimated values obtained from the ASD model and the non-ASD model. The average of this subtraction across all 7266 ROIs was computed, and the average values of ROIs were added up. Positive values were classified as ASD and negative values as controls. Nielsen et al. obtained accuracy as high as 60% for the classification ASD versus non-ASD. The authors attempted to accommodate multi-site data and sources of variance present in the ABIDE data.

Koyamada et al. [KSN+15] aim to extract a brain state from measurable brain activities by using Deep Neural Networks (DNN). They trained a fully-connected neural network with two hidden layers and a softmax output layer to classify task-based fMRI data from 499 subjects into seven categories related to the tasks: Emotion, Gambling, Language, Motor, Relational, Social and Working Memory. To regulate overfitting, they used a dropout technique. The authors achieved better accuracy using deep models (mean accuracy of 50.74%), compared to linear models (mean accuracy of 20.84%), such as linear regression and linear support vector machine.

Plis et al. [PHS+14] use deep learning to classify a combination of schizophrenia data from four sites and classify Huntington disease data from the PREDICT-HD project. In the first part of

the work, they attempted to classify 198 schizophrenic patients and 191 controls. They trained a Deep Belief Network with 3 depths (50 hidden units in the first layer, 50 in the second layer, and 100 in the top layer). The layers were pre-trained with unsupervised Restricted Boltzmann Machines and fine-tuned by adding a softmax layer to the output of each layer, and then training the output layer and its previous layers via backpropagation. The second part of the work uses data collected from healthy controls and Huntington disease of the PREDICT-HD. Huntington disease is a degenerative disease that causes the degeneration of neurons in certain areas of the brain. They propose to answer if it is possible an appropriate recognition of the disease with deep learning techniques, as well as differentiates patients into three degrees of severity (low, medium and high). For the classification task, they use 3500 scans, where 2641 were from patients and 859 from healthy controls. Trying to accomplish the task, they use a Deep Belief Network with 3 depths (50-50-100 hidden units in the first, second and the top layer respectively). Plis et al. have concluded that deep learning has a great potential in neuroimaging applications. Even shallow RBM experiments produce meaningful features and are already competitive with the traditional models. By using deep learning methods, the unsupervised pre-training mitigates the subjectivity involved in traditional feature selection.

Studies that aim to discriminate ASD and non-ASD from ABIDE dataset use a subset based on phenotypical information, such as subject's site and sex. Higher scores in site-specific studies argue that differences in image acquisition methods and research protocols affect pattern recognition methods since it limits the sensitivity for detecting abnormalities. Preprocessing methods may bias the pattern recognition and classification methods, for example for head motion correction. As shown by Power et al. [Pow12], head motion can rise systematic correlations between the subject and its condition. However, since ASD is prevalent in males, the majority of ASD studies focuses on male subjects. Despite differences in neuroimaging acquisition and preprocessing, the removal of female subjects from studies reduces the variability of brain traits. Due to the different brain development of female subjects compared to male subjects, that may lead to different affected areas. Focusing on male subjects facilitates the identification of patterns in ASD male subjects, but these patterns may not be useful for ASD diagnosis for female subjects. Since we aim a general diagnosis model for ASD, we include both groups of male and female subjects. We confirm that our neural model comprises male and female differences, as well differences in neuroimaging techniques.

Linear models, such as general linear models used by Nielsen et al. [NZF+13], do not have the capacity to capture non-linear relations between features. This is proven by Koyamada et al. [KSN+15], that has a significant increase in accuracy by using deep belief networks when comparing to linear methods. Koyamada et al. also show that regularization had an important relevance for model training, to achieve an eligible generalization error.

Plis et al. [PHS+14] demonstrate the effectiveness of deep learning methods. For Huntington disease, they show that deep models can comprise structural MRI features, and aid the discrimination between control, condition and condition severity. Plis's work, along with this work, can encourage the use of deep learning techniques by neuroscience community, to surpass disappointing results in other neurodevelopmental disorder studies [WBB+15].

5. EXPERIMENTS AND RESULTS

In this chapter, we describe the experiments we conducted on fMRI data from ABIDE consortium. By using deep learning techniques, we aim to accurately classify ASD subjects from non-ASD, and analyze which features are more relevant in the classification. In this way, more relevant features can be useful to identify potential biomarkers. First, we describe the data used to pursuit a mathematical model for ASD diagnosis. Secondly, we describe our deep learning task, along with the neural architecture and the model training. Third, we describe our interpretation method of feature relevance for classification and discuss the features identified by this method as relevant.

5.1 Data

Autism Brain Imaging Data Exchange (ABIDE)[1] is a consortium that aggregates 1035 resting-state fMRI datasets with their corresponding structural MRI and phenotypic information from 17 world-wide sites. These datasets are composed of 505 individuals with ASD and 530 age-matched neurotypicals, where 878 are male, and 157 are female. Some sites exclude female by design. Each site diagnoses ASD based on the combination of clinical judgment, by using Autism Diagnostic Observation Schedule (ADOS) and Autism Diagnostic Interview-Revised (ADI-R). The average of ADOS score per site is similar across sites, suggesting that ASD severity is consistent within subjects. Despite the lack of prior coordination, it is expected comparable subjects across sites due to the widespread use of standardized instruments of diagnosis.

All data were collected using 3 Tesla scanners. For each participant's 4D dataset, the first four time points were removed to minimize possible magnetic stabilization effects. Preprocessing phase of resting-state fMRI scans uses the Configurable Pipeline for the Analysis of Connectomes (C-PAC)[2], and includes slice-timing and motion correction, signal nuisance regression, and temporal filtering. Slice-timing corrects the differences between slices with respect to the true time of acquisition. Motion correction uses the mean image of all volumes to align functional volume by using Fourier interpolation. The mean image of all volumes after interpolation are used to compute six time series of head movements: three translational (in x, y and z-axis) and three rotational movements (pitch, yaw, and roll). The signal nuisance regression uses these time series to regress out (remove) these factor from the volumes. In order to remove potential physiological signal (such as heart beating), a component-based noise correction decomposes the signal from regions that are not supposed to have neuronal signal, such as the cerebral spinal fluid. The result of the decomposition is used in a general linear model to remove the physiological signal.

