SCHOOL OF ENGINEERING

DEPARTMENT OF INFORMATION AND

ELECTRONICS ENGINEERING

BACHELOR'S THESIS

«Machine learning methods for estimation of distinct brain signals produced after visual stimulus.»

Student's name:
Tsakiris
Nikolaos
Reg. Number:
154619

Supervisor Professor
Konstantinos
Diamantaras

20 Ιουνίου 2022

Μέθοδοι μηχανικής μάθησης για την εκτίμηση διακριτών οπτικών ερεθισμάτων με βάση την εγκεφαλική δραστηριότητα.

Κωδικός Διπλωματικής: 21259

Ονο/μο Φοιτητή:Τσακίρης Νικόλαος

Ονο/μο Εισηγητή: Διαμαντάρας Κωνσταντίνος

Ημερομηνία ανάληψης Δ.Ε.: 05-04-2021

Ημερομηνία περάτωσης Δ.Ε. 20-06-2022

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η πτυχιακή εργασία ”Μέθοδοι μηχανικής μάθησης για την εκτίμηση διακριτών οπτικών ερεθισμάτων με βάση την εγκεφαλική δραστηριότητα.” εκπονήθηκε στα πλαίσια περάτωσης των σπουδών στο τμήμα ”Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων” του Διεθνούς Πανεπιστημίου Ελλάδος. Η επιλογή του θέματος πραγματοποιήθηκε σε ελεύθερο πνεύμα και υπο τη καθοδήγηση του επιβλέποντα καθηγητή, κυρίου Κωνσταντίνου Διαμαντάρα. Επιπλέον, η κατάληξη του θέματος και των δεδομένων έγινε με σκοπό την αντιπαραβολή μίας υπαρκτής μελέτης πανομοιότυπου προβλήματος στο ίδια δεδομένα, ούτως ώστε να εξερευνηθεί περαιτέρω η εδραίωση υποψήφιων μηχανισμών μηχανικής μάθησης με σκοπό την βελτίωση της διάκρισης των εγκεφαλικών ερεθισμάτων, βασισμένων στο πολυδιάστατο εγκεφαλικό σήμα. Το συγκεκριμένο πλαίσιο έρευνας έχει ως στόχο αφενός την ανάδειξη των καλύτερων μεθόδων για αντίστοιχα προβλήματα με εφαρμογή σε μαζικό εύρος δεδομένων, ήτοι πολλαπλών εγκεφάλων και πειραματοζώων, και αφετέρου τη διεύρυνση της γνώσης μας ως προς την εσωτερική διακύμανση και ιδιοσυγκρασία του ανθρωπίνου εγκεφαλικού χάρτη.

Περίληψη

Στόχος της παρούσας πτυχιακής εργασίας είναι η δημιουργία ενός χρηστικού αγωγού μεθόδων μηχανικής μάθησης με σκοπό την βελτίωση των αποτελεσμάτων πρωτύτερης μελέτης. Για την υλοποίηση αυτού χρειάζεται η ορθή προ-επεξεργασία, έπειτα η προσεχτική συμπύκνωση, και τέλος η κατηγοριοποίησης των φιλτρασισμένων εγκεφαλικών σημάτων με σκοπό την ανάδειξη του σωστού ερεθίσματος, μία διαδικασία που απαιτεί μία πληθώρα τεχνικών και αρχιτεκτονικών. Ως επι το πλείστον, η εργασία χωρίζεται σε 3 κύρια μέρη. Κατά το πρώτο πραγματοποιείται η θεωρητική ανάλυση των μεθόδων που χρησιμοποιήθικαν, τόσο στη προ-επεξεργασία των δεδομένων, όσο και στο ίδιο το κομάτι της κατηγοριοποίησης. Το δεύτερο πραγματεύεται την έντονη χρήση των εργαλείων προ-επεξεργασίας τα οποία κατέστησαν τα δεδομένα πιό απτά και λιγότερο περιπλεγμένα. Μία απο τις βασικότερες παραμέτρους αυτής της επεξεργασίας ήταν η ορθή προσέγγιση των τελικών *features* που έπρεπε να διατηρηθούν συναρτήσει των περιφερειών του εγκεφάλου και των πέντε χρονικών βημάτων κατα τα οποία εξήχθησαν για κάθε μία παρακολούθηση ερεθίσματος. Τέλος, γίνεται περιγραφή της ολικής διαδικασίας διασωλήνωσης των μεθόδων, ξεκινώντας απο την προ-επεξεργασία και καταλήγοντας στη κατηγοριοποίηση και τη παραμετροποίηση του *gradient boosting* αλγορίθμου. Ωστόσο, η ίδια η φύση του προβλήματος ανέδειξε για άλλη μία φορά την ανάγκη των καθαρών δεδομένων έναντι της μονότονης προσέγγισης των αλγορίθμων εκπαίδευσης χωρίς τα κατάλληλα εφόδια. Η εργασία προσφέρει ένα γενικευμένο ολιστικό εργαλείο χρησιμοποιώντας το συγκεκριμένο στροιχειοσύνολο στην ολότητά του, παρέχοντας έτσι μία καλύτερη προσέγγιση που θα μπορούσε να χρησιμοποιηθεί και εξελιχθεί περαιτέρω για κάθε τέτοια διεργασία *decoding* του ανθρώπινου εγκεφάλου, είτε πρόκειται για κατηγοριοποίηση των ερεθισμάτων, είτε για ιατρικές γνωματεύσεις.

«Machine learning methods for estimation of distinct brain signals produced after visual stimulus.»

Nikolaos Tsakiris

Abstract

The aim of this thesis is to create a useful pipeline of machine learning methods in order to improve the results of an earlier study. Implementing this requires proper pre-processing, then careful concentration, and finally categorization of filtered brain signals and the correct prediction of the initial stimuli, a process that requires a wealth of techniques and architectures. For the most part, the work is divided into 3 main parts. The first is a theoretical analysis of the methods used, both in the pre-processing of the data and in the part of the categorization itself. The second deals with the extensive use of pre-processing tools which have made data more tangible and less complicated. One of the key parameters of this solution was the correct approach to the final *features* that had to be maintained depending on the regions of the brain and the five time steps in which they were extracted for each stimulus monitoring. Finally, the whole process of pipelining of the methods is described, starting from the pre-processing and ending with the categorization and configuration of the *gradient boosting* algorithm. However, the very nature of the problem has once again highlighted the need for clear data versus the monotonous approach of training algorithms without the proper supplies. The work offers a generalized holistic tool using this particular set as a whole, thus providing a better approach that could be used and further developed for any such process of *decoding* of the human brain, whether it is categorization of stimuli or medical reports and research.

Ευχαριστίες

Πρωτίστως, ευγνωμονώ τον Θεό που με αξίωσε να περατώσω με επιτυχία αυτή την ακαδημαϊκή πορεία, ευλογώντας και δίνοντάς μου δύναμη να ξεπεράσω τις όποιες δυσκολίες προέκυψαν, και παράλληλα για την ομαλή μετάβαση απο τις μακροχρόνιες μουσικές σπουδές σε αυτές του κόσμου της πληροφορικής, ένα ταξίδι που αποφάσισα να ξεκίνησω μερικά χρόνια πριν. Έπειτα, ευχαριστώ τους γονείς και την οικογένεια μου για την αδιάλλειπτη ψυχολογική, οικονομική, και πάσης φύσεως βοήθεια που έτυχε να χρειαστώ. Τέλος, ευχαριστώ βαθύτατα τον καθηγητή μου κ. Κωνσταντίνο Διαμαντάρα καθώς η συμβολή του ήταν εξέχουσας σημασίας, παρέχοντάς μου σημαντικές γνώσεις, συμπαράσταση, αλλα και καθοδήγηση, γνωρίζοντας κάθε φορά το πώς οφείλω να πορευθώ, με σκοπό την επίτευξη ενός καλύτερου αποτελέσματος.

Τσακίρης Νικόλαος

Contents

Πρόλογος . . . ii
Περίληψη . . . iii
Abstract . . . iv
Ευχαριστίες . . . v
Contents . . . vi
List of Figures . . . viii
Appendix . . . x

1 Introduction **1**
1.1 Related Work . . . 1
1.2 Contributions . . . 2
1.3 Structure . . . 2

2 Machine Learning Introduction **4**
2.1 Machine Learning as a Discreet Computer Paradigm . . . 4
2.2 History . . . 4
2.3 Categories . . . 4
2.3.1 Supervised . . . 5
2.3.2 Unsupervised . . . 5
2.3.3 Reinforcement Learning . . . 6
2.4 Problem Types . . . 7
2.4.1 Classification . . . 8
2.4.2 Regression . . . 8
2.4.3 Clustering . . . 9
2.5 Standardization . . . 13
2.6 Dimensionality Reduction . . . 13
2.6.1 PCA . . . 14
2.6.2 Correlation Matrix . . . 16
2.7 Overfitting vs Underfitting . . . 17
2.8 Metrics . . . 17

3 Machine Learning Algorithms **20**
3.1 Neural Networks Introduction . . . 20
3.2 Natural vs Artificial . . . 20
3.3 Feedforward Neural Networks . . . 22
3.4 Activation Functions . . . 23
3.5 Training and Optimization . . . 27
3.5.1 Backpropagation . . . 27
3.5.2 Stochastic Gradient Descent . . . 27
3.5.3 Cost Functions . . . 28
3.6 Autoencoders . . . 28
3.6.1 Encoder . . . 30
3.6.2 Bottleneck Layer . . . 31
3.6.3 Decoder . . . 32
3.7 Recurrent Neural Networks . . . 33
3.7.1 Long Short Term Memory Networks (LSTMs) . . . 34
3.8 Hybrid Modeling . . . 36
3.8.1 Autoencoder-LSTM . . . 36
3.9 Decision Trees . . . 38
3.9.1 Gradient Boosting . . . 39

4 Task and Dataset **40**
4.1 Task . . . 40
4.2 Dataset . . . 41
4.2.1 BOLD5000 . . . 42

5 Preprocessing, Impediments & Methodologies **45**
5.0.1 Data Preprocessing . . . 45
5.0.2 Predefined Problems . . . 49
5.0.3 Emerged Problems . . . 49
5.0.4 Methodologies . . . 51
5.1 Results . . . 54

6 Conclusions **56**
6.1 Challenges . . . 56
6.2 Future Work . . . 57
6.2.1 Incorporation of Scalograms . . . 57

7 Bibliography **58**

List of Figures

2.1 Class Separation. . . . 5
2.2 Automated Group Formation. . . . 6
2.3 Reinforcement Methodology. . . . 7
2.4 Classification Blueprint 8
2.5 Linear Regression Example 9
2.6 Clustering Procedure 10
2.7 k-Means Separation. . . . 11
2.8 Hierarchical Clustering Dendrogram. . . . 12
2.9 DBSCAN formation with signified outliers. . . . 12
2.10 Standardizing. . . . 13
2.11 Dimensionality Reduction Effect. . . . 14
2.12 Principal Component Analysis. . . . 15
2.13 Correlation Matrix Visualization. . . . 16
2.14 Underfitting-Overfitting vs Correct Model 17
2.15 Confusion Matrix, a method which demonstrates clearly the four basic sub-metrics. . . . 18
2.16 Area Under the ROC Curve 19
3.1 Basic Neural Network Architecture 20
3.2 Natural Brain Neuron 21
3.3 Artificial Neuron Blueprint 21
3.4 Simple Multilayer Perceptron 22
3.5 Step Function 23
3.6 Sigmoid Function 24
3.7 Tanh function 25
3.8 ReLu Function 26
3.9 Leaky ReLu function 26
3.10 Function as a 3-D graph. . . . 27
3.11 Full Autoencoder Architecture 30
3.12 Input, leading to the encoder. . . . 31
3.13 Bottleneck, containing the latent representation. . . . 32
3.14 Decoder part, aiming to reconstruct the initial data. . . . 33
3.15 Distinguishing Feature Between RNNs FFNNs 34
3.16 LSTM Fundamentals 35
3.17 AutoEncoder-LSTM Steps 37
3.18 Decision Tree Structure 38
3.19 Gradient Boosting Blueprint 39
4.1 Brain Signal Encoding. . . . 40
4.2 Brain Signal Decoding. . . . 41
4.3 Localizers giving prominence to brain's topological actuation. . . . 42
4.4 Time Induced Brain Masks & Voxels as Time-Series 43
4.5 Progress of stimuli-brain reaction coupling. . . . 44
5.1 Time Induced Brain Masks & Voxels as Time-Series 45
5.2 Animal image from COCO dataset. Dedicated animal images tend to avoid including many differentiating factors and elements that might produce further brain reaction. . . . 46
5.3 Artifact image from ImageNet dataset. Artifacts should be demonstrated as single posing elements of artificial constructions that don't incorporate any kind of deep scenery, with the context being as bland as possible and the camera focusing on the artifact itself. . . . 47
5.4 Scene image from SUN dataset. In juxtaposition, scenery images should include a plethora of colors, elements, buildings, deep context, shapes and non-focused artifacts, this providing the subject with the opportunity to view an open space with many interconnected pieces of furniture or other stuff. . . . 48
5.5 Image from the ImageNet dataset, presenting a blurred class distinction. Inversely, this image presents the exact problem of indiscriminately distinguished classes, since inside of it exists a fence, meaning an artifact, grasses and plants, and even a scene in the background, so both of our classes of interest can be contained in this specific image, so each person's brain might react differently. . . . 49
5.6 Animal Class Scalogram 50
5.7 Artifact Class Scalogram 50

5.8 Scene Class Scalogram 51
5.9 Encoder Compression 52
5.10 Proposed method 53
5.11 Plain LSTM method of [12]. 54
5.12 Normalized Confusion Matrix results of proposed method. 55
5.13 Confusion Matrix results of proposed method. 55

Appendix

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
CS	Computer Science
AI	Artificial Intelligence
ML	Machine Learning
SL	Supervised Learning
RL	Reinforcement Learning
PCA	Principal Component Analysis
MLP	Multilayer-Perceptron
FFNN(s)	Feed Forward Neural Networks
NN(s)	Neural Networks
ANN(s)	Artificial Neural Networks
GD	Gradient Descent
AE(s)	Autoencoder(s)
RNN(s)	Recurrent Neural Network(s)
LSTM	Long-Short-Term-Memory
CNN(s)	Convolutional Neural Network(s)
AutoLSTM	Autoencoder-LSTM
BOLD	Blood Oxygenation Level Dependent
fMRI	functional Magnetic Resonance Imaging
ROI(s)	Regions of Interest
PCA	Principal Component Analysis
DT(s)	Decision Tree(s)
GB	Gradient Boosting

Chapter 1: Introduction

Since the large adoption of machine learning frameworks, many problems are being solved via those methodologies, ranging from fashion designing, to disease prevention and autonomous driving. Ironically enough, neural network architectures are inspired by the brain's ones, so it's a remarkable feat to actually use them for the examination and research of their very first element of influence. This particular endeavor has gained significant traction, approximately in the last 12 years, where more datasets are being available, giving the opportunity to scientists and engineers to amplify their efforts.

Although Brain functions are an important yet still developing area of research, they still pose as a black box, with the research of interest being separated into two main tasks: those who examine the brain in order to draw medical conclusions, and those who are interested at the topological, functional, and cognitive side of the brain, whether it's about comprehending the inner functions of it, or the reaction to an external component and how this information is formed inside of it.

This work examines the case of brain's decoding, meaning the classification of the stimuli that caused a certain reaction to it. The benefit of this type of work manifests as the progressive memo of what components of the brain are eventually used for when viewing a certain stimuli, therefore eliminating or adding more regions of it, resulting in a more solid declaration of cause & effect. Moreover, the approach this paper follows is that of the all-in philosophy, where the whole dataset is being considered as one brain, incorporating all the differentiating, and sometimes problematic features of each separate subject, thus providing a more robust mechanism, ready to tackle physiological, psychological, or even incomplete data.

The main idea of this work is based on a heavy dimensionality reduction framework, before feeding the data into a classifier. This reduction must be executed in a careful manner, since fMRI data tends to have a significant number of dimensions, while also taking into consideration that fMRI gets extracted through time steps, so this compression should be generic enough for all the subjects in order for the final outcome to present a true representation of the initial multi-faceted, multi-subject points.

1.1 Related Work

Since the outbreak of brain related datasets in recent years, many people have opted to discover and apply machine learning methods on them, with a significant proportion of them trying to extract meaningful results from the existing datasets, yet there still was an important lack of a truly large dataset, incorporating many subjects **and** classes, thus providing us with a truly grand-scale view of the brain itself. That's exactly what BOLD5000 [13] offered, with the large amount of images being used as stimuli for four subjects of different physiological and psychological disposition.

Related work to this dataset is presented at [12], whose authors took the initiative to formulate the BOLD5000 dataset in a certain manner and used three distinct classes for labeling, hence animals, artifacts and scenes. Their work incorporated a plain LSTM model, which produced significant results, yet lacked the any dimensionality reduction aspect of the temporal information, while also being unable to correctly separate scenes from artifacts, and especially animals, which is a truly important feat since

scenes might incorporate animals inside their field of view.

Moreover, the authors at [12] presented a straightforward yet simplified method by adopting the usage of a plain LSTM model, thus giving the chance for a further approach both at the pre-processing and the classifying part itself.

Although the above work presented a respected result, it didn't offer a generic pipeline approach for an all-in procedure which might potentially work on another set of subjects with differing characteristics and traits, leaving open a good potential for additional work on a generalized method, firstly by incorporating the element of data compression which must work no matter the number of subjects and classes, and secondly, a more refined and dedicated approach for a classifier which can easily be manipulated to the needs of each compression outcome.

Finally, it was critical to offer a more robust solution, so this paper examines the BOLD5000 dataset based on the initiated data format provided by the authors of [12] so that more research is conducted on this particular framing and three distinct classes.

1.2 Contributions

The contributions of this paper can be presented as a compact list, below:

- The research of an existing voxel dataset that was framed in a particular set of classes.
- The construction of a generalized pipeline that takes upon both the tasks of heavy-preprocessing - a much needed procedure for BOLD data - **and** classification of such data.
- A better result on the separation of 2 of the classes which can be overlapped, thus animals and scenes.
- A clearer understanding of the underlying mechanisms and the regions of interest that trigger certain brain parts based on a distinguished stimuli.
- The usage and proof that a gradient boosting classifier can be deployed for such a problem, giving prominence to its ability and versatility, much needed when creating a machine learning pipeline which expects heavily compressed fMRI data.
- Providing an expandable tool for further research.

1.3 Structure

Chapter 2 More specifically, in the second chapter there is an introductory reference to the basic principles of the broader machine learning, as well as the problems that it is capable of solving.

Chapter 3 In the third chapter, an analytical reference is made to machine learning algorithms, and especially in neural networks, and in particular to autoencoders, an important element in reducing the dimensions of signals. Also, reference is made to long-short-term memory networks, as well as

to the hybrid neural network architecture, as the autoencoder consisted of LSTM elements, being necessary to complete tasks on sequential data. Moreover, the configuration of decision trees is being presented as well as the technique of gradient boosting, as this is the final classifier and the last piece of the pipeline.

Chapter 4 The fourth chapter defines the structure, components, and problems of this data nature, as well as a special reference to the bold5000 typeset, as it is in itself a remarkable work that illuminates the human brain from many angles.

Chapter 5 The fifth chapter refers to the whole process of the pipelining methods, starting from the pre-treatment and ending with the results, summarizing among the pre-anticipated problems, as well as the upcoming ones that were presented during the conduct of the research.

Chapter 6 Subsequently the sixth chapter completes the work, capturing the results.

Chapter 7 Lastly, the seventh chapter offers the future perspective of this study and the new methods that might be used for this particular dataset formation.