[1] http://fcon_1000.projects.nitrc.org/indi/abide/
[2] http://fcp-indi.github.io/

After skull removal, the registration phase uses the Montreal Neurological Institute (MNI) 152 template, a standard template applied to normalize the brain shape across subjects. The entire pipeline produces many derivatives, such as the functional parcellation using *Craddock-200* [CJ12]. This method divides the brain into 200 functional parcels, defined by a spatially constrained clustering algorithm, which gathers points by their similarity. Qualitatively, the clustering in 200 parcels is adequate to accommodate individual anatomic variability, when comparing it to common atlas derived from anatomical (i.e. not functional) parcellation methods. By grouping voxels with *Craddock-200* parcellation, we achieve a representative reduction of data dimensionality without losing features significance, since voxels within a group are supposed to behave similarly.

To compute the functional connectivity, we compute the Pearson correlation coefficient between each *Craddock-200* parcel. To use the values in the correlation matrix as features, we remove the upper triangle values, including the diagonal, since it represents an area correlating to itself. Later, we flatten the remaining triangle, to retrieve a features set. The number of features is defined by Equation 5.1:

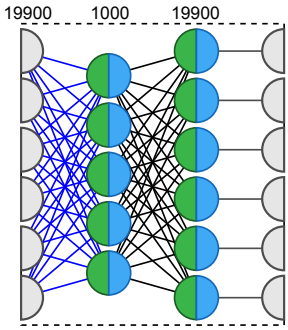
$$S = (N - 1)N/2 \tag{5.1}$$

where N is the number of voxels or regions to correlate. Since we use the *Craddock-200* parcellation, our N is 200, resulting in a feature set with 19900 features per subject.

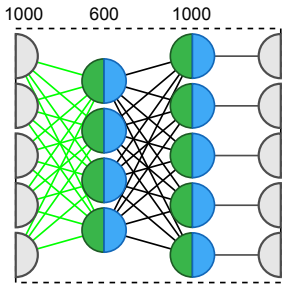
Table 5.1 compiles phenotypical data per site and class: the mean age of subjects and subject count.

Table 5.1: Phenotype summary. M: Male, F: Female.

	ASD		non-ASD	
Site	Mean Age	Count	Mean Age	Count
CALTECH	27.4421	19 (M 15, F 4)	27.4421	19 (M 15, F 4)
CMU	26.3571	14 (M 11, F 3)	26.3571	14 (M 11, F 3)
KKI	10.019	20 (M 16, F 4)	10.019	20 (M 16, F 4)
LEUVEN	17.7517	29 (M 26, F 3)	17.7517	29 (M 26, F 3)
MAX_MUN	26.0833	24 (M 21, F 3)	26.0833	24 (M 21, F 3)
NYU	14.7375	75 (M 65, F 10)	14.7375	75 (M 65, F 10)
OHSU	11.4308	12 (M 12, F 0)	11.4308	12 (M 12, F 0)
OLIN	16.5263	19 (M 16, F 3)	16.5263	19 (M 16, F 3)
PITT	19.0038	29 (M 25, F 4)	19.0038	29 (M 25, F 4)
SBL	35	15 (M 15, F 0)	35	15 (M 15, F 0)
SDSU	14.7193	14 (M 13, F 1)	14.7193	14 (M 13, F 1)
STANFORD	9.99826	19 (M 15, F 4)	9.99826	19 (M 15, F 4)
TRINITY	16.8127	22 (M 22, F 0)	16.8127	22 (M 22, F 0)
UCLA	13.0067	54 (M 48, F 6)	13.0067	54 (M 48, F 6)
UM	13.1697	66 (M 57, F 9)	13.1697	66 (M 57, F 9)
USM	23.4563	46 (M 46, F 0)	23.4563	46 (M 46, F 0)
YALE	12.7496	28 (M 20, F 8)	12.7496	28 (M 20, F 8)



(a) Autoencoder AE1: 19900-1000-19900



(b) Autoencoder AE2: 1000-600-1000

Figure 5.1: Two autoencoders structure. We reduce their number of units in order to ease the visualization of the structures.

5.2 Deep Learning task

Our deep learning architecture uses two stacked denoising autoencoders, which compress our 19900-dimensional data into 600 features, illustrated by Figure 5.1.

For the first autoencoder, called *AE1* for brevity, its task is to reconstruct the input: the subject's functional connectivity. The input layer has 19900 features, connected to 1000 units from the hidden layer. The binomial corruption module, used to corrupt data, is parametrized to corrupt (set to zero) a feature with 70% of chance.

The second autoencoder, called *AE2* for brevity, attempts to compress the output of the first autoencoder i.e. 1000 features into 600 features. The binomial corruption module of the second autoencoder is parameterized to corrupt a feature with 90% of chance.

Autoencoders unsupervised training is done one layer at a time, using the greedy layer-wise training. Both autoencoders were initialized following the heuristic rule of suggested by Glorot et al. [GB10]:

$$w \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \tag{5.2}$$

where $U\left[-a, a\right]$ is the uniform distribution in the interval $(-a, a)$ and n is the number of features. In order to achieve a better generalization rate for autoencoders, we choose the parametrization that during training minimize the mean squared reconstruction error for the valid set.

To utilize the knowledge extracted with the autoencoders, we apply the encoders weights to a multi-layer network (called DeepNN for brevity) with the configuration: 19900-1000-600-softmax. This process is demonstrated by Figure 5.2.

This resultant network is trained in a supervised way. Since the network already contains adjusted weights based on autoencoders, this training process is a fine-tuning, to adjust it to output the expected classes and minimize prediction error on this supervised task. The output layer contains a softmax output: each output represents the probability of an input be classified as classes ASD or non-ASD. The dropout technique constrains the DeepNN with 40% of omission chance of a unit for

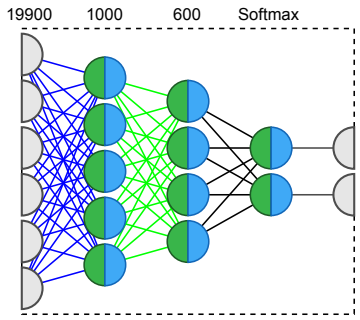


Figure 5.2: Transfer learning from autoencoders AE1 and AE2 to a neural network classifier.

the first layer, and 20% of chance for the second layer. In order to achieve a better generalization rate, we choose the parametrization that during training maximize the accuracy for the valid set. Table 5.2 summarizes the hyperparameters used for model training. We choose these hyperparameters based on multiple interactions of each part from our architecture: for autoencoder hyperparameters, we choose based on the best RMSE achieved for the validation set, and for DeepNN hyperparameters, we choose based on the best accuracy achieved for the validation set.