Chapter 2: Machine Learning Introduction

2.1 Machine Learning as a Discreet Computer Paradigm

When people refer to Artificial Intelligence (AI), they make the assumption of a hyper-intelligent entity capable of superhuman feats, yet the truth is of a different, humbler caliber. First and foremost, it's important to distinguish between different types of AI, while simultaneously explain their main characteristics, what they can offer to the world of Computer Science (CS), and their key points of difference with traditional computer paradigms. Frankly, one of the most obvious yet concealed types of AI is called Machine Learning (ML), with its applications being an important part of people's lives, usually without them realising.

ML differs from classical paradigms due to its guiding inherent philosophy of making the computer "think", "predict", "assist" and reach conclusions without the need of an explicit set of instructions manifested as the programming guidance of a human. Subsequently, the computer has to "learn", and this process begins with the observations of data so that it can extract patterns and information, and eventually be better at responding to similar data at a later time. Consequently, one can describe the ML paradigm as such: **Machine Learning aims to make the computer learn in an automatic way and adjust its functions accordingly, with minimal human effort.**

As an example, one might consider the functionality of a bipolar *if* clause. If the first condition is met, the algorithm proceeds to the first block of code, or alternatively, the second block is accessed. This type of determinism is a fixed paradigm of a limited pool of inputs and certain outcomes. In fact, it's a natural way of thinking to humans, yet it does not offer much in terms of the requirements of broader and harder problems.

2.2 History

Machine Learning (ML) plays a vital role into our everyday lives since it's incorporated in almost any device we use since its learning abilities makeσ a great tool for efficiency, speed, and smarter usage of time. Interestingly enough, until 1970s, it was an embedded part of the wider world of AI, with its first baby steps as a separate and distinct block of science starting with the start 1970s onwards. The very first model was created by Donald Hebb in 1949 and presented the same premise as today were natural brain cell interactions are the inspiration of a big chunk of the corresponding algorithms. Subsequently, the next and truly remarkable emergence was that of the Perceptron, created by Frank Roseblatt in 1957, demonstrating the very fist formula for the neural network architecture based on previous works, yet his ideas faded up until the final resurgence of machine learning research during the decade of 1990s, leading to today with its wide and global adoption.

2.3 Categories

As a general rule, machine learning algorithms are placed into three distinct subgroups based on their underlying way of execution, arrangement, how they address the respective datasets and what's their core training process. Below are presented those three categories.

2.3.1 Supervised

Supervised Learning (SL) uses the paradigm of input-output mapping, meaning that the model is trained on input training examples with specified and known targets, thus, you know what to expect and what not to. Essentially, the polarity of right and wrong exists, since the model must learn specific patterns with their corresponding results.

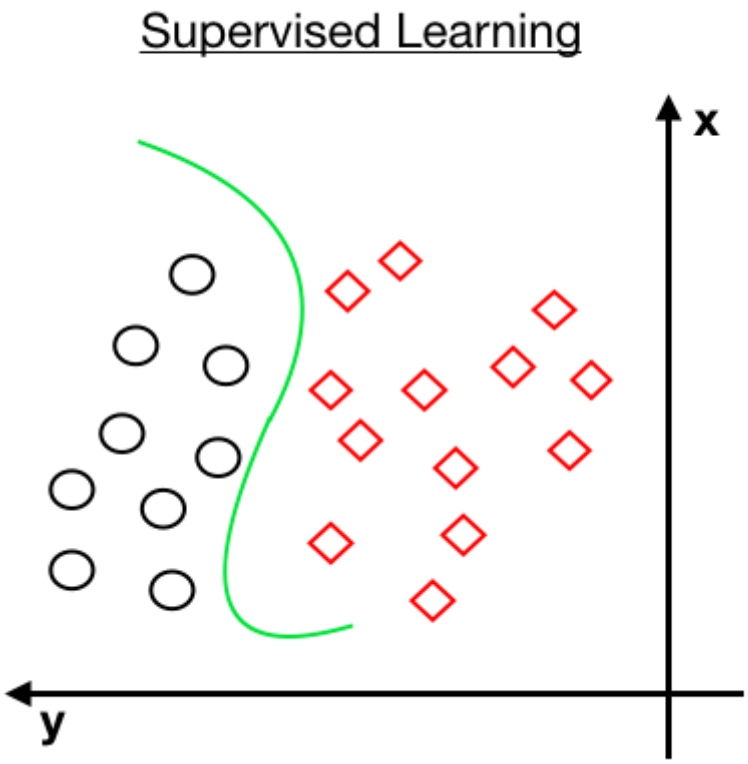


Figure 2.1: Class Separation.
Source

Essentially, such algorithms usually need a large amount data in order to produce a sufficient performance, so that it can correctly learn the relationship between points and labels, so the more a pattern is formed, the better the results. Moreover, supervised algorithms are generally separated into two distinct groups, **classification** and **regression** ones.

2.3.2 Unsupervised

The goal of Unsupervised Learning (UL) is to make sense and bring order into data with no apparent and clear target values. Reversely from SL, UL tries to find underlying structures, it exhibits a self-organizing nature and can correctly capture or build novel output data.

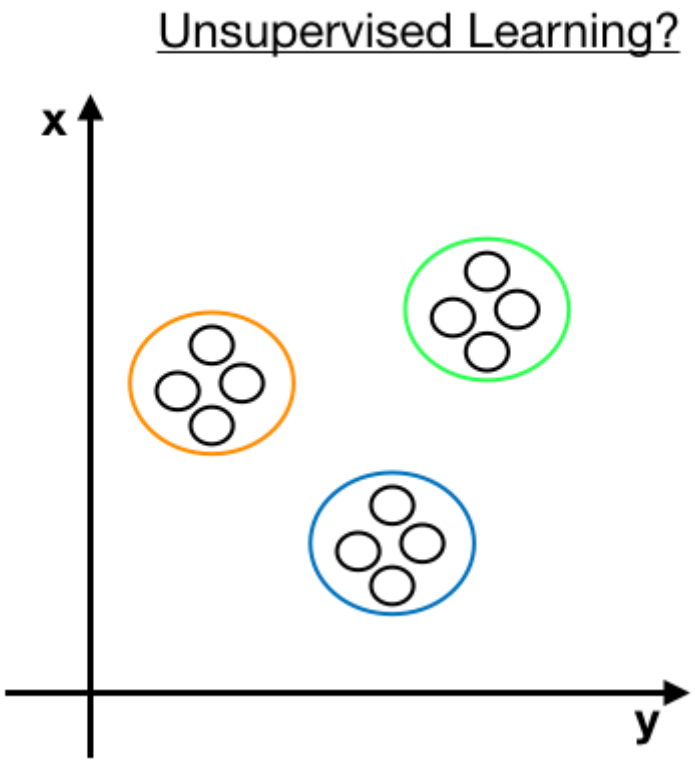


Figure 2.2: Automated Group Formation.
Source

Unsupervised learning thrives when data is unclear and messy, since its job is to actually generate the classes on its own instead of relying on already predefined labels, and are separated into the below categories:

- Clustering
- Autoencoders
- Association
- Anomaly Detection

2.3.3 Reinforcement Learning

Reinforcement Learning (RL) uses the idea of agents who can manifest a set actions and navigate through an environment based on rewards and punishments, with the goal of maximizing the former and avoiding the latter. Correspondingly, RL basis sets the proposition of balancing exploitation of current knowledge and exploration of unknown territory, with no need of labeled data nor any kind of output error adjustment whatsoever.

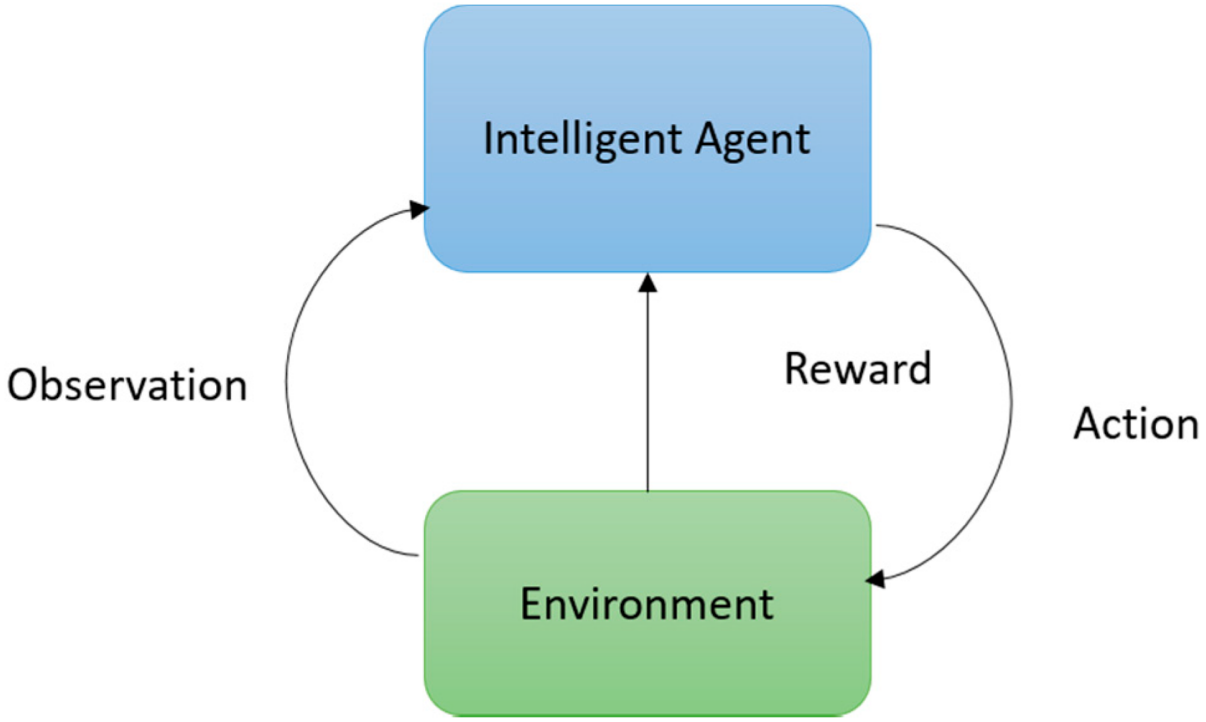


Figure 2.3: Reinforcement Methodology.
Source

The basic element of these algorithms are the agents who must solve a gradual problem in order to reach a certain goal, so they become rewarded or punished accordingly, in accordance to their ongoing performance. In general, two of the most prominent methods are presented below:

- Q-Learning, which aims at creating a scheme were Q values are updated based on the performing actions *a* and current state *s*.
- Markov Decision Process, where a certain amount of states is predisposed alongside its corresponding actions and rewards for each one, while adding the transition model, yet this methods generally need prior knowledge, thus not that optimal for uncertain environments.

2.4 Problem Types

The very first way to discriminate between the problem types machine learning can solve is that based in the feedback of the system itself, meaning that there has to be a certain interaction between the model's input and output.

Secondly, another collective can be formed when considering solely the outputs of the problem, completely ignoring how the input is formed and what's exactly its influence on the progress whatsoever.

Starting here, the following subsections will offer a compact yet analytical summary of those clusters and how we can use them in order to acknowledge *when* is the correct moment to apply each one of them.

2.4.1 Classification

Such problems belong to the supervised algorithms cluster and aim to solve a quite straightforward problem; what's the true identity of each newly presented data-point, and what's the most probable data group that I should integrate it into? Moreover, models aiming at solving classification problems are being built on the basis of previously acquired data in order to create a generic platform which might tackle potential problems, like outlier values which might reduce its accuracy capabilities.

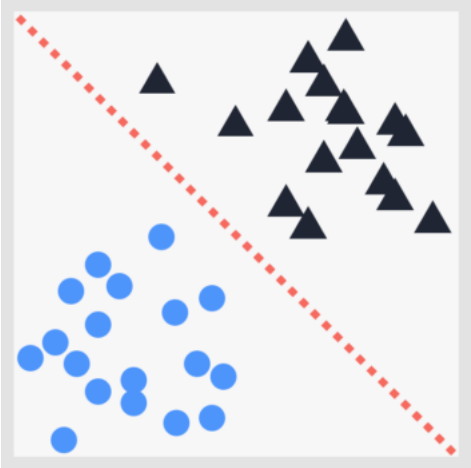


Figure 2.4: Classification Blueprint
Source

Furthermore, considering a new data point *n*, two types of classification problems emerge, multiclass and multilabel. Following the expression below,

$$f : R^n \rightarrow \{1...k\} \tag{1.1}$$

a multiclass solution will require from the model to specify only 1 of the available classes, *k*, whilst a multilabel one permits more than one options *n*, based on the available *k* classes, and being less or equal than the latter.

2.4.2 Regression

Regression belongs to the supervised algorithms cluster, where input features and labels are used too, yet, such problems require a different approach, since their goal is to fit a certain line and educe a generic estimation of how each variable influences the other(s), instead of predicting the most defining group for a newly introduced point. Importantly, the most basic expression of regression is stated as:

$$\gamma : \alpha + \beta\chi + \epsilon \tag{1.2}$$

where γ is the dependent variable (estimation), α the **Y** intercept, β the slope coefficient, χ the independent variable, and ε the error. An important aspect of regression problems is the concept of **variance**, which is directly influenced by the data's plurality. In essence, variance is the percentage of the target function differentiation in case of dissimilar or novel data introduction. Therefore, a good regression model abides by the law of generalization, thus, little variance translates into a robust model since it can

successfully transfer its estimation capabilities to other data-sets. Lastly, **bias** is also present and manifests as the algorithm's inclination to favor certain aspects of the data whilst being unable to produce correct estimations, possibly creating a chain of wrongly assumptions, which directly change the course of the regression objectivity and line's truthfulness, whether linear or polynomial.

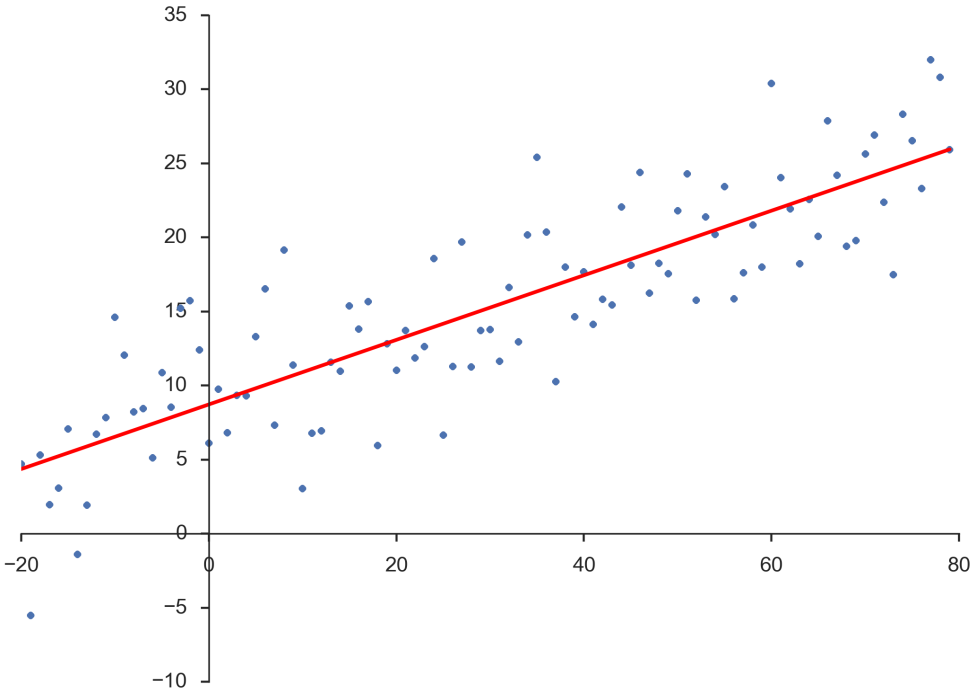


Figure 2.5: Linear Regression Example
Source

2.4.3 Clustering

The method of grouping together unlabeled points of data is called clustering. In short, such algorithms try to foster the distinguishing features of the data, therefore forming their own classes and making assumptions of the affiliation of each sample, and this is being carried out through a term called **similarity measure**. Usually, clustering is a foremost method in a machine learning pipeline since its innate abilities can aid the visualization of the data, presenting a more clear and transparent apprehension and how it can be properly framed.

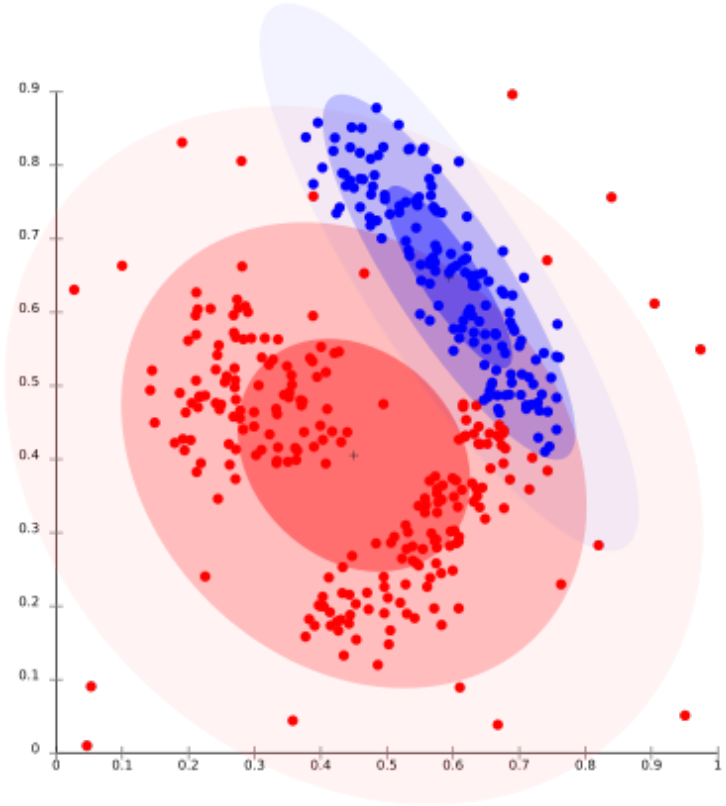


Figure 2.6: Clustering Procedure
Source

Hence, clustering has the tendency of *group formation* instead of presenting a single or multiple dividing lines between the points, a trait quite powerful since it doesn't lack effect on high dimensions. Subsequently, the more homogenous the points are, the more they can be expressed as a single and separate formed cluster, so they might be represented by the most defining one inside their group, therefore aiding **dimensionality reduction**. Lastly, three of the most known clustering methods are:

k-Means Clustering, where partitions of the data are separated into K distinct groups,

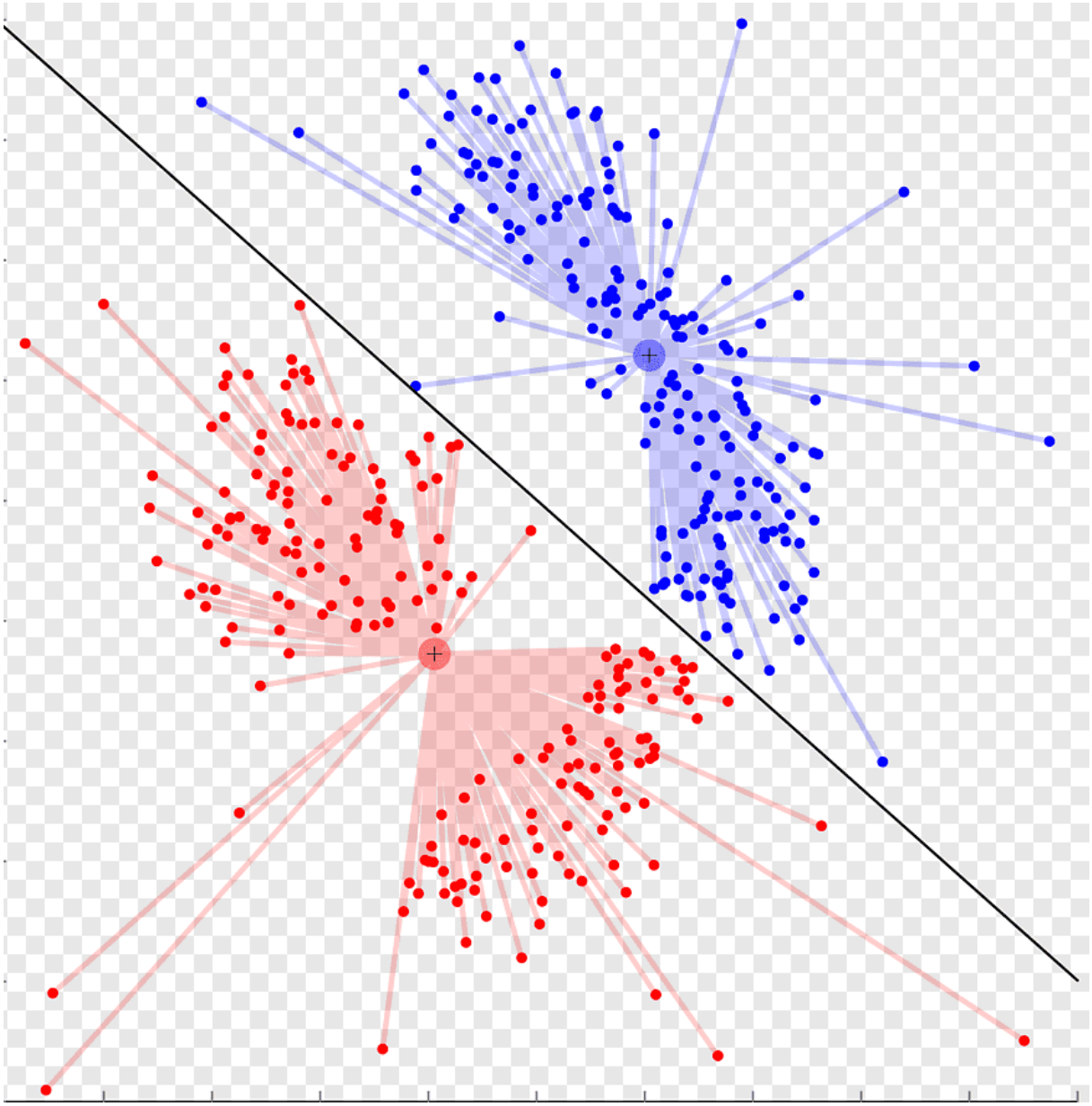


Figure 2.7: k-Means Separation.
Source

Hierarchical Clustering, which aims at progressively connecting closely related datapoints, starting by considering each point a single cluster and finally reaching a interconnection,

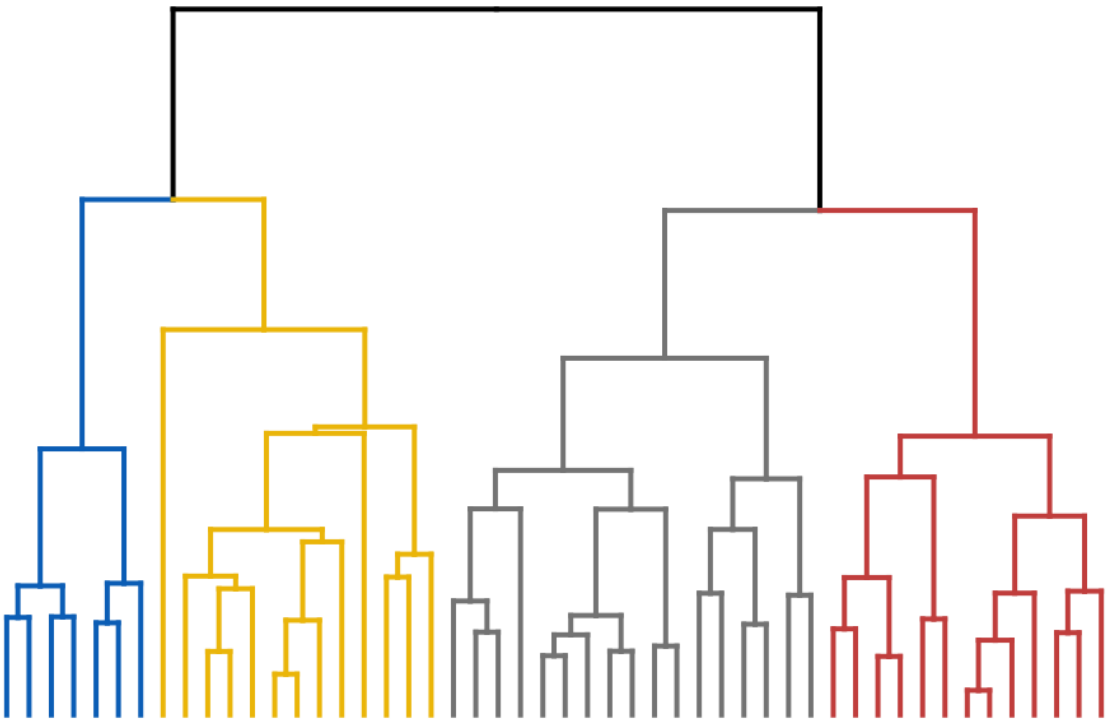


Figure 2.8: Hierarchical Clustering Dendrogram.
Source

and

DBSCAN, a powerful algorithm incorporating the concept of noise, where density-based groups are formed, with each outlier being considered a definite noise, thus omitted.

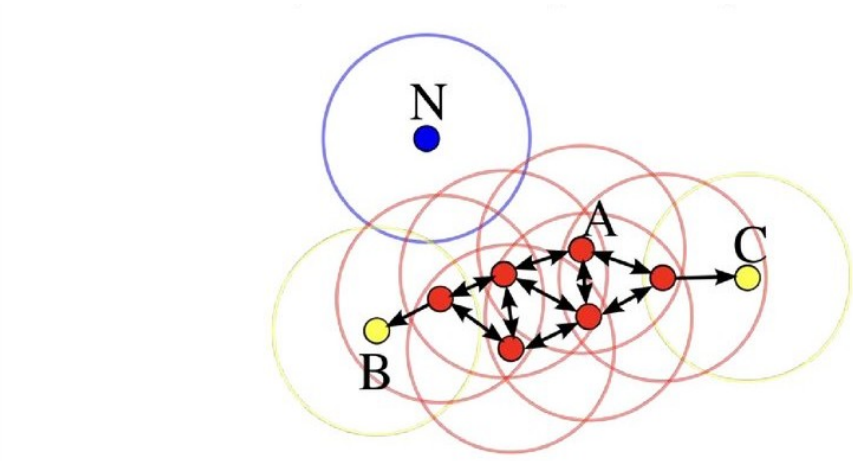


Figure 2.9: DBSCAN formation with signified outliers.
Source

2.5 Standardization

Before any endeavor conducted upon datapoints, whether pre-processing or training, one should always take into consideration the distribution of their corresponding data. Therefore, standardization, also called Z-Score Normalization, has become the norm for bringing multi-faceted data into the same scale. Two of the core elements that are included inside the Standardization procedure are the:

- Variance, and
- Standard Deviation.

Variance is extracted by taking the average of the squared difference of the mean, as presented at the equation below:

$$\sigma^2 = \sum(x_i - x)^2/n - 1 \tag{1.3}$$

Standard Deviation is the square root of variance, represents the average aberration of the values from the mean, and is being demonstrated at the expression below:

$$\sigma = \sqrt{\sum(x_i - \mu)^2/N} \tag{1.4}$$

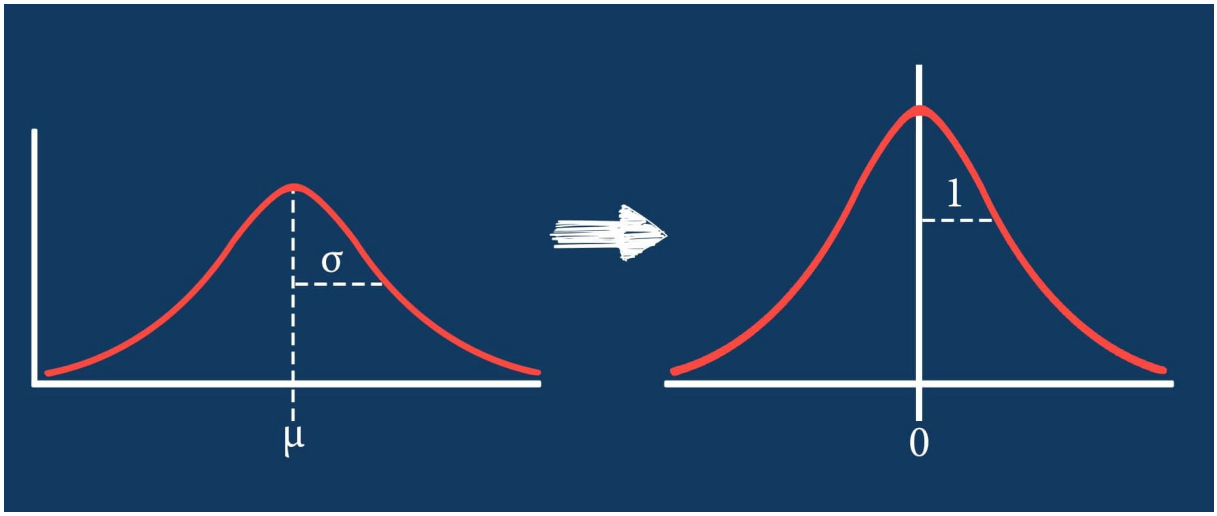


Figure 2.10: Standardizing.
Source

As such, the final goal of standardization is to place the dataset inside the spectrum of having a normal distribution of 0, and variance of 1. Therefore, scale differences are eliminated.

2.6 Dimensionality Reduction

Machine Learning is supposed to aid human activities, so real world data always pose the epitome of benchmarks for whether an algorithm, a method, or a mechanism solve a real problem. Hence, real world data tends to incorporate many pitfalls that formulated or synthesized data don't, like missing values,

false or not representative features, and **high dimensions**. This last trait is a very usual problem that causes a large hindrance to plain algorithms, so that their generalizing abilities cease to offer any real benefit, rendering them obsolete.

One solution to that problem is the introduction of fresh and more data, so that the algorithm can explore a bigger scheme and try to fill the missing spots by increasing the pool where the data manifests itself. Aside from that, a quite useful, extensively used and tremendously helpful method is that of **Dimensionality Reduction**, which aims at reducing the complexity of the features at hand by mapping the features to a lower dimension, acquiring a plethora of benefit:

- Data can be easily visualized thus offering a more direct and less blurred view of it.
- Training becomes easier, faster, and with more promising results.
- Independent, highly correlated variables get dropped accordingly, since they increase the complexity yet don't offer any utility on the model's generalizing ability.
- Battling low variance which decreases performance by comparing the variances of the continuous features.

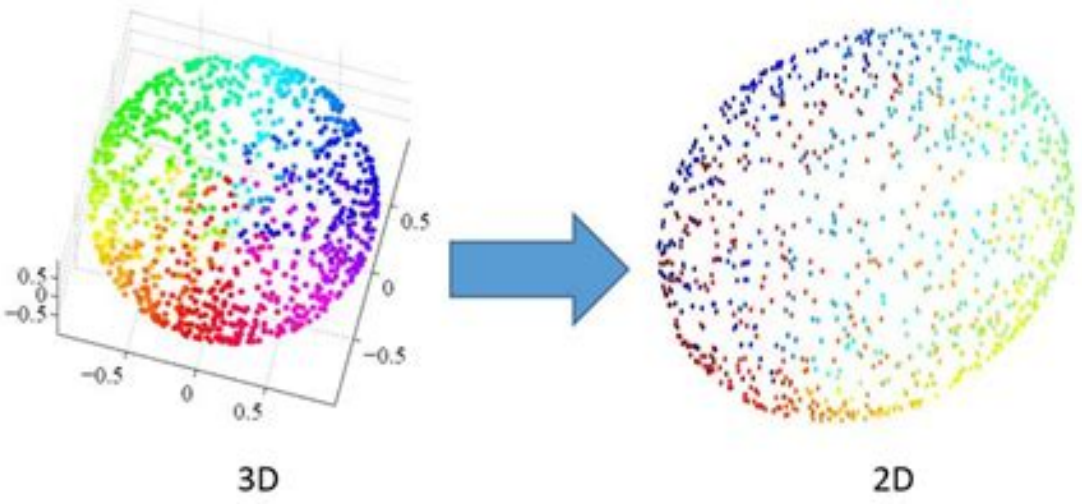


Figure 2.11: Dimensionality Reduction Effect.
Source

2.6.1 PCA

Principal Component Analysis (PCA) is one of the most prominent algorithms for applying dimensionality reduction. Essentially, this algorithm is based on the transubstantiation of a **high dimension vecto** to a lower one. One should note down that PCA follows a **linear** approach, which transforms the old high dimensional data to new, uncorrelated ones which are called the principal components.

Moreover, PCA aims at assisting the dimension reduction by keeping the greatest possible variance. Subsequently, the formula that it follows is that of the first and foremost component which accrues the

largest variance, and then follow the rest components, with a gradually reduced variance. Assuming that the data is linearly separated, PCA can successfully construct the global structure and accent the patterns, yet non-linearity can easily make this ability pointless.

To futher undrestand PCA, one can imagine a dataset of 30 variables, with a variable being manifested as a single dimension. This dimensionality is impossible to visualize, and could be proven to be a limited approach for training, so PCA transforms those dimensions to a limited set of highly distinguishable components.

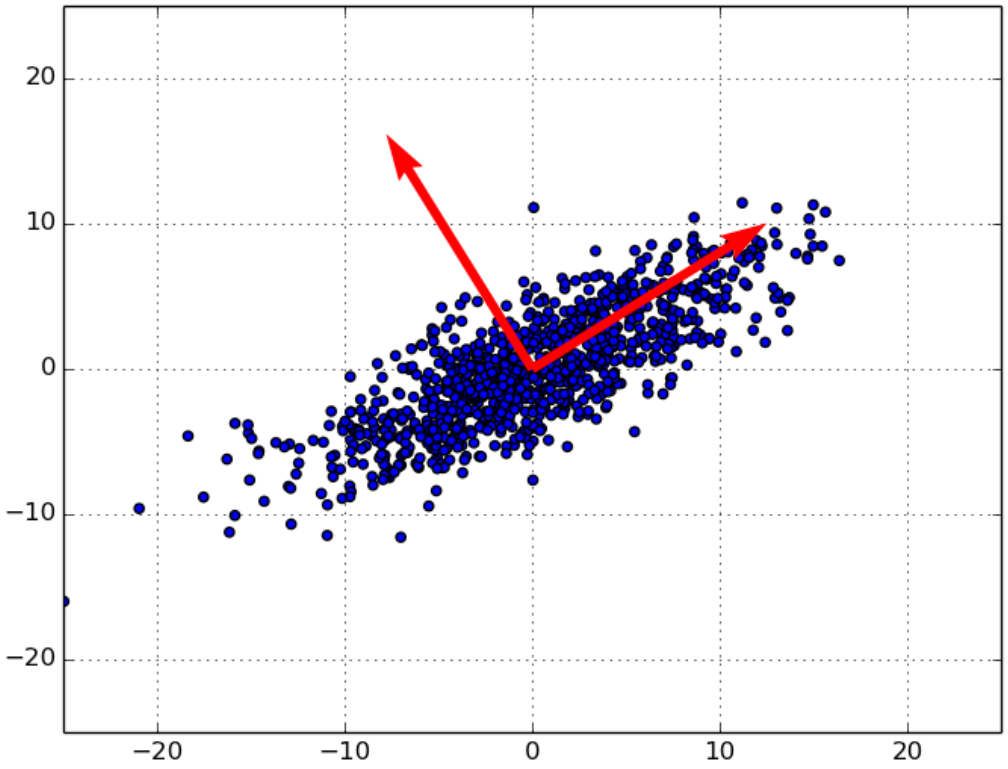


Figure 2.12: Principal Component Analysis.
Source

PCA is quite handy especially for bio-datasets, which tend to form highly dimensional sequencies of data, resulting in a great need of pre-processing before actually being fed to a model. In addition, PCA benefits can be summarized as the set below:

- Correlated features are set aside in order to increase variance.
- Improves visualization capability.
- Improves algorithm performance.
- Overfiting can be limited.

2.6.2 Correlation Matrix

A correlation matrix is typically represented as a table which contains the interconnected variable coefficients, with each cell presenting the according coefficient of each coupling, with its usage being helpful for information gain, variable erasure, and advanced insight.

The typical table places the variables in rows and columns so that every single feature is presented in accordance to every other one, with the main diagonal being the self-correlation, and the parts below and above it being mirrored, since the matrix is symmetrical. The most usual correlation statistic is that of *Spearman's Correlation*, which is non-parametric and quite insusceptible to outliers.

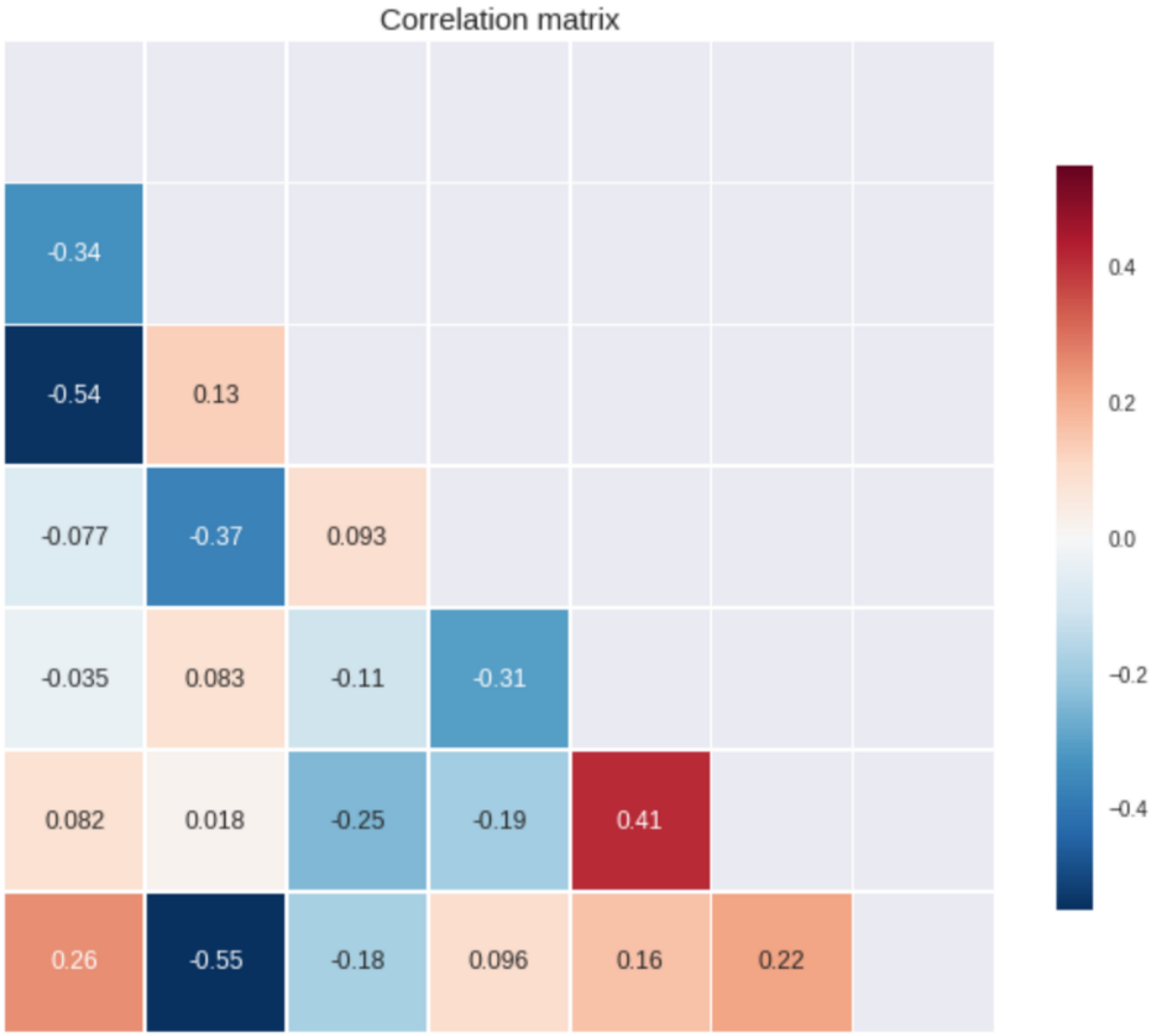


Figure 2.13: Correlation Matrix Visualization.
Source

Key aspects of a correlation matrix can be expressed as:

- The ability to clearly demonstrate the patterns of heavily crowded data.
- Pre-processing insight before applying certain dimensionality reduction techniques.