To compare the performance of this deep architecture, we compute the accuracy of Linear SVM and for a regular multi-layer neural network (called RegNN, for brevity) with the same training hyper-parameters. RegNN parametrization only differs from the initialization of weights. It follows the heuristic rule of [GB10], instead of the parameters configuration achieved by pre-trained autoencoders. All models are trained using the k-fold schema, with $k = 10$. Other schemas for further analysis were considered, such as the leave-one-site schema. In order to test if there is a bias or noise from a specific site, each training iteration of our neural network architecture should remove the site. The result of this kind of analysis could suppress data variations that are not linked to the disorder, but from sources such wrong image processing or different paradigm instructions. However, given the unbalanced number of subjects from each site, this type of analysis could become cumbersome, so we focus on the k-fold approach. The classification results are summarized in Table 5.3.

Sensitivity (correct detection rate within ASD subjects) is higher for DeepNN (73%), indicating that our model better identify ASD subjects when compared with the other methods. On the other hand, DeepNN's specificity (correct detection rate within non-ASD subjects) is lower than RegNN's (64% vs. 72%), thus indicating that the DeepNN has a worse performance to identifying non-ASD subjects. Nevertheless, misevaluated non-ASD subjects can be correctly identified during further interventions. The main focus for an ASD/non-ASD classifier must be to correctly identify ASD, so that to address these subjects to specialists for proper supervision and therapies.

To construct our denoising autoencoder and multi-layer neural network, we use Pylearn2 [GWFL$^+$13] [3], a deep learning framework that implements several neural network modules, from artificial neuron units to complete neural network architectures. Pylearn2 is developed over Theano [BBB$^+$10] [4]. Theano is a math expression compiler that creates a transparent interface to GPUs.

[3] https://github.com/lisa-lab/pylearn2/
[4] https://github.com/theano/theano/

Table 5.2: Model hyperparameters

Model	Hyperparameter	
AE1	Encoder Activation function	Hyperbolic Tangent
	Decoder Activation function	Linear
	Epochs	8000
	Learning Rate	0.0001
	Error function	Mean Squared Reconstruction Error
	Denoising Corruption	70%
	Batch size	100
AE2	Encoder Activation function	Hyperbolic Tangent
	Decoder Activation function	Linear
	Epochs	2000
	Learning Rate	0.0001
	Error function	Mean Squared Reconstruction Error
	Denoising Corruption	90%
	Batch size	10
DeepNN	Output	Softmax
	Epochs	100
	Learning Rate	.0005
	Error function	Cross-entropy
	Dropout Layer 1 Omission Prob	40%
	Dropout Layer 1 Scaling	1.66
	Dropout Layer 2 Omission Prob	20%
	Dropout Layer 2 Scaling	1.25
	Momentum: initial epoch	0.1
	Momentum: final epoch	0.9
	Momentum adjustment	Linear

Table 5.3: Summary of results for the proposed classification task.

Method	Accuracy	Sensitivity	Specificity	F-Score
DeepNN	0.69 std. 3.1	0.73	0.64	0.70
SVM	0.65 std. 2.7	0.68	0.62	0.67
RegNN	0.65 std. 3.8	0.65	0.72	0.68

This interface allows to compute mathematical models in GPU using NVIDIA CUDA (Compute Unified Device Architecture), a parallel computing architecture that enables dramatic increases in computing performance.

5.3 Biomarkers

In order to extract which features are more relevant to our classification, we analyze the neural network weights. For each neuron, weights have the role of factors that reflect neuron's input importance, in a weighted average fashion. Considering a single neuron using a sigmoidal activation function, the inputs are proportionally combined, and this final value is squashed into a specific

boundary. After its training, the neurons are expected to output a value within the activation function range. Since the weight initialization follows the heuristic rule proposed by Glorot et al. [GB10], the weights are not updated to values that tend to infinity boundaries of activation function during training.

In our analysis, we use bounded features: the correlation matrix assumes values between -1 and 1, a property also observed in the hyperbolic tangent function. Since features are on the same scale, it is possible to infer that weights' magnitude from input to the first hidden layer control how the input influence on the output of a given unit. In inner layer weights, the values that cross through the network are from activations of previous units i.e. results of the hyperbolic tangent function. Therefore, inner weights control input influence in the same way that weights of visible layers, since their magnitudes are on the same scale.

In our case, weights are in the same magnitude. Therefore, to analyse which features of our input yield better accuracy, the following formula can be used:

$$F = W^{(1)} \cdot W^{(2)} \cdot W^{(3)} \tag{5.3}$$

where $W^{(l)}$ is a matrix of weights between layer l and $l + 1$. This formula results in F, that is an N-vector (where N is the data dimensionality). Each value of F represents the resultant weight of the respective feature. To analyze F, we use its absolute value $|F|$, since we are interested in its magnitude.

The model average for neural networks is cumbersome since a different configuration of weights may appear, based on the stochastic the processes of training phase. Thus, to evaluate our methods, we use the model that yields better accuracy for a given test set. The model with better accuracy can give us a good approximation of which features are more relevant for the classification task. For this analysis, we retrieve the mean values of features from training set for each class: ASD and non-ASD. We use these mean values of features, that yields $P(c|x) = 1.0$ for their respective classes c i.e. the model reliably classifies the mean as being from one class or other.

After retrieving F coefficient for each feature, we sort features by features that have lower F magnitudes i.e. less relevant features. Iteratively, for each class, we remove the n less relevant features by setting their value to the mean value of the opposite class. This schema is illustrated by Figure 5.3.

By removing less relevant features, we expect that the reliability sustains as much as n increases i.e. that softmax outputs $p(y = \text{ASD}|\mathbf{x}) = 1.0$ when the mean of ASD subjects from training set is used, and $p(y = \text{non-ASD}|\mathbf{x}) = 1.0$ when the mean of non-ASD subjects from training set is used. It is possible to observe in Figure 5.4 that the 2700 most relevant features can sustain the classification ($p(y = ASD|\mathbf{x}) > 0.7$) since by removing the lesser relevant features does not interfere much with accuracy. Only by removing these 2700 features, that presents a high impact on the classification reliability.

This method limits our analysis from 19900 features to the 2700 most relevant features.