- Further analysis before a linear regression model.

2.7 Overfitting vs Underfitting

Even after a successful hyperparameter tuning and data preprocessing, data, and especially real-world data, has the tendency to *not be perfect*, so there's no model infalibility nor a perfectly composed data distribution due to the existence of outliers, uneven distributions, problematic class balancing, or any kind of an uknown varible which might contribute to error prone models. As a result, there's a two part explanation of whether a model is not properly adjusted, and this is directly manifested on its final outcomes.

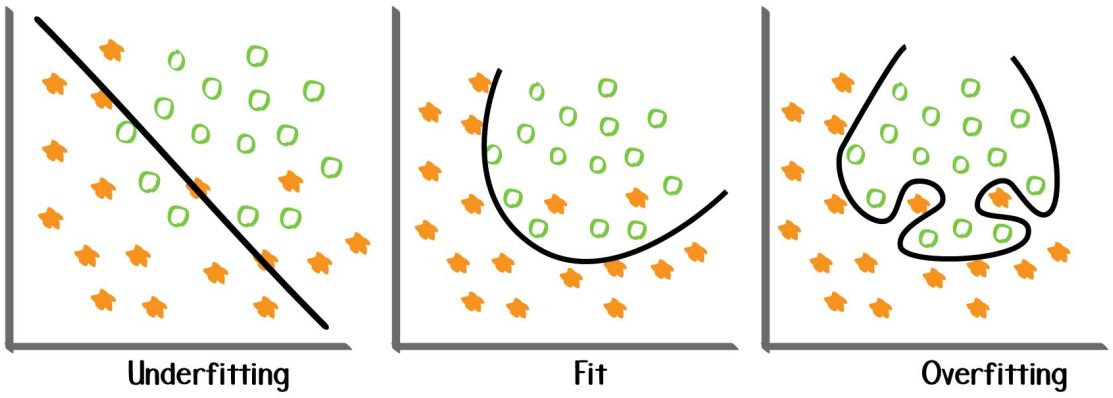


Figure 2.14: Underfitting-Overfitting vs Correct Model
Source

- **Underfitting**, representing the model's tendency to fall into the pitfall of underwhelming mapping between inputs and outputs, therefore not being able to place the data into any kind of distinction. This problem usually happens when bias is high so it can't offer reach a holistic approach of the datapoints, or even when the model is too simple to be able to catch underlying relationships.
- **Overfitting**, which means that the model has the tendency to memorize rather generalize, thus prone to dependence on any kind of noize, outliers, eminent flunctuations. Consequently, generalization is not fulfiled. Overfitting is very common when a model is overly large and therefore being **too** able on the corresponding dataset.

2.8 Metrics

Confusion Matrix, a useful descriptive table which aids at the evaluation of a model's response to real data compared to test predictions. Establishing its usage, one should note down a few important terms:

Accuracy, meaning the portion of total correct predictions,

Sensitivity/Recall, which means the **actual** positive cases correctly classified,

Specificity, describing the portion of **actual** negative, correctly identified points, and

Precision, a term representing the positive points correctly classified.

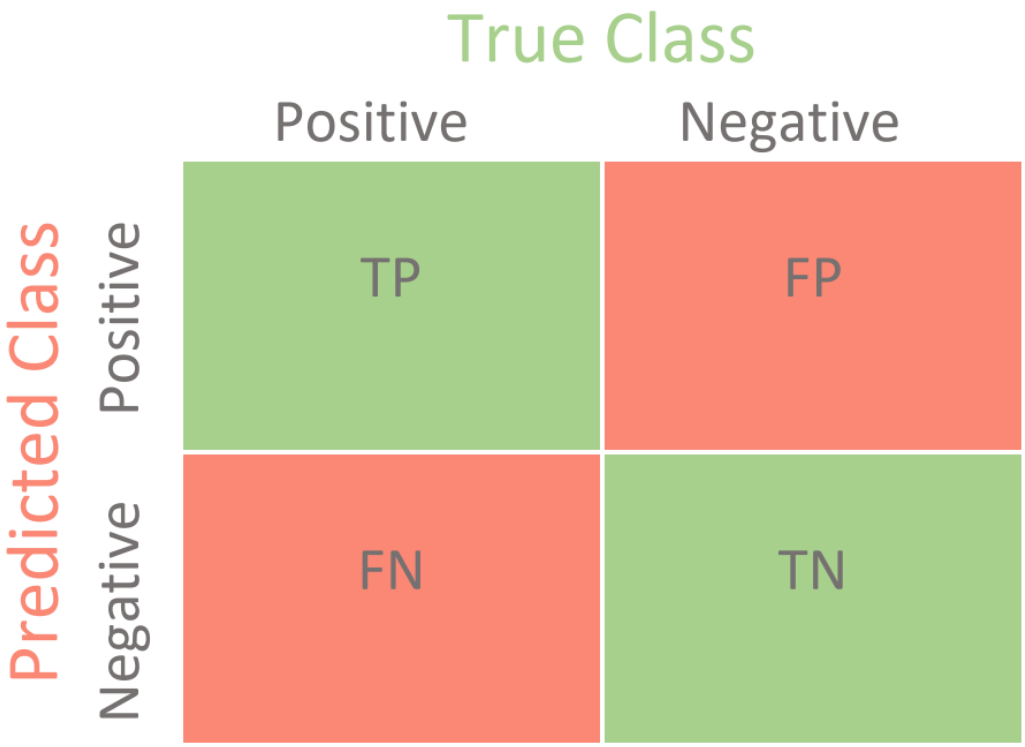


Figure 2.15: Confusion Matrix, a method which demonstrates clearly the four basic sub-metrics.

F1 Score, which is the representation of the best balance between the aforementioned concepts of precision and recall. Subsequently, such high score signifies good predictive power, and can be expressed as:

$$F1 = \frac{recall^{-1} + precision^{-1}}{2} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{1.5}$$

ROC - Area Under the Curve, or alternatively, Area Under the curve of Reicver Operating Characteristic, is an visual expositor for classification models, demonstrating the class distinguishing capability of a binary classifier. The closer the curve is to the central straight line, the worse the result, signifying a potential random guess, rather a truly distinguishing capability. The farther from the diagonal line, the better the result. Moreover, the understood drawn vertical lines for each single point present the corresponding *True Positive:False Positive* ratio, so as the more the curve travels to the upper left corner, the more this ratio leans towards the antecedent.

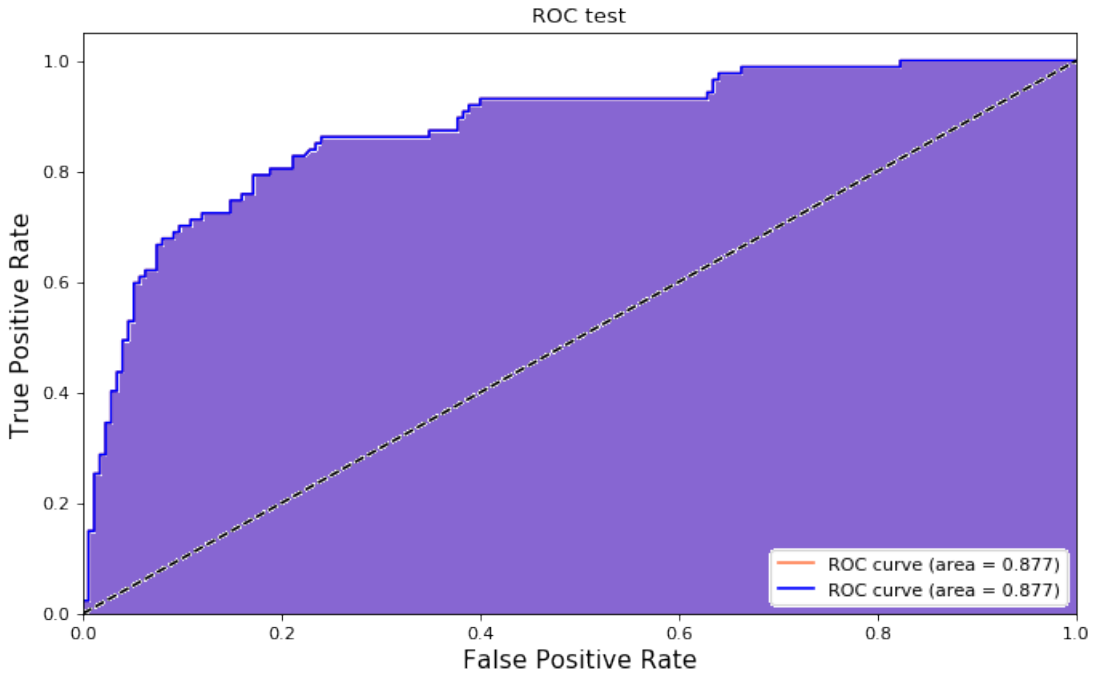


Figure 2.16: Area Under the ROC Curve

Chapter 3: Machine Learning Algorithms

3.1 Neural Networks Introduction

Artificial Neural Networks (ANNs) serve a fundamental part of humans trying to imitate nature's capabilities and channel them into technological advancements and attainments. As a general rule, **ANNs** present a modeled version of the biological ones by creating interconnected weighted nodes , and they have demonstrated a remarkable ability at adaptive control, function approximations, predictions-classifying of unseen datapoints, self-driving cars, stock predictions, etc.

Two notable figures of the ANN's early history are Warren Sturgis McCulloch and Walter Pitts, an neurophysiologist and logician respectively, who presented the first type of contemporary computational models based on algorithms.

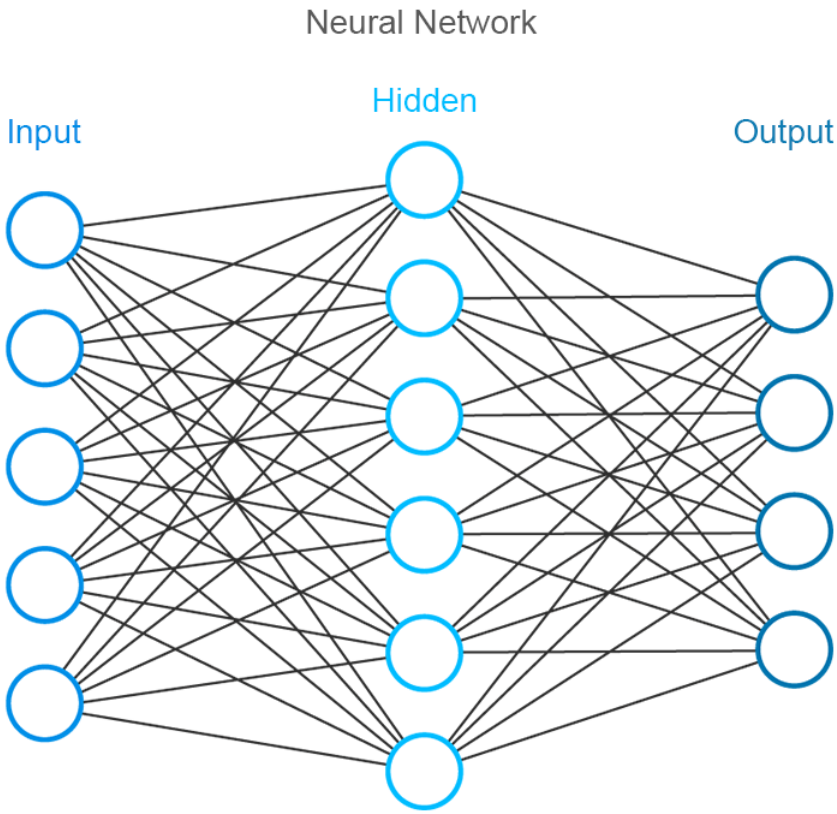


Figure 3.1: Basic Neural Network Architecture
Source

3.2 Natural vs Artificial

Brain connections are comprised of neural cells, also called neurons, which have the ability of neural impulse, meaning the arousal to electrical signal. Those cells follow a tripolar architecture of the body, the dendrites, and the axon, and are typically connected together via the synapses. Moreover, natural neurons can be dissevered into three distinct groups. Those of **sensory** ones, which take upon the task of

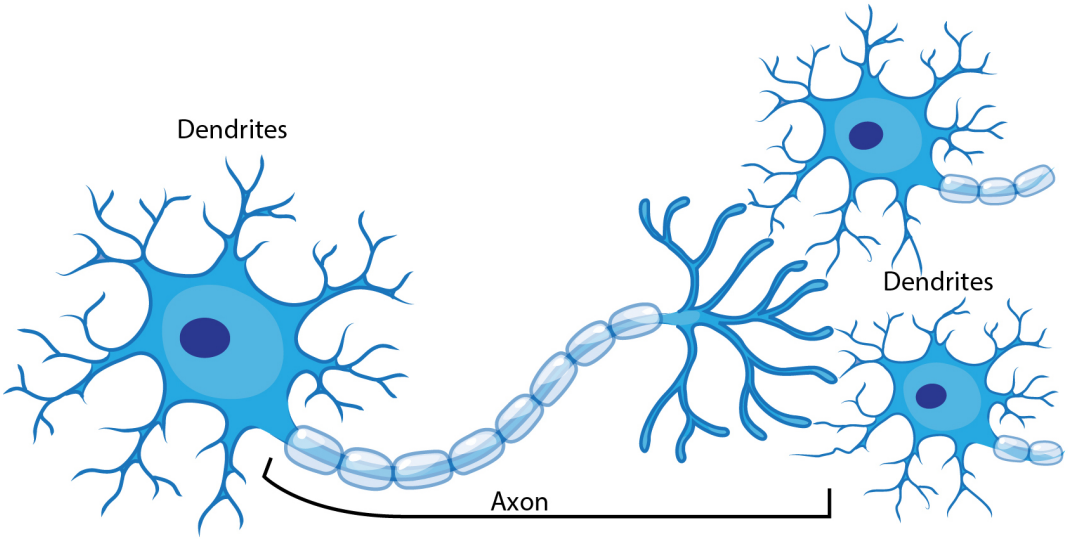


Figure 3.2: Natural Brain Neuron
Source

responding to signals directly produced from the five senses, like touch or smell, **motor** neurons, which follow the commands of the brain itself and pass according signals to the muscles, and the **interneurons**, those responsible for the internal liaison of the neuron-to-neuron interfaces of the brain or the spinal cord.

In juxtaposition, artificial neurons follow a simpler approach. As a whole, they accept single or multiple inputs and produce the sum of them. Furthermore, they usually incorporate the idea of weights, which put a different focus and gravitation to each neuron, thus amplifying or hushing their underlying power in a contextually suitable manner. As shown below, given certain signals x{0...m} and weights w{k0...km} (where x0 and wk0 here represent the fixed bias), the output yk is the result of the threshold function φ (phee). Comparably, this output is the equivalent of the biological **axon** which transfers the final value to the next destination via the **synapse**.

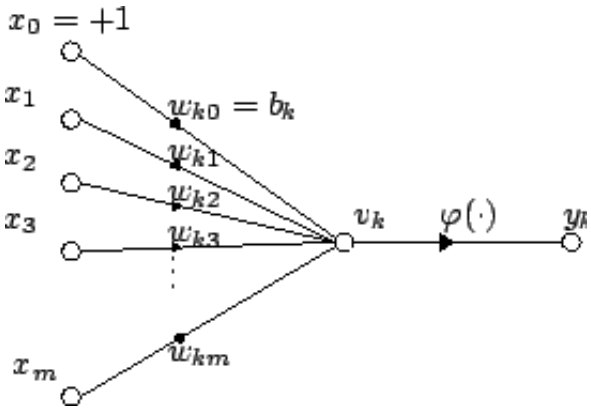


Figure 3.3: Artificial Neuron Blueprint
Source

3.3 Feedforward Neural Networks

Multilayer Perceptrons (MLPs), also known as Feedforward Neural Networks, are a special yet commonly used types of neural networks, typically applied through multiple variations, architectures and blueprints. Their basic charge is to work a function approximators, meaning the mapping of an input x to an output class y, with the base function as:

$$f(x) = y \tag{2.1}$$

MLPs create the mapping by incorporating the learning of parameters θ that arrange the best function approximation accordingly, and is stated as:

$$f(x;\theta) = y \tag{2.2}$$

Their name (feedforward) stems from their direct flow of computations, where the input is passed down through the weighted nodes and the output is produced after applying the corresponding threshold function. Their nature does not contain any type of recurrent or backwards procedures whatsoever. Additionally, their construction is formed as the order below:

1. **input** layer,
2. **n-hiden** layer(s), which can be multiple and stacked, creating a deeper flow, and
3. **output** layer.