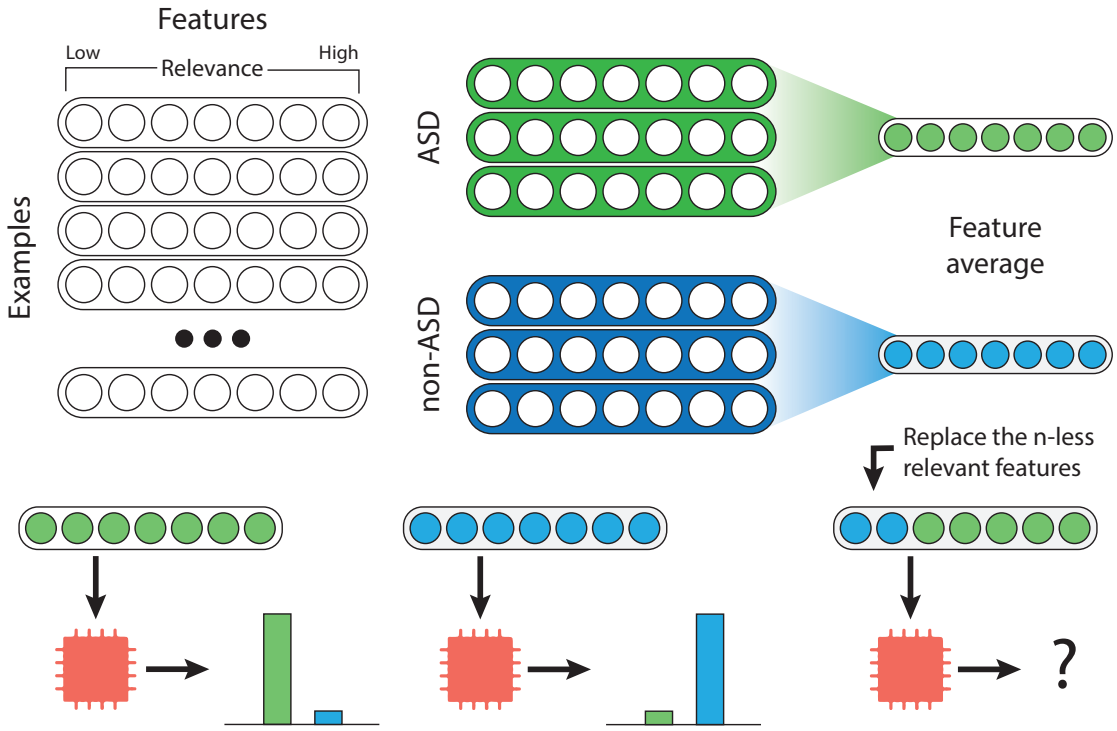


Figure 5.3: Schema to analyse the confidence of our classifier against the removal of less relevant features from a class.

To analyze these 2700 features, we group the parcels into regions from MNI Structural atlas, composed of: frontal lobe, temporal lobe, parietal lobe, occipital lobe, thalamus, caudate, cerebellum, insula, and putamen. Each region is illustrated by Figure 5.5.

We plot chord diagrams, to analyze the relevant features and their mean values for each class. In the chord diagrams, each chord represents a feature: the correlation between two parcels from different lobes. The chord colors represent if two parcels are under or over-connected. If they are under-connected i.e. they have negative correlation, the color is blue. Otherwise, if the parcels are over-connected i.e. they have positive correlation, the color is red. We multiply the mean values of features for each class by their respective value F, to highlight the most relevant features in the diagram.

The diagrams confirm the presence of under-connected regions in autistic brains. Most relevant under-connected parcels belong to the temporal lobe, as illustrated by the gray portion in Figure 5.6. However, like in other studies, the autistic brain is not only under-connected, but it also contains over-connected patterns. Specifically, the parietal lobe (brown) and the occipital lobe (purple) have the most relevant over-connected parcels, illustrated at Figure 5.8.

Our results demonstrate the potential of deep learning techniques. They significantly surpass the state of the art that uses the whole ABIDE dataset. The neural network structure comprises the brain variations, independently of age or sex. The findings of neural network analysis bring consistent patterns with ASD literature. Given a large number of relevant features, the analysis shows that there are diffuse systems across the brain network, rather than concentrated regions.

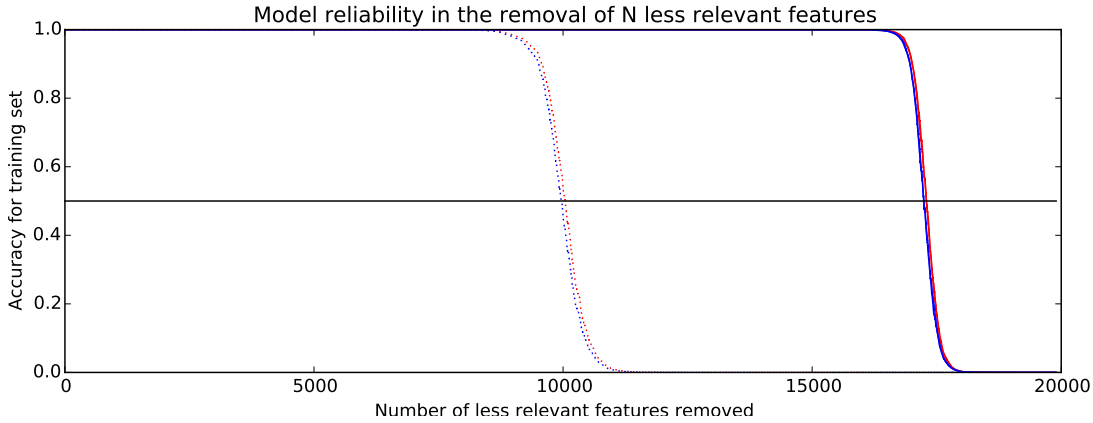


Figure 5.4: Analysis of feature relevance based on reliability for ASD (red) and non-ASD (blue). Irrelevant features do not interfere in model's reliability. Dotted lines represent the model reliability for removal of random features.

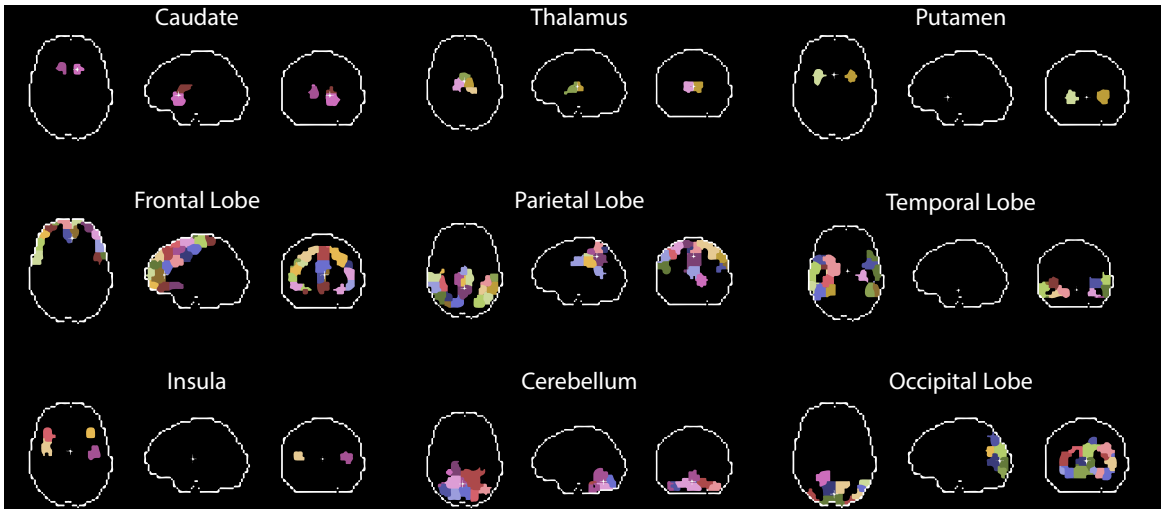


Figure 5.5: MNI Structural regions.

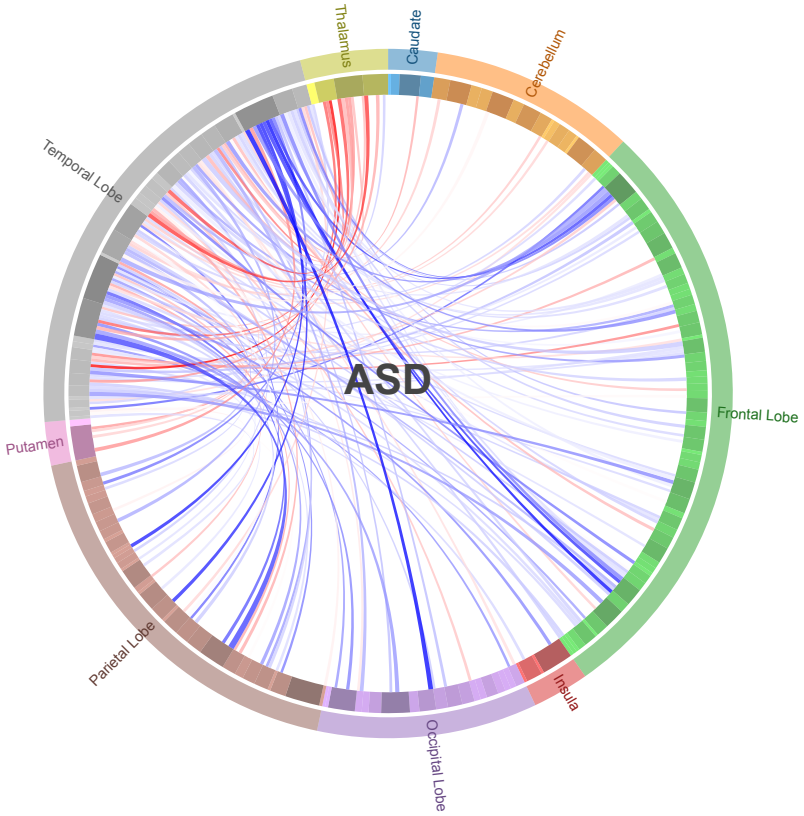


Figure 5.6: Most relevant under-connected parcels from temporal lobe for ASD.

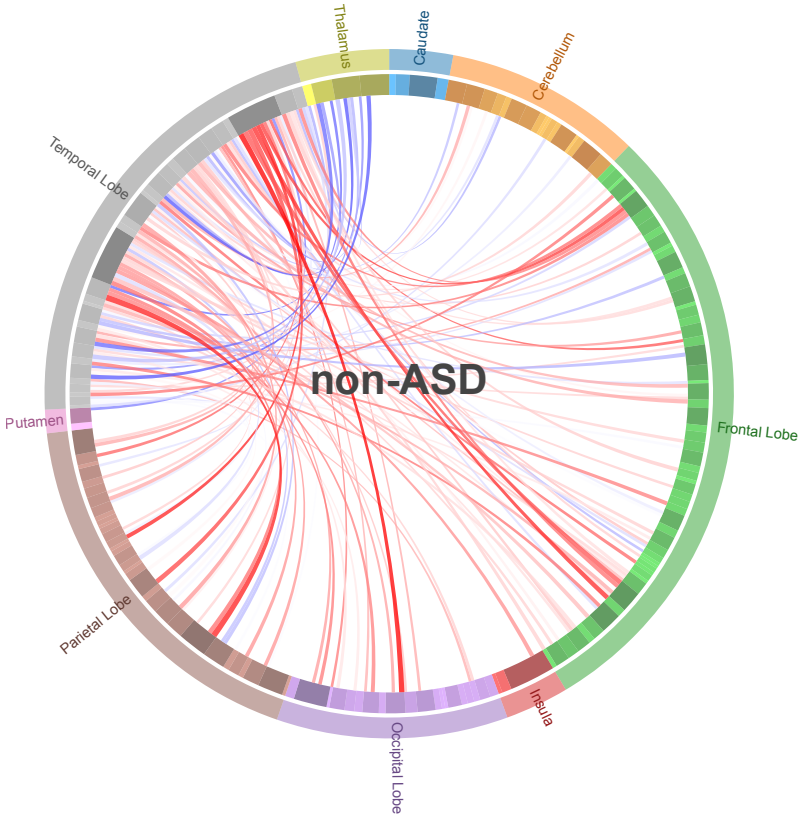


Figure 5.7: Most relevant under-connected parcels from temporal lobe for non-ASD.