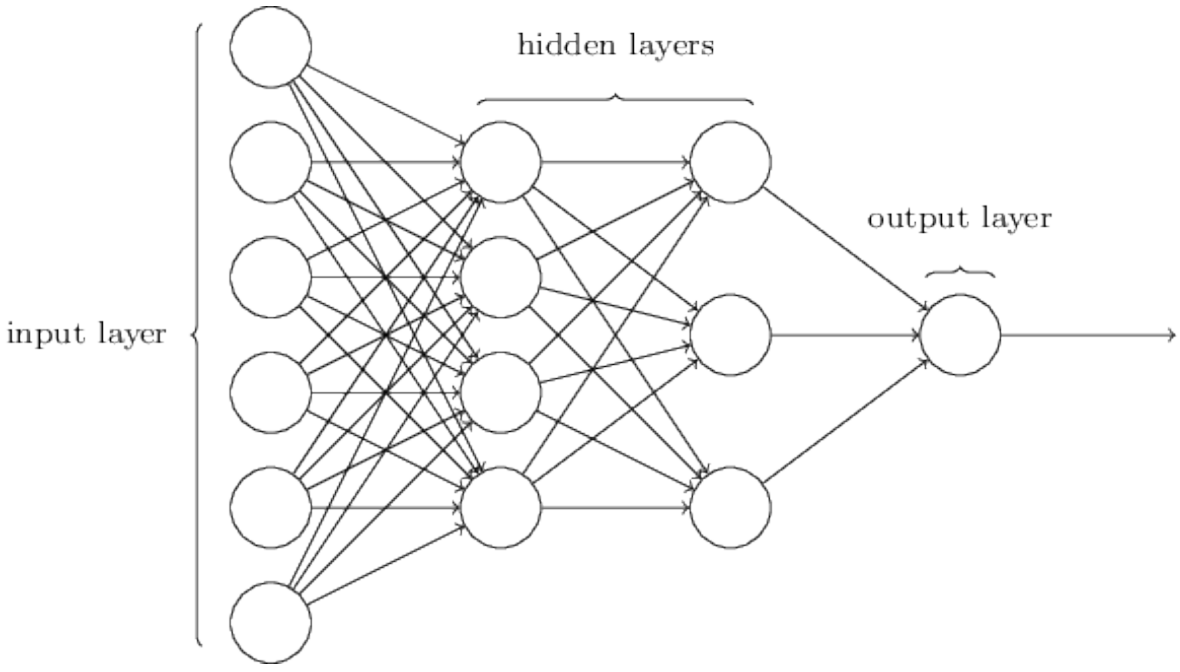
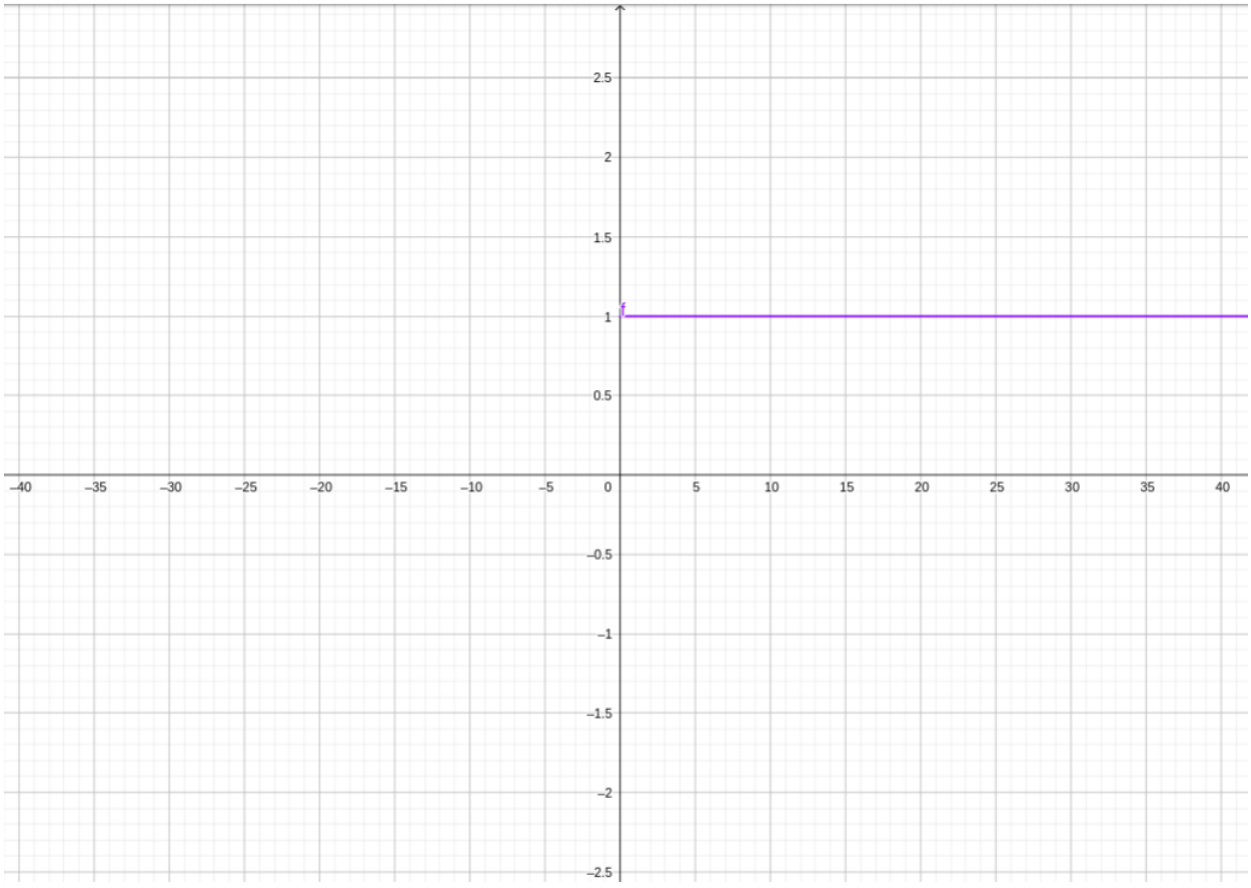


Figure 3.4: Simple Multilayer Perceptron
Source

3.4 Activation Functions

Neurons can be fired on not. Therefore, the tool that takes the responsibility of firing up each single neuron is called activation function, and its role is to derive the final output from the ones that were fed as input to the neural network. Moreover, they add the element of non-linearity, so the weighted nodes don't behave the same way resulting in a simple regression model. Subsequently, the most usual functions are:

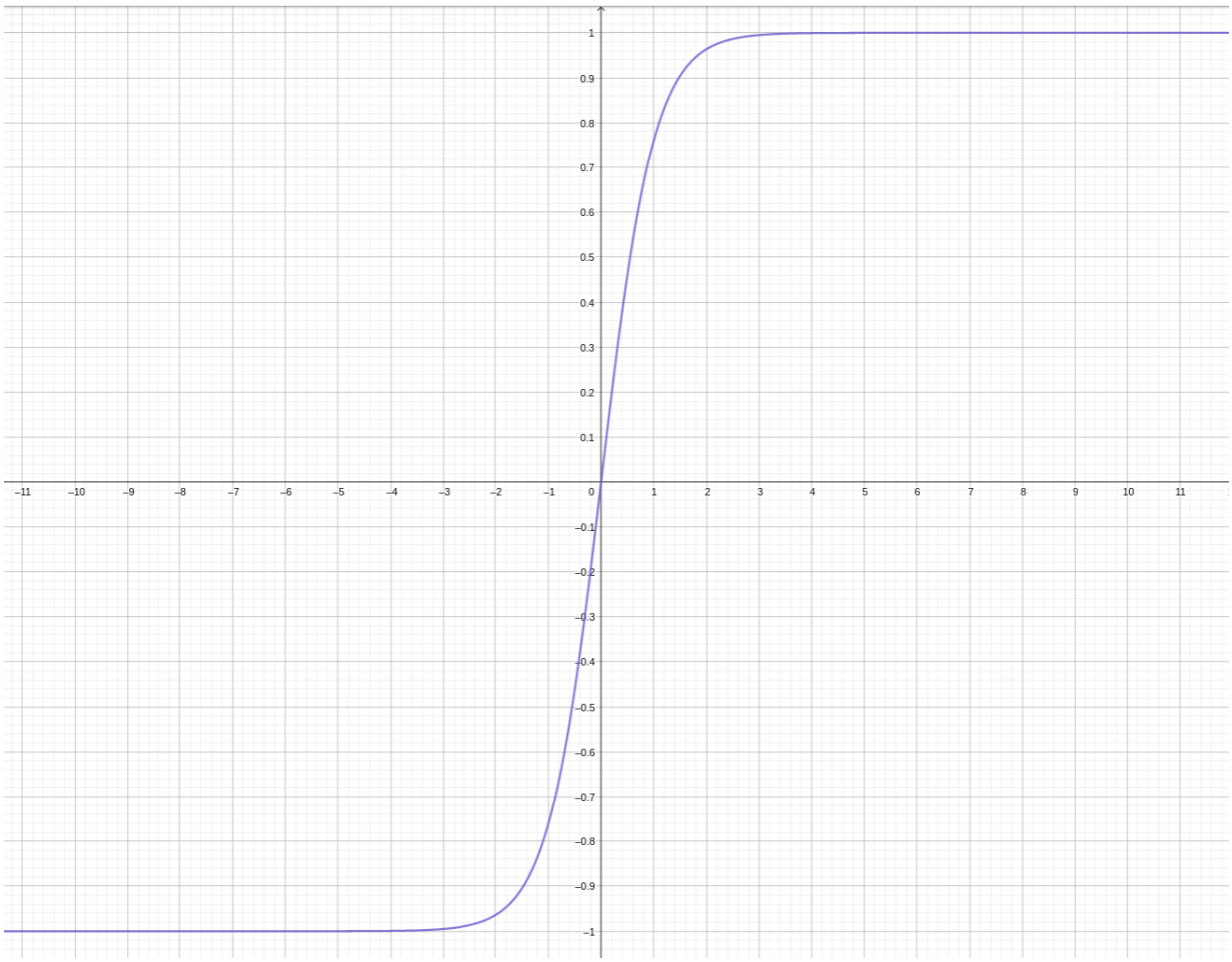


$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

Figure 3.5: Step Function

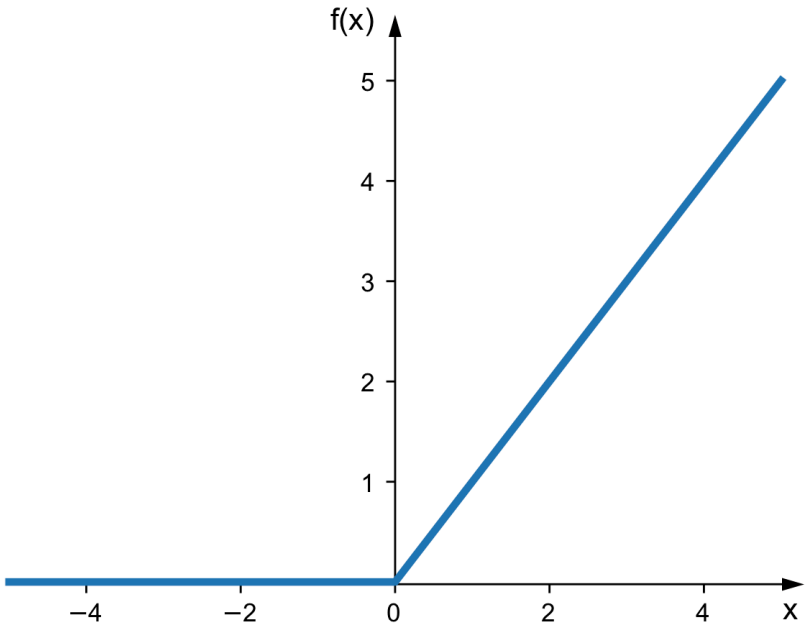
$$f(x) = \frac{1}{1 + e^{-x}}$$

Figure 3.6: Sigmoid Function



$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

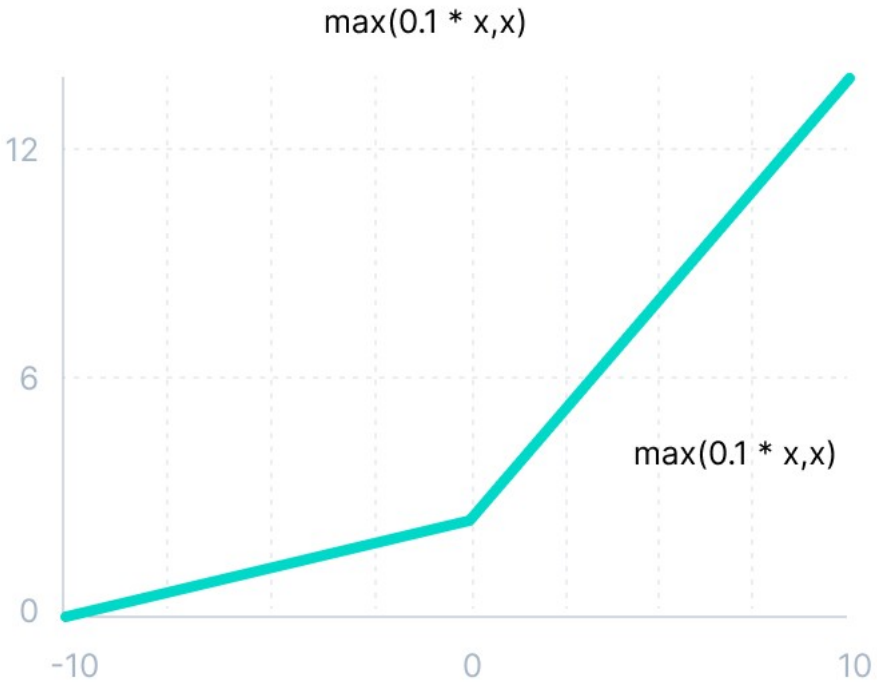
Figure 3.7: Tanh function



$$f(x) = max\,(0, x)$$

Figure 3.8: ReLu Function

Source



$$f(x) = max\,(0.1x, x)$$

Figure 3.9: Leaky ReLu function

Source

3.5 Training and Optimization

3.5.1 Backpropagation

Although feed-forward NNs follow a direct flow, it lacks a correct application of weight tuning via the measured error, and this is where the algorithm of backpropagation comes into play. Essentially, this method obtains the error rate of the previous training epoch in order to assume the next step for weight calculation, whereupon it manages to micromanage every tiny bit of the network nodes in order to reach the best approximation. All in all, the key term here is **backwards**, with the algorithm incorporating the usage of **gradient descent** (GD) in order to calculate the gradients of each layer in a bidirectional stream, with the employment of partial layer gradient calculations between each step. Hence, the gradient of the last layer is calculated first, and the one from the first is being calculated last.

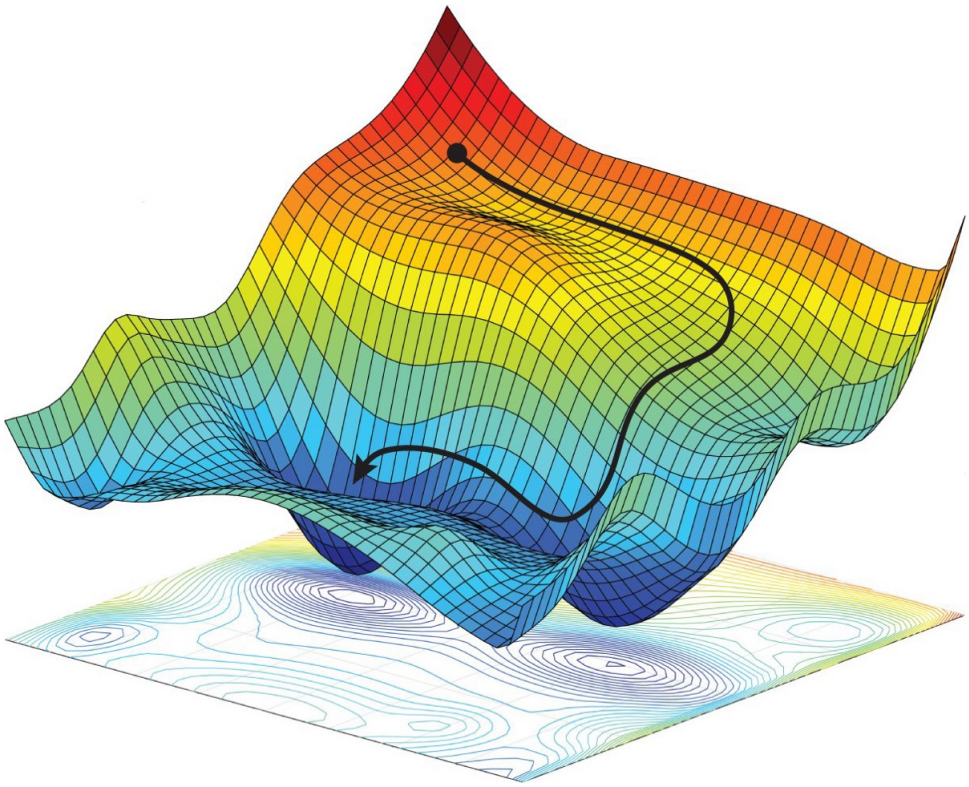


Figure 3.10: Function as a 3-D graph.
Source

3.5.2 Stochastic Gradient Descent

As figure 2.10 suggests, the clearest explanation of a gradient is the best formulated slope of a surface, with its descent being the lowest function point found, thus finding the optimal x where y is the minimum. Although GD is a great algorithm as a basic idea, it can manifest as a whopping procedure due to the fact that it needs to calculate the derivative for the corresponding function with respect to each single feature provided, while also taking care of a large number of sum of squared residuals, the largest the dataset is. So, *"stochastic"*, which stands for *"incidental"*, is an optimizing method which adds a unique variance to the algorithm by picking a single datapoint for each iteration instead of calculating the whole dataset. Consequentially, this does not only manage to omit the dependence on dataset size, but it makes the

descent take smaller steps which eventually add up as a more faster, more precise and careful mapping to the **global** minimum. As such, the equation can be formed as:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x, y) \tag{2.3}$$

1. where θ are the biases, weights and activations,
2. η is the learning rate
3. ∇ represents the gradient, and
4. J the Loss Function.

Interestingly, from step ***3*** and onward is described the algorithm of backpropagation.

3.5.3 Cost Functions

These functions are the deciders of what's the model's predictions offset from the original input and are being used in supervised algorithms. Generally, they calculate how *"further"* the model got from the input, with a greater distance meaning greater loss. Moreover, as the weights get tuned with each iteration, the cost function directly outputs the iteration's error so that the next one will apply a better fine-tuning to the weights, based on the desired goal that the given loss must be minimized and not stagnate. Two of the most commonly used functions are

Logarithmic Loss, commonly referred to as **Log Loss** or **Binary Cross Entropy** (BCE), which is used as a definite punishment for wrong **binary** classifications, meaning only 0 or 1:

$$BCE = \frac{-1}{N} \sum_{i=1}^{N} y_i \log p(yi) + (1 - y_i) \log(1 - p(y_i)) \tag{2.4}$$

, and

Categorical Cross Entropy, a modified version for multiple, mutually exclusive class problems:

$$CCE = \frac{-1}{N} \sum_{i=0}^{N} \sum_{i=0}^{J} y_i \log \hat{yj} + (1 - y_j) \log(1 - \hat{y_j}) \tag{2.5}$$

3.6 Autoencoders

An Autoencoder (AE) is a Neural Network which belongs to the unsupervised algorithms cluster and is comprised of a tripolar architecture; the Encoder, the Bottleneck Layer, and the Decoder which are typically placed in a symmetrical manner. AEs manage really well to handle noisy and multidimensional data by flushing out obsolete parts of it, and do so by mimicking the input as output, while subsequently producing a reduced representation in between. Generally, those networks try to present a direct reconstruction of the original input. Essentially, the idea is to pass in an input which might be particularly noisy,

feed it to the intermediate layer called bottleneck, or embedding, which is responsible for down-sampling the previous input into a simpler, reduced and cleared from noise representation, and then proceed to turn it back to its original form, thus connecting the dots as to what's the underlying non-linear relationship, without the need of any kind of labeling, since this is an unsupervised algorithm, as stated before.

Subsequently, AEs have demonstrated a remarkably good performance at extracting the concealed features of images, thus providing a sufficient solution to image related problems, with one being the solution of increasing the given resolution to an image system, namely super-resolution. This efficiency can be easily traced back to successful studies like [1], where the model managed to reap a significant amount of context related imaging information and eventually leading to results up to par with proven state-of-the-art methods.

Moreover, outlier detection is a common task which incorporates the entrance of errors and assortment into any dataset, thus enabling hindrance to a good generalization opportunity. [2] presented an important randomly connected AE approach to the anomaly detection problem using Cardiotocography data with fetal heart rate signal measurements. Outliers are particularly significant to medical applications since they might signify just an error, or a truly urgent case. The results accurately described the hidden points leading to the proposition that AEs can handle this essential task effectively.

Reversely, this neural network architecture manages to be robust enough to solve problems similar to this paper's project, which is time-series feature selection and clustering, as proven by study [3], where researchers completed a successful experiment on financial time-series data, which are particularly noisy and include obsolete or low significance features. Labels generated by the K-means algorithm were fed into an AE neural network that eventually extracted the hidden and most significant features, a function much necessary to this fMRI-related project since a large number of parameters, close to 5600, must be compressed and clustered to groups based on an importance percentage.