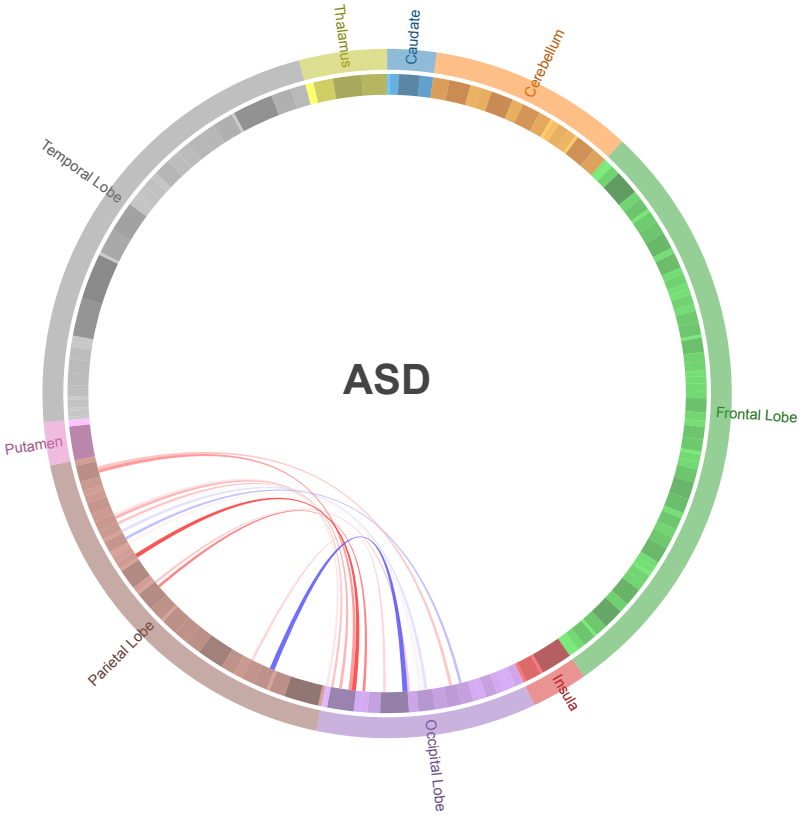


Figure 5.8: Most relevant over-connected parcels for ASD.

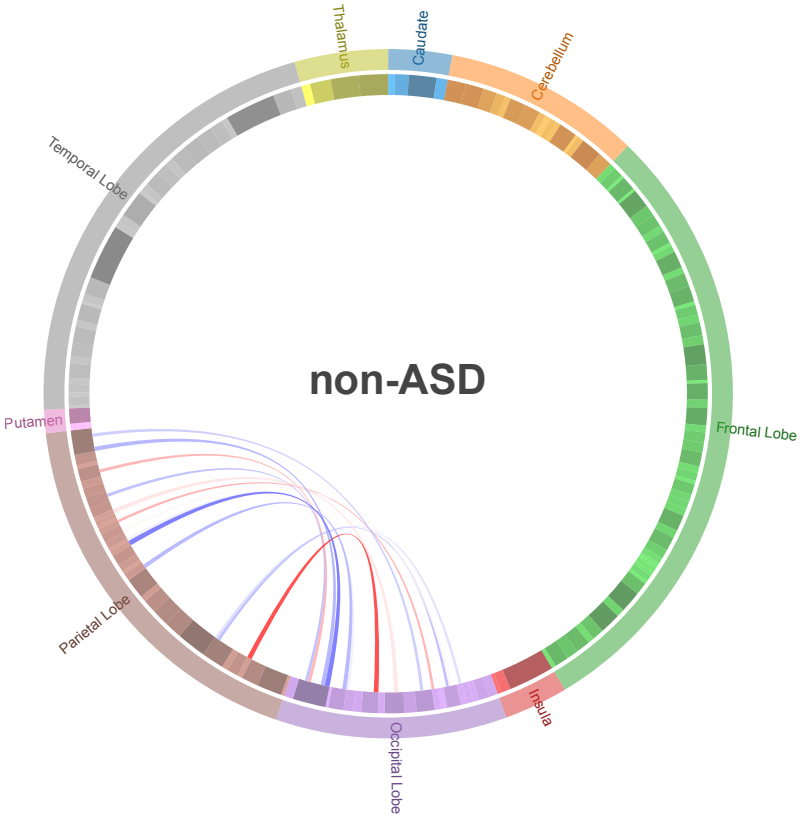


Figure 5.9: Most relevant over-connected parcels for non-ASD.

6. CONCLUSION

In this work, we contribute in finding patterns in resting-state fMRI to diagnose the cognitive states associated with autism spectrum disorder. By creating diagnose models that do not require phenotypical filtering, such by only using male samples, we surpass the state-of-the-art and account for brain state variations that are not included in other studies. To accomplish this, we have applied deep learning techniques to automatically extract the relevant features from fMRI data, so it does not require a laborious feature selection process, or the aid of an expert to point out regions of interest. In the experiments, we compared our technique with commonly used techniques in neuroscience.

Our results show that non-linear models can achieve better results and that a pre-training process gives an efficient starting point for neural network's optimization. Thus, we outperform the state-of-the-art by achieving an accuracy of 69%. By identifying which features are more relevant to the classification, we extracted possible biomarkers that can be used for precise ASD diagnosis.

As future work, we plan to use different neural network architectures, such as convolutional networks. By using convolution networks, we expect that the model handle possible spatial variation in neuroimages. We aim to use transfer learning techniques in different datasets, from neuroimaging to genomics, to aid the extraction of relevant features for a given discriminant tasks. We believe that the mixture of different imaging techniques may reveal new insights for neurodevelopmental diagnosis.

BIBLIOGRAPHY

[ADF+11] Anderson, J. S.; Druzgal, T. J.; Froehlich, A.; Dubray, M. B.; Lange, N.; Alexander, A. L.; Abildskov, T.; Nielsen, J. A.; Cariello, A. N.; Cooperrider, J. R.; Bigler, E. D.; Lainhart, J. E. "Decreased interhemispheric functional connectivity in autism", *Cerebral Cortex*, vol. 21, May 2011, pp. 1134–1146.

[Ant01] Anthony, M. "Discrete Mathematics of Neural Networks: Selected Topics". Society for Industrial and Applied Mathematics, 2001, 131p.

[APF+13] Atluri, G.; Padmanabhan, K.; Fang, G.; Steinbach, M.; Petrella, J. R.; Lim, K.; Macdonald, A.; Samatova, N. F.; Doraiswamy, P. M.; Kumar, V. "Complex biomarker discovery in neuroimaging data: Finding a needle in a haystack", *NeuroImage: Clinical*, vol. 3, Aug 2013, pp. 123–131.

[BBB+10] Bergstra, J.; Breuleux, O.; Bastien, F.; Lamblin, P.; Pascanu, R.; Desjardins, G.; Turian, J.; Warde-Farley, D.; Bengio, Y. "Theano: a CPU and GPU math expression compiler". In: Python for Scientific Computing Conference (SciPy), 2010.

[Ben09] Bengio, Y. "Learning deep architectures for AI", *Foundations and Trends in Machine Learning*, vol. 2, Jan 2009, pp. 1–127.

[Ben11] Bengio, Y. "Deep Learning of Representations for Unsupervised and Transfer Learning". In: Journal of Machine Learning Research, 2011.

[Ben13] Bengio, Y. "Deep learning of representations: Looking forward", *Computing Research Repository*, vol. 2, May 2013, pp. 1–127.

[Bis06] Bishop, C. M. "Pattern Recognition and Machine Learning". Springer, 2006, 738p.

[BLPL07] Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H. "Greedy Layer-Wise Training of Deep Networks", *Advances in Neural Information Processing Systems*, vol. 19, Sep 2007, pp. 153–160.

[Bux02] Buxton, R. "Introduction to Functional Magnetic Resonance Imaging: Principles and Techniques". Cambridge University Press, 2002, 470p.

[BYHH95] Biswal, B.; Yetkin, F. Z.; Haughton, V. M.; Hyde, J. S. "Functional connectivity in the motor cortex of resting human brain using echo-planar MRI", *Magnetic Resonance in Medicine*, vol. 34, Oct 1995, pp. 537–541.

[CHHM09] Craddock, R. C.; Holtzheimer, P. E.; Hu, X. P.; Mayberg, H. S. "Disease state prediction from resting state functional connectivity", *Magnetic Resonance in Medicine*, vol. 62, Dec 2009, pp. 1619–1628.

[CJ12] Craddock, R.; James, G. "A whole brain fMRI atlas generated via spatially constrained spectral clustering", *Human Brain Mapping*, vol. 33, Jul 2012, pp. 1914–1928.

[CKM13] Chen, C. P.; Keown, C. L.; Müller, R.-A. "Towards Understanding Autism Risk Factors: a Classification of Brain Images With Support Vector Machines", *International Journal of Semantic Computing*, vol. 07, Jun 2013, pp. 205–213.

[CRC+11] Cuingnet, R.; Rosso, C.; Chupin, M.; Lehéricy, S.; Dormont, D.; Benali, H.; Samson, Y.; Colliot, O. "Spatial regularization of SVM for the detection of diffusion alterations associated with stroke outcome", *Medical Image Analysis*, vol. 15, Oct 2011, pp. 729–737.

[Cyb89] Cybenko, G. "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control, Signals and Systems*, vol. 9, Dec 1989, pp. 303–314.

[dIIVMMJ+13] de la Iglesia-Vaya, M.; Molina-Mateo, J.; Jose, M.; S., A.; Marti-Bonmati, L. "Brain Connections - Resting State fMRI Functional Connectivity". In: *Novel Frontiers of Advanced Neuroimaging*, InTech, 2013.

[DYL+14] Di Martino, A.; Yan, C.-G.; Li, Q.; Denio, E.; Castellanos, F. X.; Alaerts, K.; Anderson, J. S.; Assaf, M.; Bookheimer, S. Y.; Dapretto, M.; Deen, B.; Delmonte, S.; Dinstein, I.; Ertl-Wagner, B.; Fair, D. a.; Gallagher, L.; Kennedy, D. P.; Keown, C. L.; Keysers, C.; Lainhart, J. E.; Lord, C.; Luna, B.; Menon, V.; Minshew, N. J.; Monk, C. S.; Mueller, S.; Müller, R.-A.; Nebel, M. B.; Nigg, J. T.; O'Hearn, K.; Pelphrey, K. a.; Peltier, S. J.; Rudie, J. D.; Sunaert, S.; Thioux, M.; Tyszka, J. M.; Uddin, L. Q.; Verhoeven, J. S.; Wenderoth, N.; Wiggins, J. L.; Mostofsky, S. H.; Milham, M. P. "The autism brain imaging data exchange: towards a large-scale evaluation of the intrinsic brain architecture in autism", *Molecular Psychiatry*, vol. 19, Jun 2014, pp. 659–667.

[ERRJ+10] Ecker, C.; Rocha-Rego, V.; Johnston, P.; Mourao-Miranda, J.; Marquand, A.; Daly, E. M.; Brammer, M. J.; Murphy, C.; Murphy, D. G. "Investigating the predictive value of whole-brain structural MR scans in autism: A pattern classification approach", *NeuroImage*, vol. 49, Jan 2010, pp. 44–56.

[GB10] Glorot, X.; Bengio, Y. "Understanding the difficulty of training deep feedforward neural networks". In: International Conference on Artificial Intelligence and Statistics, 2010.

[GR10] Goldstein, S.; Reynolds, C. R. "Handbook Of Neurodevelopmental And Genetic Disorders In Children". The Guilford Press, 2010, 588p.

[Gre08] Greicius, M. "Resting-state functional connectivity in neuropsychiatric disorders", *Current Opinion in Neurology*, vol. 24, Aug 2008, pp. 424–430.

[GWFL+13] Goodfellow, I. J.; Warde-Farley, D.; Lamblin, P.; Dumoulin, V.; Mirza, M.; Pascanu, R.; Bergstra, J.; Bastien, F.; Bengio, Y. "Pylearn2: a machine learning research library". arXiv:1308.4214, Aug 2013.

[Hay98] Haykin, S. "Neural Networks: A Comprehensive Foundation". Prentice Hall, 1998, 936p.

[Hin14] Hinton, G. "Dropout : A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, vol. 15, Jun 2014, pp. 1929–1958.

[HS06] Hinton, G. E.; Salakhutdinov, R. R. "Reducing the dimensionality of data with neural networks", *Science*, vol. 313, Jul 2006, pp. 504–507.

[HSM04] Huettel, S. A.; Song, A. W.; McCarthy, G. "Functional magnetic resonance imaging". Freeman, 2004, 537p.