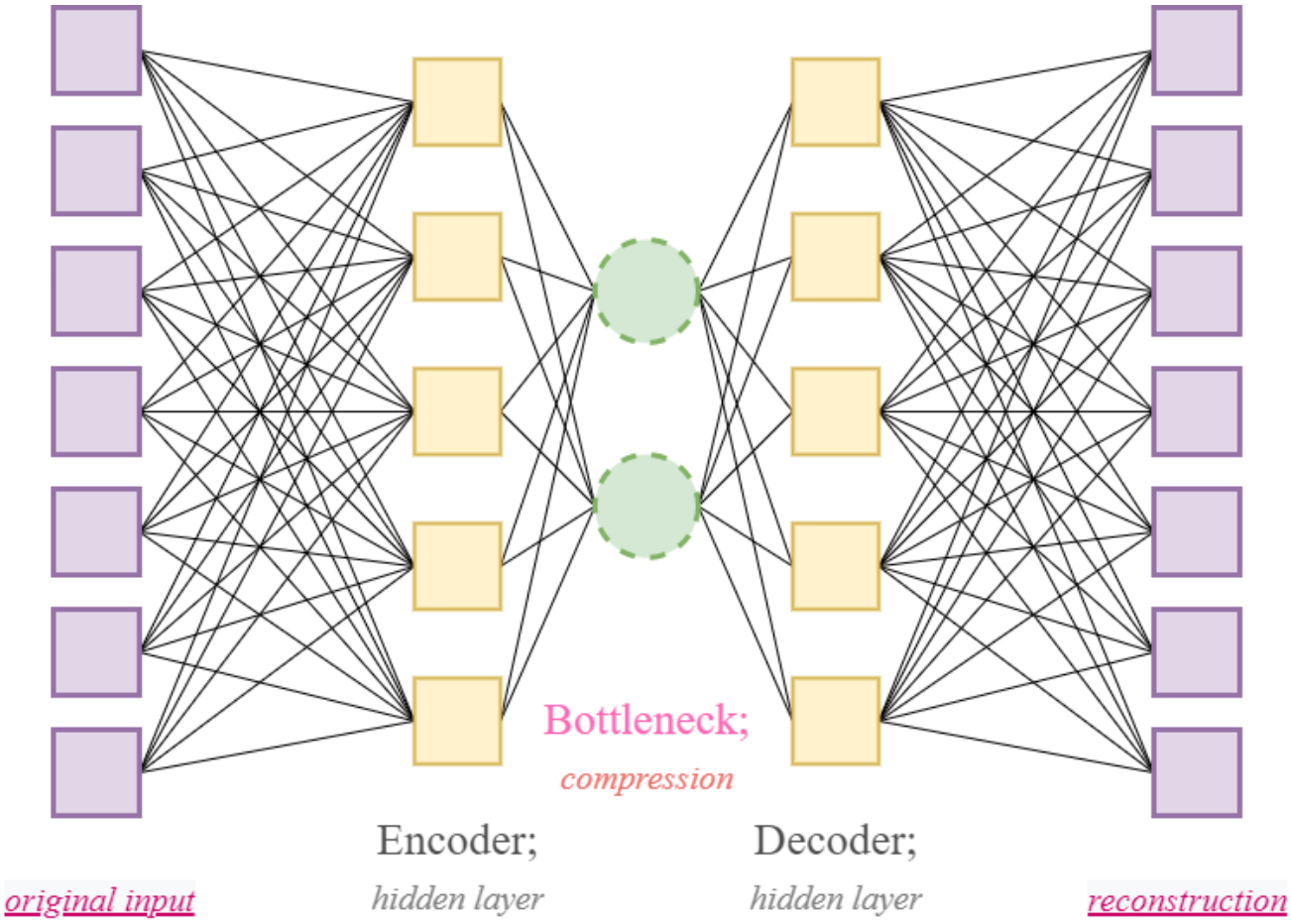


Figure 3.11: Full Autoencoder Architecture

3.6.1 Encoder

The encoder layer represents the entrance of the network, which always takes on the according input, and can be represented as a typical function type:

$$h = f(x) \tag{2.6}$$

describing a mapping of x to h, where x is the input, and h the extracted low dimension representation, with this layer typically including a symmetrical decrease of neural network connections, something which must be mirrored in an increasing manner to the decoder layer as shown in the image below, leading to the hidden middleman layer, the embedding. Note that g needs to definitely be of a lower dimension of the initial x input, otherwise it's called **undercomplete**, and aims at finding the most obvious patterns instead of the underlying ones.

Furthermore, when the first layer is applied alongside a type of regularization, the decreasing size of the nodes aids at the possible deduction of unique and novel patterns in the raw data. Moreover, since AEs can be considered a non-linear dimensionality reduction tool, quite like a kernel PCA, this downsampling pattern prepares the data for the bottleneck by starting the production of lower dimension reproductions of the initial data, a trait which is the epitome of the AE concept altogether.

However, it's important to note that AEs can be separated into distinguished variations based on the input's peculiarities. For example, Denoising AEs can usually handle input differently than Contractive AEs by letting in a certain amount of outliers, while the latter completely ignores any type of input aberration. A particularly good description of the encoder, especially at image problems, could be 'the challenger' since its role is to actually feed the hidden network with a strained version of the input with the requirement to reach the original state. Thus, the general architecture of AEs incorporates the idea of indissolubly conjoined parts, connected, yet each one taking upon a separate task of the down-sampling -> lowest representation -> reconstruction scenario.

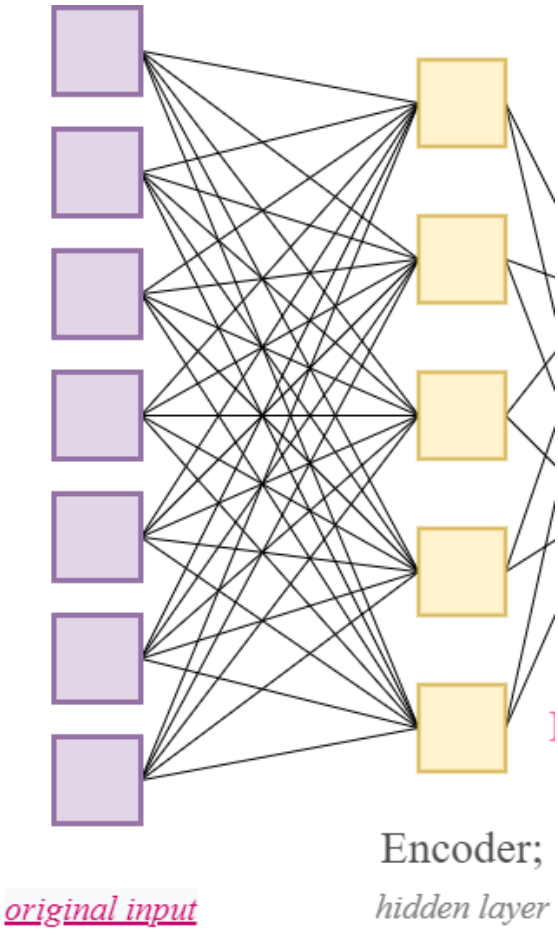


Figure 3.12: Input, leading to the encoder.

3.6.2 Bottleneck Layer

The bottleneck, alternatively called embedding or code is the final outcome of the encoder and usually is of the smallest size as a subset of the wholeness of the network, and can be comprised of one or many layers, forming a multi-layer AE. It's size is modeled as a hyperparameter, fixed and set beforehand, thus predestining the type an depth of the AE in accordance to each use case. Subsequently, it serves as the final input to the decoder layer. Moreover, as a general rule, although the encoder's and decoder's sizes might not always be of the same caliber, thus breaking the mirror pattern, the bottleneck needs to be of a lesser size between both of them, since the compression is by default of a lower dimension. Interestingly enough, when more than one, code layers offer the opportunity to create a plethora of representation

choices by letting the engineer decide which compression fits the most.

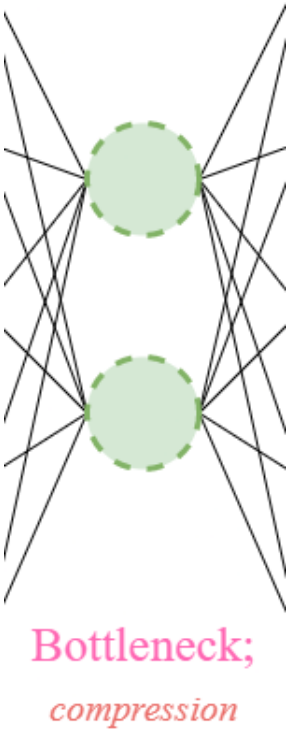


Figure 3.13: Bottleneck, containing the latent representation.

3.6.3 Decoder

The Decoder is the last part of an AE and can be represented in accordance to the Encoder as:

$$x' = g(h) \tag{2.7}$$

which can can be explained as the second part of (1), where *x'* represents the reconstruction of the initial input via the compressed one coming straight from the bottleneck. Furthermore, the Decoder's performance must be assessed under the scope of how closely the produced reconstruction reaches the original input, a task which can use either Cross Entropy or Mean Squared Error, and can be stated as the following expression:

$$L(x, g(f(x)) \tag{2.8}$$

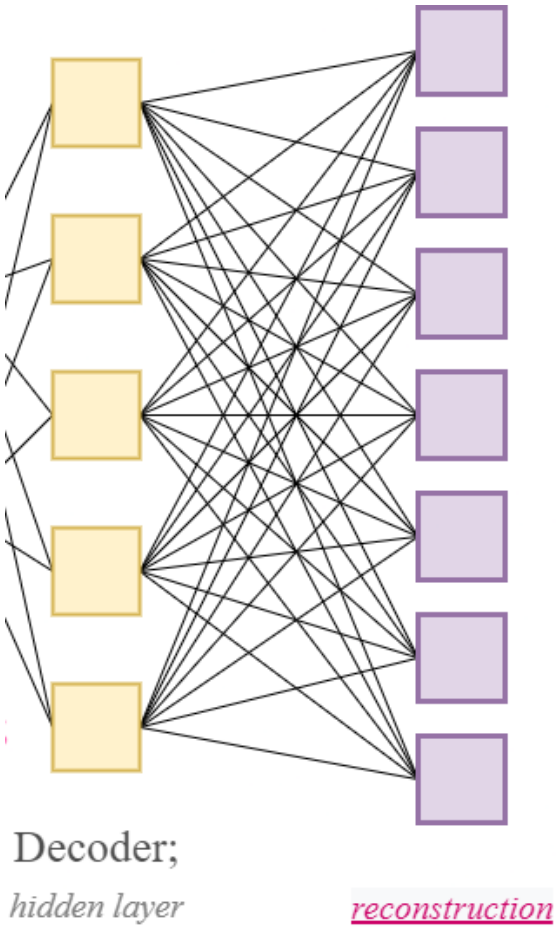


Figure 3.14: Decoder part, aiming to reconstruct the initial data.

3.7 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) is an algorithm with a long history dating back to the 80's, not meeting their true potential up until recent years, since the distribution of corresponding technology which permits enough computational power has massively improved since their birth time. One of the fundamental principles of RNNs is the idea of past inclusion to the present, meaning that they take into consideration significant parts of the input, "remember", in order to map a delicate and precise approach to the incoming data. Consequently, these networks are widely used for time series data, like speech and signal processing, or financial data. Thus, the most particular and successful description of this method is stated as **the ability to extract the underlying connection between incoming and outgoing sequences of the provided data** [4].

Hereupon, the mathematical expression of an RNN activation is as presented below:

$$h_{t+1} = f(x_t, h_t, w_x, w_h, b_h) = f(w_x x_t + w_h h_t + b_h) \tag{2.9}$$

, where **h** denotes the hidden values at time t (or $t + 1$, accordingly), **x** the input at time t , **w** the corresponding weights, **b** the defined bias.

As the 2.3 figure suggests, RNNs are fundamentally different from Feed-Forward Networks since they

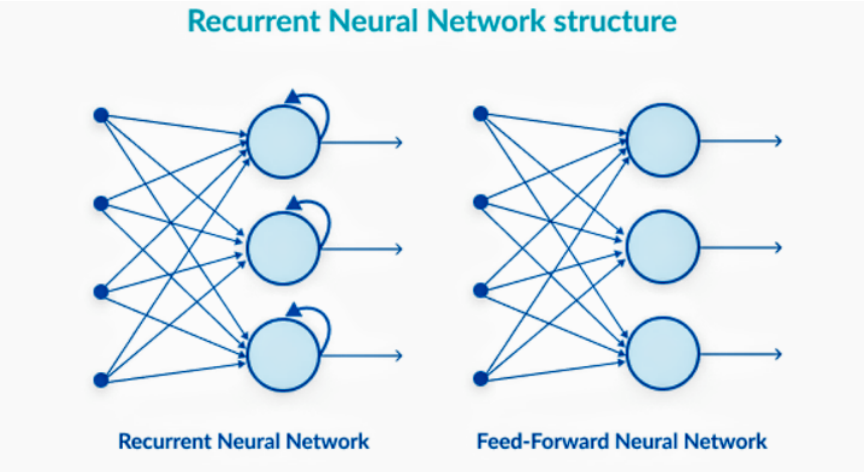


Figure 3.15: Distinguishing Feature Between RNNs FFNNs
Source

tend to loop bits of input information, creating a sequence of historical connection between predisposition and prediction, with the latter being in immediate association with the former. Essentially, the power of RNNs lies on the ability to also map differentiating quantities of input to output, which include:

RNN Applications
One to One (most, usual application attributing one yield to a single input)
One to Many (mapping singular contributions to multiple outputs, like a single chord forming a whole song)
Many to One (reversely, demonstrating the compression of a multiple data contribution)
Many to Many (a concept alluding multiple inputs to according outputs, just like a translation)

3.7.1 Long Short Term Memory Networks (LSTMs)

Although RNNs are a remarkable way of drawing information after incorporating the element of ”time”, their approach ceases to expand into a larger time-frame, with their spectrum being limited to closely associated events, rather than distantly aroused instances. Correspondingly, an demonstrating example of this important difference is the placement of words inside a statement. Provided the ”Greece is a country of the South, below Germany, and Above Egypt, with significant amount of sunlight during an extended period of time”, RNN(s) might successfully follow the narrative of ”**Greece**” is a ”**country**”, located in the ”**South**”, yet the amount of ”**sunlight**” might be outside of their capabilities, since they lack Long-Term memory association. This is were Long-Short-Term-Memory Networks come at play, introducing

the ability to both expanding the topological array of knowledge connection, while simultaneously being able to forget unnecessary parts of data, by introducing the Forget Gate.

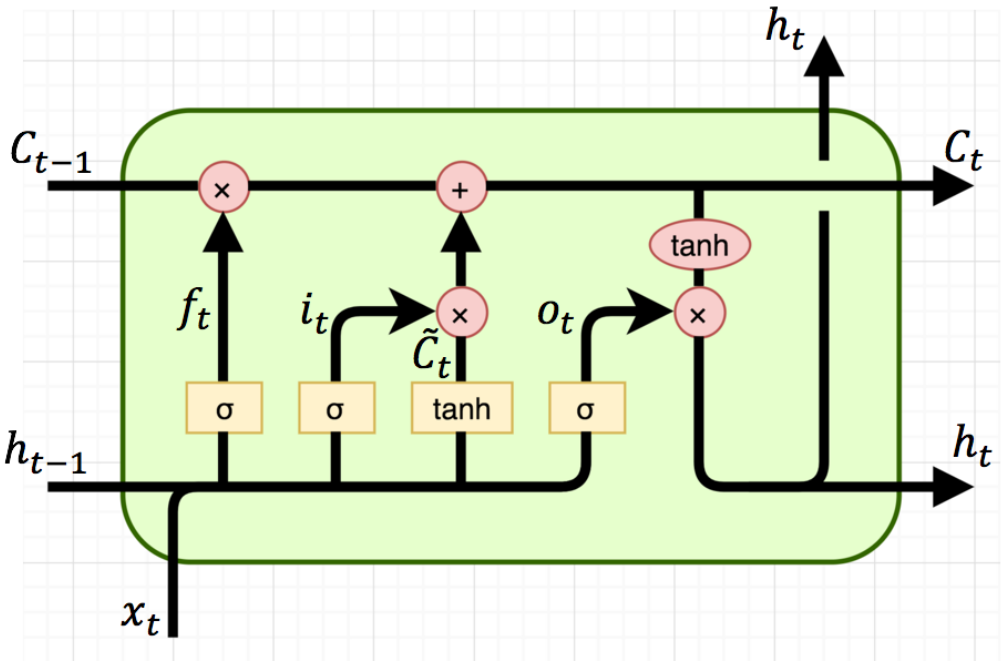


Figure 3.16: LSTM Fundamentals
Source

The above figure precises the most fundamental LSTM fabrication, and can be explained in six simple, interconnected expressions:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.10}$$

, illustrating the initial state which decides what information will be thrown away, where *h* is the previous input, *x* the current, *b* the bias, and σ the sigmoid layer, adjudicating which values will be updated.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.11}$$

, presenting the next step of actually "**forgetting**" the unnecessary information, essentially creating a fresh update.

$$C_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.12}$$

, which incorporates the *tanh* function in order to create a vector of the new eligible values.

$$C_t = f_t * C_{t-1} + i_t * C_t \tag{2.13}$$

, here, the update is taking place therefore C_{t-1} becomes the new, renewed C_t, while also multiplying the old state by f_t, applying the repudiation, and adding $i_t * C_t$, signifying the scale of the update on each value.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.14}$$

, demonstrating the filtered result of the output, which passes through a sigmoid function finalizing the overpass parts, and lastly

$$h_t = o_t * tanh(C_t) \tag{2.15}$$

, using *tanh* again in order to produce values of range [-1,1], multiplying by the filtered output.

3.8 Hybrid Modeling

One interesting take on the architectures of Neural Networks is that although there's already a plethora of one-dimensional blueprints which are usually used for a certain task, like Convolutional Neural Networks (CNNs) for image data, there's also the opportunity to create variations and differing combinations based on those initial blueprints, in order to expand, blend, amplify the model's capabilities. One such example is at [5], where researchers used a CNNs to form a 3D version of an Autoencoder in order to differentiate MRI images between users of psychoactive substances and the control group. Additionally, one other example is presented at [6] where a conflation of LSTM and CNNs was used in order to provide a sufficient solution to cryptocurrency time series forecasting. Subsequently, hybrid models have proven to be an effective strategy for fMRI data procedures too [7].

3.8.1 Autoencoder-LSTM

Forming an LSTM-based Autoencoder requires the usual mirrored architecture, although the units themselves are LSTM ones, presenting the potential to manage, compress, and finally reconstruct sequential data. One key aspect of this method is the usage of two distinct layers provided by *Keras*:

RepeatVector, which is located on the bottleneck of the pipeline, and its key function is to take the last layer's resulted output as input and multiply by *n* times, providing the final compressed version of the corresponding sequential data.

TimeDistributed, an instrumental layer located at the last part of the decoder, accountable for applying the contextually applied layer to all the sequential time-steps autonomously, forming the reconstruction of the initial input, thus that of shape *(sample, timestep, feature)*.

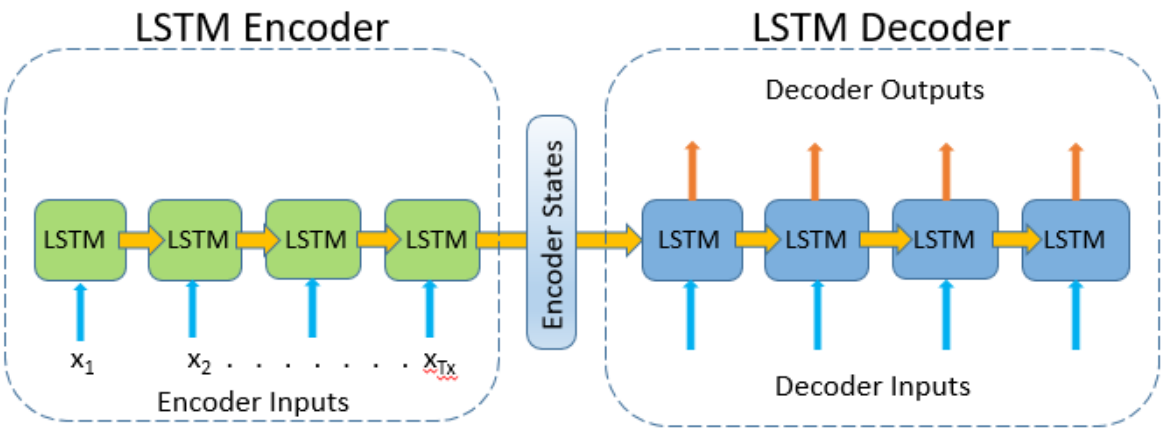


Figure 3.17: AutoEncoder-LSTM Steps
Source

3.9 Decision Trees

Simply put, a Decision Tree (DT) is a specialised flowchart which starts from a basic proposition and leads to a final decision, thus, the classification itself. Generally, the tree starts from its root, also called *node*, and then follows a branch splitting based on simple *if* clauses, with each split representing a different decision, eventually forming a tree-like shape, justifying its title.

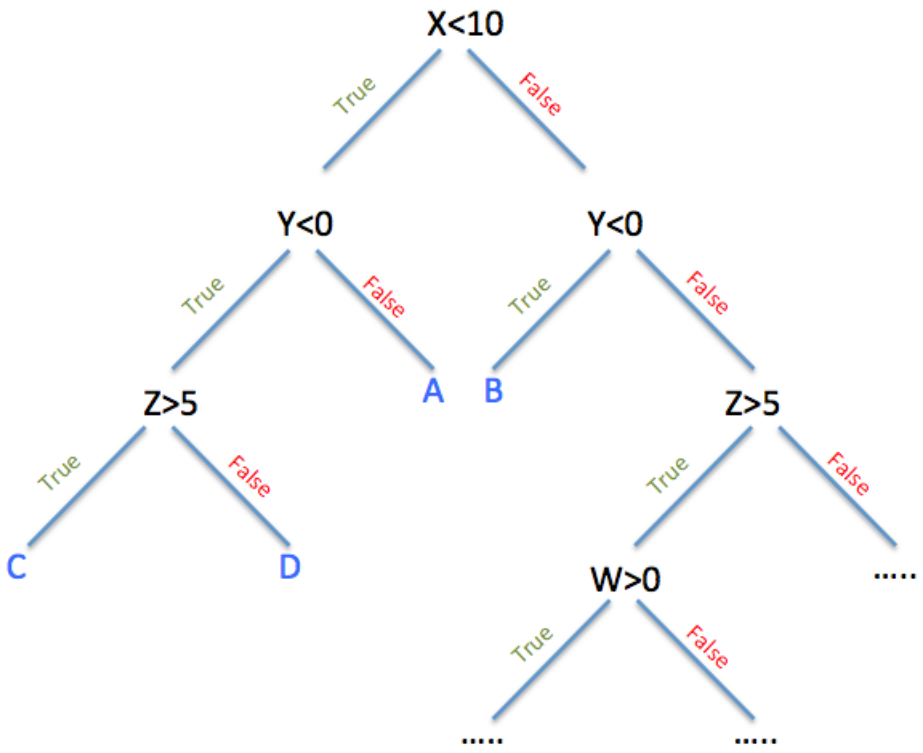


Figure 3.18: Decision Tree Structure
Source

As a matter of fact, DTs are separated in three distinct sectors:

- **Decision Nodes**, showing a decision,
- **Chance Nodes**, representing the probability and uncertainty,
- **End Nodes**, demonstrating the final outcome.

Those nodes can be used sequentially to form the branches which in turn can be used multiple times so that a more complex tree is being constructed. Consequently, as figure 3.1 demonstrates above, the **root** node starts with the initial clause *x<10*, while the blue outputs of *A, B, C and D* are the **leaf** nodes, representing the outcomes. Subsequently, the tree contains the **internal** nodes, which might be **chance** or **decision** ones and form the basis of the architecture of each corresponding tree. One important note is that one can easily distinguish the internal nodes since they are located between another set of nodes,

manifesting a double connection. Moreover, figure 3.1 clearly presents the **splitting** procedure, where a variety of caveats are being applied onto the features (X, Y, Z, W, etc...), creating the effect of **branching**. Lastly, an important aspect of DTs is the **pruning** effect, a method similar to Neural Network Dropout, where branches are being cutoff when the complexity of the tree is being tremendously amplified, leading to possible overfitting and lower accuracy. Correspondingly, a mathematical formula for expressing a DT is being presented as the function below:

$$T : x \rightarrow y \tag{3.1}$$

T expresses the Tree, x is a feature set, and y the continuous value, or a class outcome. Subsequently, the approach of this algorithm follows the stream of consciousness where finding the best *T* such for *(x, f(x)*, *T(x)* comes as much closer to *f(x)* as it can manage to, with the respective loss expressed as:

$$loss(T(x), f(x)) \tag{3.2}$$

3.9.1 Gradient Boosting

Although one model might be able to produce satisfying results, scientists and engineers use a sum of multiple models. Correspondingly, gradient boosting (GB) is this exact method of using a plethora of *weak* learners in order to build a single, strong one. Also, this type of modeling is usually called *"ensemble"*. Therefore, *boosting* refers to the aggregated nature of multiple models, while *gradient* covers the aspect that we have two or more derivatives of the **same** function.

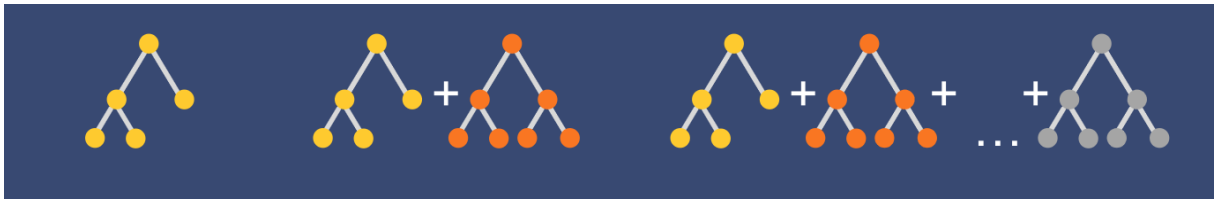


Figure 3.19: Gradient Boosting Blueprint
Source

Correspondingly, it's important to note three main components of gradient boosting:

Weak Learner, which is the learner that produces poor results, therefore being unable to stand up on their own. Interestingly enough, their approach could not be that further than a simple random guess. Usually, decision trees are the most usual algorithm used as gradient boosting learners.

Additive Model, describing the procedure of stacking the (weak) learners. Each step should produce a positive result in order to further minimize the according loss function.

Loss Function, being the usual method for estimating the aberration between the original datapoints and the model's results, and is configured in accordance with the problem at hand (regression, classification).

Chapter 4: Task and Dataset

4.1 Task

Blood Oxygenation Level Dependent (BOLD) signals represent the observations of fMRI scans and pose a particularly difficult type of signal to interpret mainly due to the fact that their nature consists of multiple layers of neuron connections which might or might be not activated. Therefore, source and meaning interpretation behind a single one or a combination of many requires a delicate yet efficient, commonly multi-layered approach [8].

Essentially, one type of such interpretation is the reconstruction of the whole or a missing part of the signal, which goal is the exploration of the brain's underlying connections and why such connections are activated for each corresponding stimulus, leading to a more deep understanding of the grand structure, thus aiding medical, sociological and psychological research (e.g. Alzheimer's prevention, Depression prolepsis, Cognitive Impairment) [9] [10].

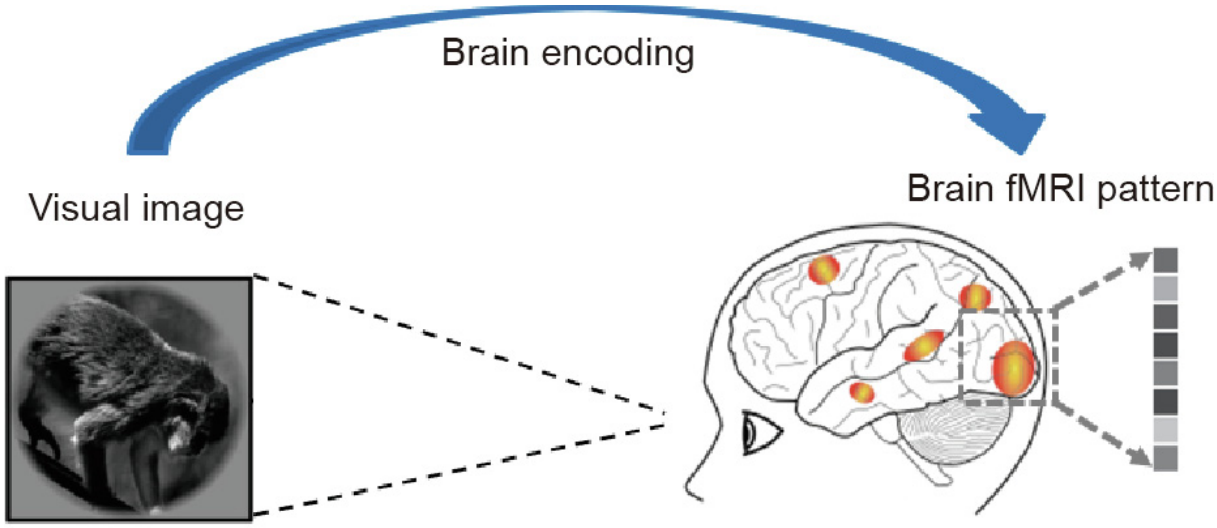


Figure 4.1: Brain Signal Encoding.
Source

By contrast, the other type of interpretation aims at the stimulus's decoding based on the brain's signal itself. This procedure gives less focus to the technical architecture of the signal while simultaneously trying to extract an intention-based comprehension of the underlying meaning and the person's motive. Consequently, this paper examines the latter type by solving a stimuli classification problem using the BOLD signal (voxel activation) as the features.

Therefore, fMRI procedures can be examined through a double, bidirectional set of problems, with the first one being the **Encoding**, where the scientist or engineer has to reconstruct and predict the **brain's assumed activity** [11], starting from the stimuli itself, while this paper conducts experiments on the second part of the set, the **Decoding**, just like at [12], where the same Bold5000 [13] dataset was used for this exact reverse approach, which aims at the stimulus's prediction based on the brain's localized

mapped signal [14] [15].

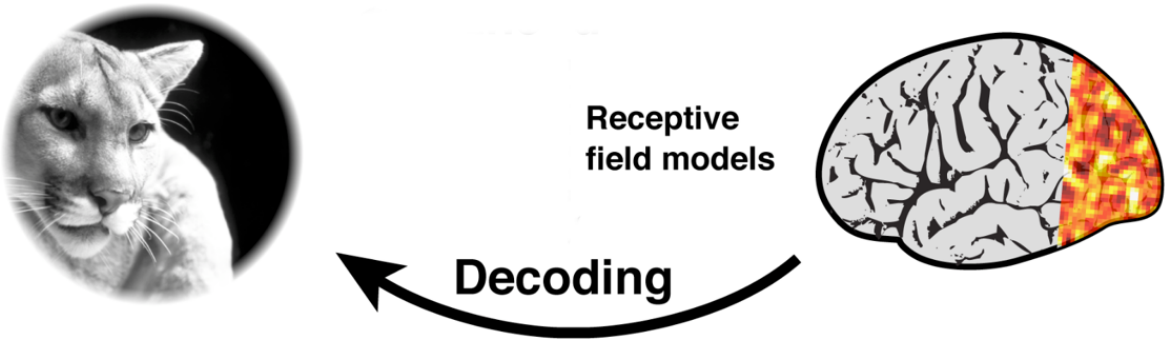


Figure 4.2: Brain Signal Decoding.

One important unifying aspect of the aforementioned approaches is that although they tend to follow differing directions, they usually accommodate the requirements of neuroscientists who need an algorithmic learning approach which must cover a large spectrum of brain's matter, with the former approach being a more visual one, creating a conjecture of how the brain might respond, thus a truly envisioning angle and a test of our abilities of brain understanding, and the latter being the translator of temporal and hidden information, hence, producer of direct interpretations of how the brain chronicles and responds, reconstructing (estimating) the **stimuli**.

4.2 Dataset

These type of datasets often have a special trait of containing a plethora of terms based on multiple regions of interest (ROIs) that must be properly comprehended before they are fed to any kind of model, and that's due to the fact of the presence of interdependently connected data inside the brain signals posing the question of how those are eventually produced [16] [17]. Moreover, a widely used, dedicated neural network architecture for pre-processing and segmentation on such biomedical data is that of the U-Net [18] [19]. Therefore, below is presented a compact explanation for the most significant ones before I proceed to the methodology analysis.

fMRI stands for functional Magnetic Resonance Imaging. Its usage lies on the observation of matching changes of patterns in the brain signal. Consequently, it alternates between states of task and control.

Localizers. In general, each brain has a unique anatomy, much like the fingerprint, therefore, this paradigm doesn't offer the luxury of generating a result via multiple prototypes like the usual classification problems. On the contrary, one has to extract a meaning while examining one brain at a time due to the fact that its activation locations will be absolutely personalised, thus the sensitivity locations will differ for each person. Hence, a localizer is described as the **block design experiment** which aims to map those activation paths.

TR, which stands for Repetition Time, and represents the data read from the same brain location. between discreet time frames (points) and is demonstrated as the length of time between each coupled observation.

Trial, meaning a single stimulus display.

Block, hence a sequence of trials of the same stimulus category.

Run, which represents a sequence of blocks displayed to a subject, and lastly

Label, an already known term in the machine learning spectrum which showcases what was the subject observing.

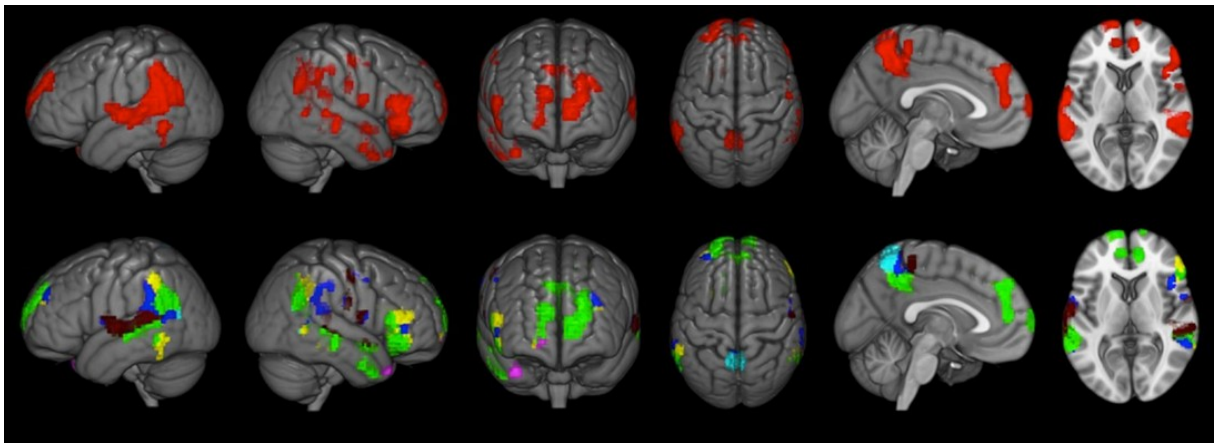


Figure 4.3: Localizers giving prominence to brain's topological actuation.
Source

One of the oldest fMRI datasets is Haxby (2001) [20], which is comprised of 4 (four) subjects who follow the same procedure, hence, direct stimulus presentation with alternating resting states. Although its's brain maps contain a signifiantly less amount of voxel numbers, the subjects were presented with a total amount of 8 (nine) labels: **bottles, cats, chairs, controls, faces, houses, scissors, and shoes**, presenting a large obstacle since the inverse proportion of voxels to labels is directly involved to a curbed generalizing potential.

4.2.1 BOLD5000

BOLD5000 is comprised of scans of 4 (four) subjects who where presented with direct stimuli display from 3 (three) distinct image databases, namely COCO, Imagenet and SUN, totaling 5254 images per subject (CSI1, CSI2, CSI3), 3108 for CSI4, with the demonstrations spanning 15 sessions each. Accordingly, the image distribution from each database is that of 2000, 1916 and 1338 respectively. Moreover, 4916 distinct images were demonstrated once, with 113 being repeated 3 times. Lastly, trials are comprised of 10 seconds each, with a TR of 2 seconds.

This particular dataset offers a unique approach to the experiments and it's a good proponent of acquiring a generic solution to **mass** brain signal decoding, meaning the usage of **multiple subjects** as one large congeries, which can be efficient enough no matter the amount of voxels, or runs between the subjects,

while simultaneously trying to bypass the singular nature of each brain; a trait directly adopted by the nature of fingerprints. Consequently, the masked data used for this paper is shaped as time-series tables, following a line from start to finish, including both stimuli and resting phases, as presented at the figure below.

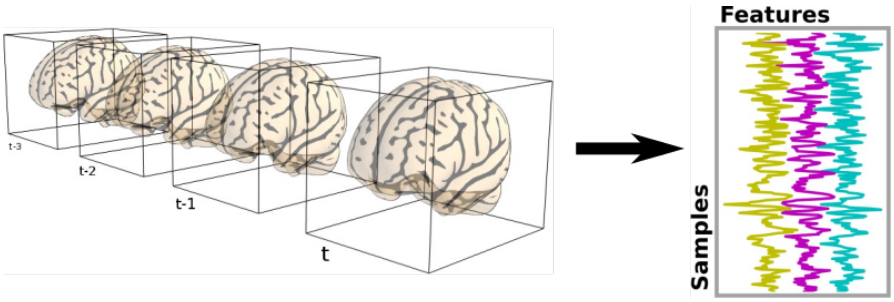


Figure 4.4: Time Induced Brain Masks & Voxels as Time-Series
Source

Finally, the image below presents an insightful view inside the average reactions of a subject's brain during the the stimuli demonstration, showing the slow pace of both the start and the "exhalation" of the activity. The activity for this particular moment is that of 1 single run inside a session.