[HTF01] Hastie, T.; Tibshirani, R.; Friedman, J. "The Elements of Statistical Learning". Springer, 2001, 745p.

[HW79] Hartigan, J. a.; Wong, M. a. "A K-Means Clustering Algorithm", *Journal of the Royal Statistical Society*, vol. 28, Jan 1979, pp. 100–108.

[KSN+15] Koyamada, S.; Shikauchi, Y.; Nakae, K.; Koyama, M.; Ishii, S. "Deep learning of fMRI big data: a novel approach to subject-transfer decoding". arXiv:1502.00093, Jan 2015.

[LMS09] Levy, S. E.; Mandell, D. S.; Schultz, R. T. "Autism", *The Lancet*, vol. 374, Nov 2009, pp. 1627–1638.

[LRG+89] Lord, C.; Rutter, M.; Goode, S.; Heemsbergen, J.; Jordan, H.; Mawhood, L.; Schopler, E. "Austism diagnostic observation schedule: A standardized observation of communicative and social behavior", *Journal of Autism and Developmental Disorders*, vol. 19, Jun 1989, pp. 185–212.

[LRL94] Lord, C.; Rutter, M.; Le Couteur, A. "Autism Diagnostic Interview-Revised: A revised version of a diagnostic interview for caregivers of individuals with possible pervasive developmental disorders", *Journal of Autism and Developmental Disorders*, vol. 24, Oct 1994, pp. 659–685.

[Men11] Menon, V. "Large-scale brain networks and psychopathology: A unifying triple network model", *Trends in Cognitive Sciences*, vol. 15, Sep 2011, pp. 483–506.

[Mil11] Miles, J. H. "Autism spectrum disorders—A genetics review", *Genetics in Medicine*, vol. 13, Apr 2011, pp. 278–294.

[Mit97] Mitchell, T. M. "Machine Learning". McGraw-Hill, 1997, 432p.

[MMR+01] Muller, K.-R.; Mika, S.; Ratsch, G.; Tsuda, K.; Scholkopf, B. "An introduction to kernel-based learning algorithms", *IEEE Transactions on Neural Networks*, vol. 12, Mar 2001, pp. 181–201.

[MOM12] Montavon, G.; Orr, G.; Müller, K.-R. "Neural Networks: Tricks of the Trade". Springer, 2012, 769p.

[MSK+11] Müller, R. A.; Shih, P.; Keehn, B.; Deyoe, J. R.; Leyden, K. M.; Shukla, D. K. "Underconnected, but how? A survey of functional connectivity MRI studies in autism spectrum disorders", *Cerebral Cortex*, vol. 21, Oct 2011, pp. 2233–2243.

[MW07] Minshew, N. J.; Williams, D. L. "The new neurobiology of autism: Cortex, connectivity, and neuronal organization", *Archives of Neurology*, vol. 64, Oct 2007, pp. 945–950.

[NZF+13] Nielsen, J. A.; Zielinski, B. A.; Fletcher, P. T.; Alexander, A. L.; Lange, N.; Bigler, E. D.; Lainhart, J. E.; Anderson, J. S. "Multisite functional connectivity MRI classification of autism: ABIDE results", *Frontiers in Human Neuroscience*, vol. 7, Sep 2013, pp. 1–12.

[OL90] Ogawa, S.; Lee, T. "Brain magnetic resonance imaging with contrast dependent on blood oxygenation", *Proceedings of the National Academy of Sciences*, vol. 87, Dec 1990, pp. 9868–9872.

[PBM15] Plitt, M.; Barnes, K. A.; Martin, A. "Functional connectivity classification of autism identifies highly predictive brain features but falls short of biomarker standards", *NeuroImage: Clinical*, vol. 7, Sep 2015, pp. 359–366.

[PHS+14] Plis, S. M.; Hjelm, D. R.; Salakhutdinov, R.; Allen, E. a.; Bockholt, H. J.; Long, J. D.; Johnson, H. J.; Paulsen, J. S.; Turner, J. a.; Calhoun, V. D. "Deep learning for neuroimaging: a validation study.", *Frontiers in Neuroscience*, vol. 8, Jan 2014, pp. 1–229.

[PMB13] Pascanu, R.; Mikolov, T.; Bengio, Y. "On the difficulty of training Recurrent Neural Networks". In: 30th International Conference on Machine Learning, 2013.

[Pow12] Power, J. D., Barnes, K. A., Snyder, A. Z., Schlaggar, B. L., & Petersen, S. E. "Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion", *NeuroImage*, vol. 59, Feb 2012, pp. 2142–2154.

[RN10] Russell, S. J.; Norvig, P. "Artificial intelligence: a modern approach". Prentice Hall, 2010, 1152p.

[Sam59] Samuel, A. L. "Some studies in machine learning using the game of checkers", *IBM Journal*, vol. 3, Jul 1959, pp. 210–229.

[SKE03] Schwardt, O.; Kolb, H.; Ernst, B. "Discovery and development of biomarkers of neurological disease", *Current topics in medicinal chemistry*, vol. 3, Mar 2003, pp. 1–9.

[SMDH13] Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. "On the importance of initialization and momentum in deep learning". In: IEEE International Conference on Acoustics, Speech and Signal Processing, 2013.

[ST10] Strimbu, K.; Tavel, J. a. "What are biomarkers?", *Current Opinion in HIV and AIDS*, vol. 5, Nov 2010, pp. 463–466.

[Vap95] Vapnik, V. N. "The Nature of Statistical Learning Theory". Springer, 1995, 314p.

[vdHH10] van den Heuvel, M. P.; Hulshoff Pol, H. E. "Exploring the brain network: A review on resting-state fMRI functional connectivity", *European Neuropsychopharmacology*, vol. 20, Aug 2010, pp. 519–534.

[VLBM08] Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.-A. "Extracting and composing robust features with denoising autoencoders". In: 25th international Conference on Machine learning, 2008.

[VLL+10] Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.-A. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", *Journal of Machine Learning Research*, vol. 11, Jan 2010, pp. 3371–3408.

[WBB+15] Wolfers, T.; Buitelaar, J. K.; Beckmann, C. F.; Franke, B.; Marquand, A. F. "From estimating activation locality to predicting disorder: A review of pattern recognition for neuroimaging-based psychiatric diagnostics", *Neuroscience & Biobehavioral Reviews*, vol. 57, Oct 2015, pp. 328–349.