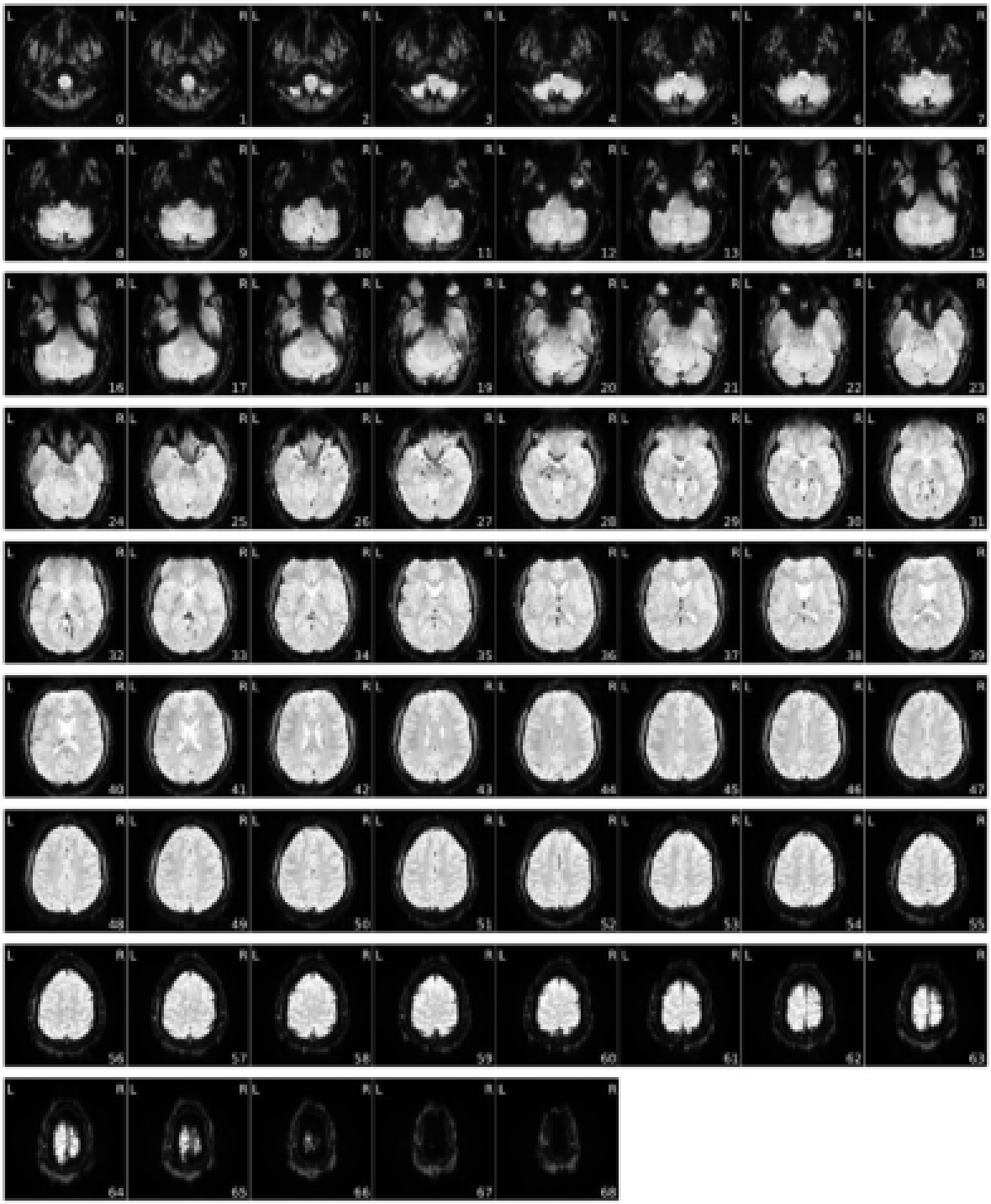


Figure 4.5: Progress of stimuli-brain reaction coupling.
Source

Chapter 5: Preprocessing, Impediments & Methodologies

5.0.1 Data Preprocessing

As stated before, BOLD data comes in a complex form and should be pre-processed in order to get into a matrix one, thereby being eligible for training and In general, there are 3 procedures that fMRI data must undergo: **corrections, transformations and alignments.**.

I used the same formation as [12], hence for COCO images, the category IDs were used to map the exact super-category, same as ImageNet, where each image was shown twice, therefore the mix from those two databases resulting in an overlap of animal and artifact images since they contained both, while all the SUN images where placed under the definite class of scenes. Also, it's important to note that all 10 voxel visual regions that where provided by BOLD5000 were used, namely LHPPA, RHPPA, LHRSC, RHRSC, LHOPA, RHOPA, LHEarlyVis, RHEarlyVis, LHLOC and RHLOC, containing both high and low-level visual regions.

As the image below illustrates, there is a specific ROI reaction for each of the 5 TRs which demonstrate a larger spike in general, and even though certain visual regions could be more useful and contain greater information for the immediate stimuli that each subject was viewing each particular time, I chose to conduct my establishment in the same way, so all regions were subsequently used.

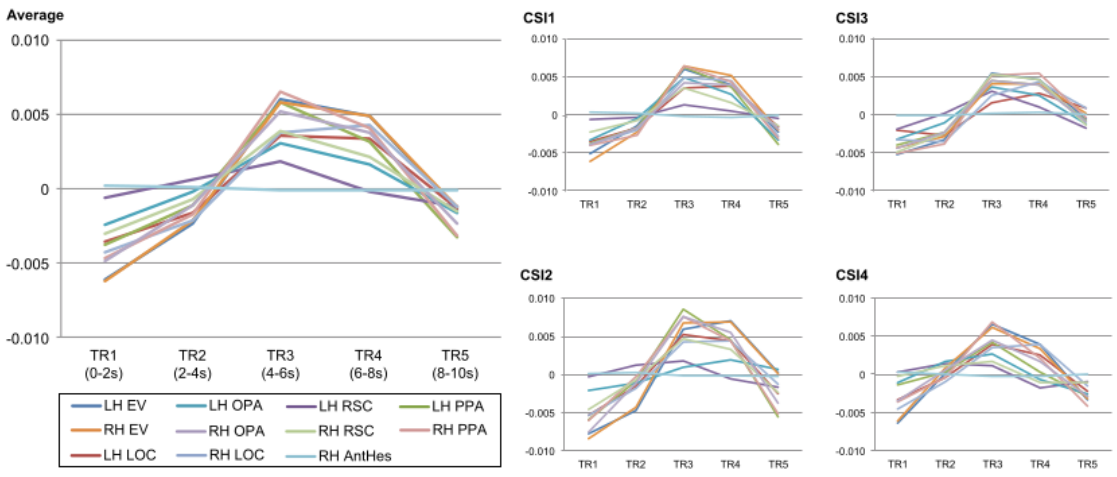


Figure 5.1: Time Induced Brain Masks & Voxels as Time-Series
Source

The images below demonstrate a very clear distinction between the three datasets and the corresponding classes that are being used, namely **animal, artifact and scenes**. All the same, this distinction is not always this straightforward, since there could be an overlap between the classes, resulting in a distinguishable brain activation mapping which might skew the model's understanding of the underlying connections between voxel activation - class.

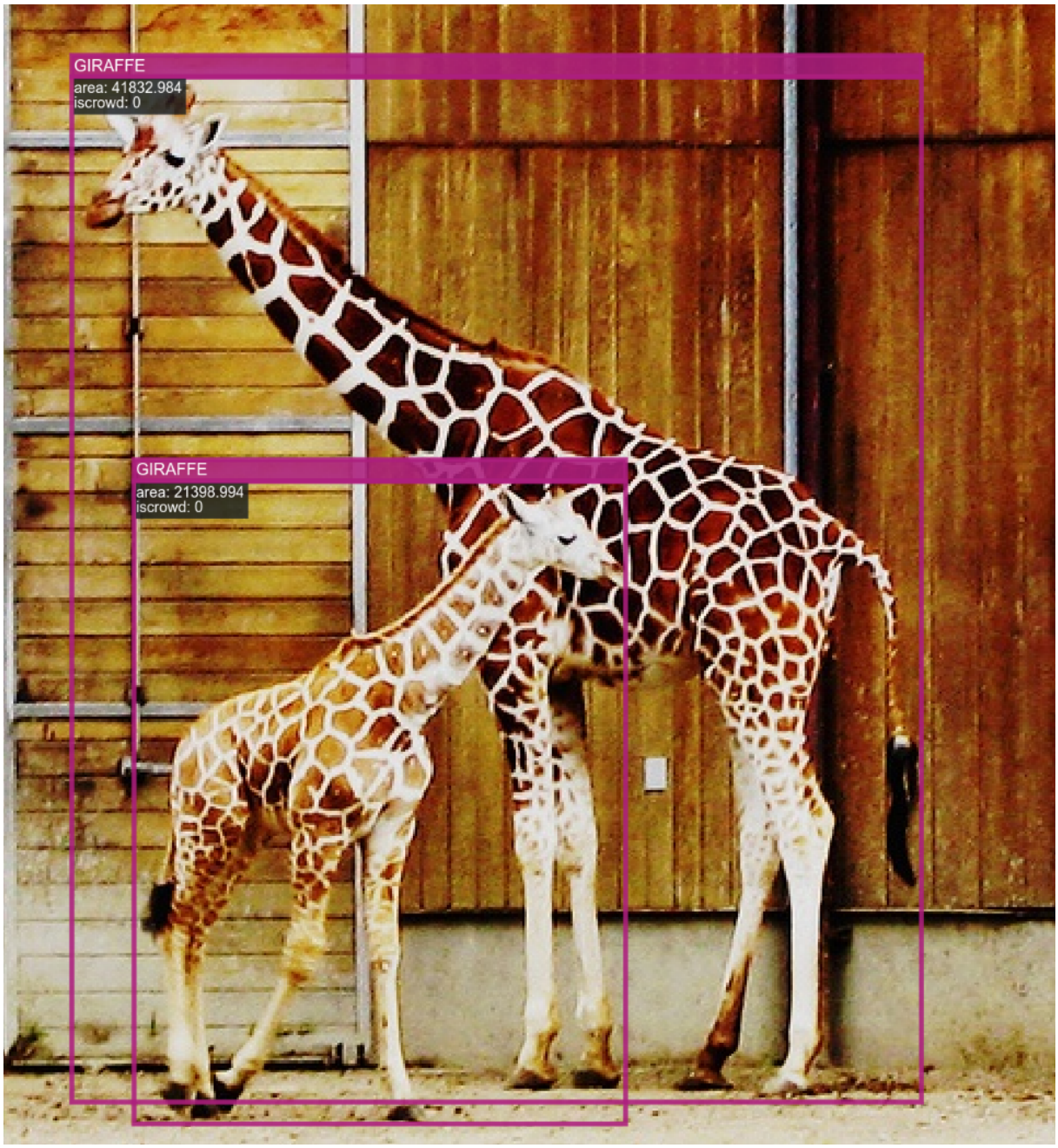


Figure 5.2: Animal image from COCO dataset. Dedicated animal images tend to avoid including many differentiating factors and elements that might produce further brain reaction.

Figure 5.3: Artifact image from ImageNet dataset. Artifacts should be demonstrated as single posing elements of artificial constructions that don't incorporate any kind of deep scenery, with the context being as bland as possible and the camera focusing on the artifact itself.

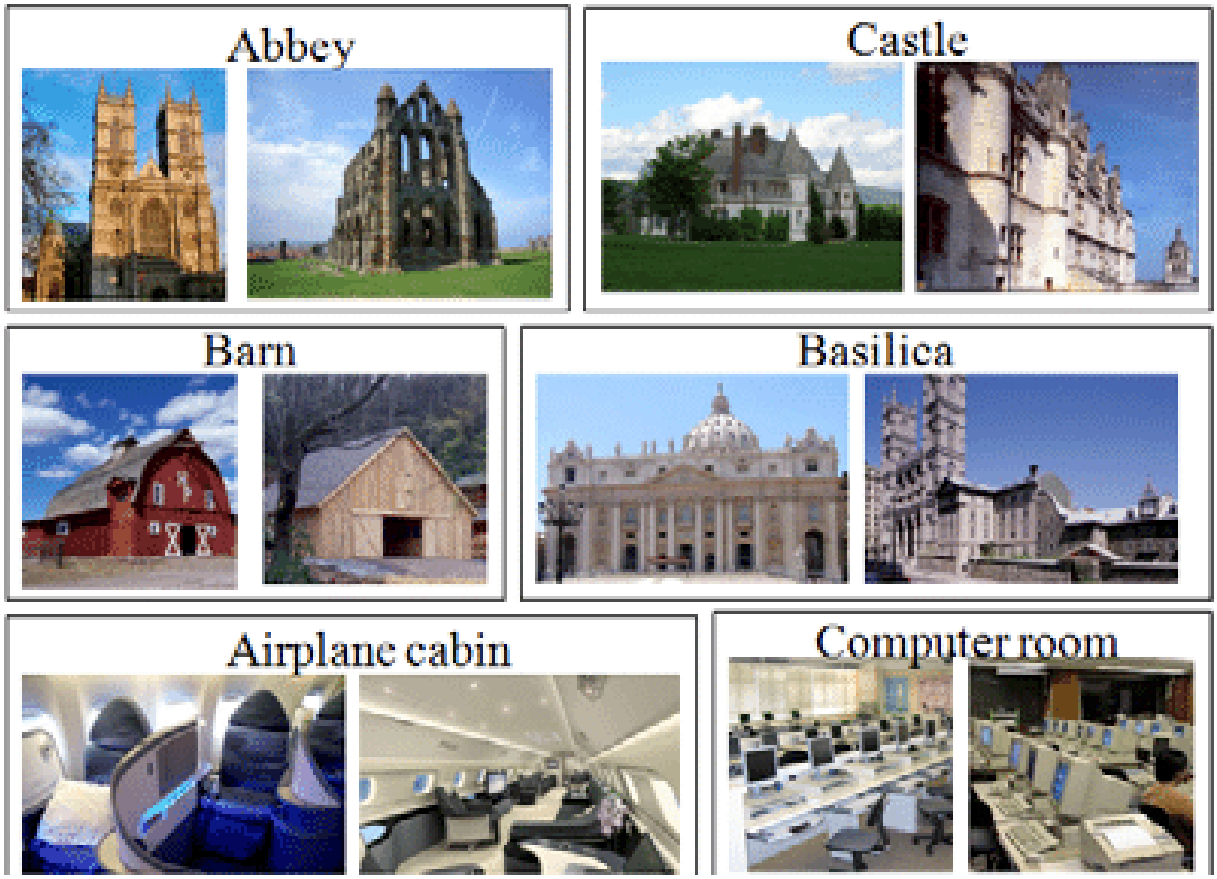


Figure 5.4: Scene image from SUN dataset. In juxtaposition, scenery images should include a plethora of colors, elements, buildings, deep context, shapes and non-focused artifacts, this providing the subject with the opportunity to view an open space with many interconnected pieces of furniture or other stuff.

Figure 5.5: Image from the ImageNet dataset, presenting a blurred class distinction. Inversely, this image presents the exact problem of indiscriminately distinguished classes, since inside of it exists a fence, meaning an artifact, grasses and plants, and even a scene in the background, so both of our classes of interest can be contained in this specific image, so each person's brain might react differently.

5.0.2 Predefined Problems

The basic premise was ta start by using the masks and the visual localizer files in order to work on the image files themselves, with the probability of using transfer learning in order to extract visual information representations [21] [22]. Yet, one reason which quickly led to the abandonment of that idea was that BOLD5000 already provides us with the extracted ROIs time series. Furthermore, the dataset itself is of a whopping 160GB size , which might further lead to greater problems needing a further preprocessing like increase in resolution, a technique thoroughly discussed at [23], a dramatic accretion at cost and time. In addition, due to the very nature of the dataset, it was important to test proposed models in a contrasting manner to [12], in order to test and examine based on the same principles and parameters, so the time series where eventually used.

5.0.3 Emerged Problems

Noisy data require a thorough compression, yet it's not always that easy to reach a good balance in order to not over-compress and compromise important data information. As such, the most usual problem I consecutively faced was the pre-defined size of the encoder itself, while simultaneously what's the best size for the pre-processed matrix in order to feed into the hybrid LSTM Autoencoder (AutoLSTM). Namely, trial and error proved that the AutoLSTM needed a direct connection between its predecessor

and itself.

Moreover, there is a unique idea of turning the ROIs into scalograms, as demonstrated at [24], [25] and [26], which incorporates the usage of visual models like Keras's applications, U-net, or plain Convolutional Network architectures [27], yet my experiments didn't manage to produce sufficient results, possibly due to the low information in the extracted images. Scalogram (Scaleogram) represent a visual form of wavelet series transform and are widely used for time series and bio-signals classification. Generally, they present the ability to capture sufficiently both the frequency and time domains, hence explaining the exact time of which each frequency made its presence. As shown below, the classes contain certain parts that are easily distinguishable even to the naked eye, therefore it's important to note that this direction is a quite interesting one and opens doors for further exploration of differing approaches for this particular dataset and its corresponding framing.

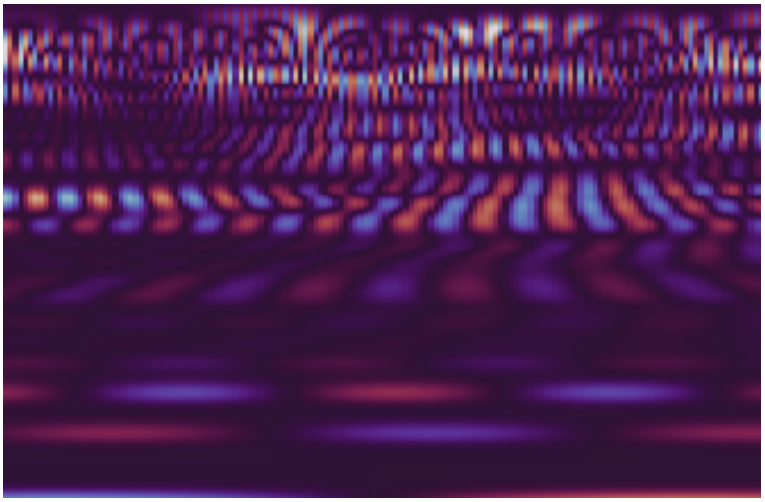


Figure 5.6: Animal Class Scalogram

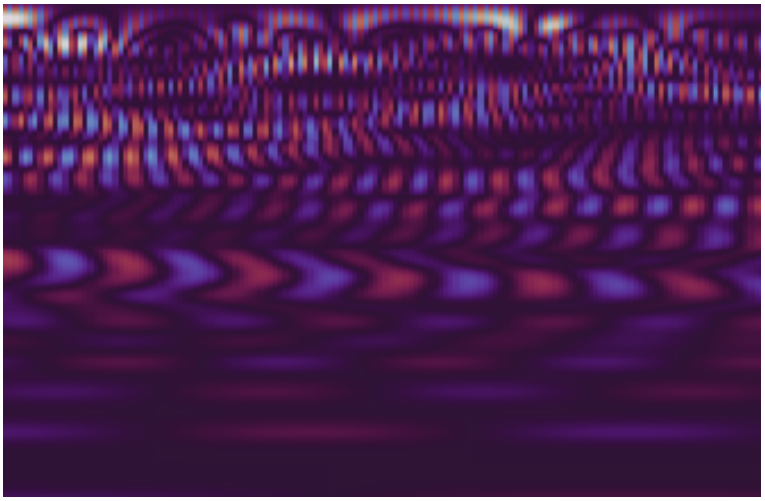


Figure 5.7: Artifact Class Scalogram

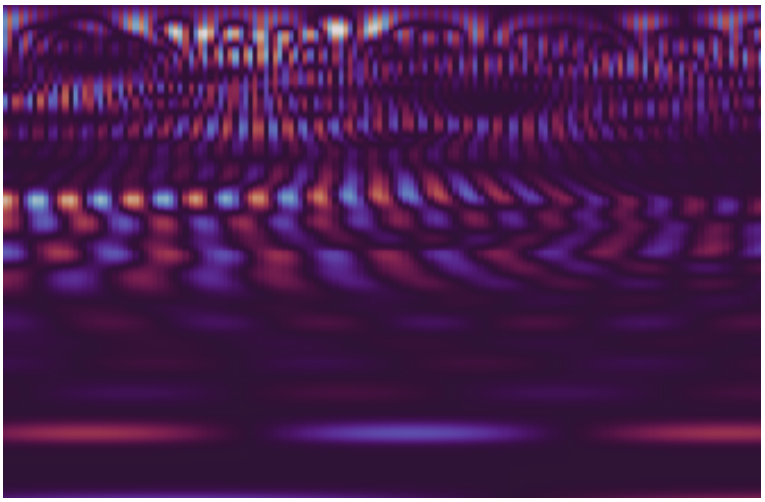


Figure 5.8: Scene Class Scalogram

5.0.4 Methodologies

The main procedure follows a mix of hybrid modeling [28], and a final gradient boosting classifier as the addressee of the latent representation outcome [29]. Yet, due to the large amount of noise and complexity in the data, I focused on a balanced and calculated reduction of the dimensions, and this required a certain flux of operations. The certain part was to pass the data into the AutoLSTM, but it should be noted that since an autoencoder is a dimensionality reduction tool by itself, I had to decide whether it would be wise to pass the raw ROI series or operate a reduction before-hand. After a further experimentation, I reached a point where the best first step was to actually reduce the dimensions for each one of the five signals after producing a correlation matrix - since voxels are multiple, multi-layered and in need of a clear-out based on corresponding algorithms [30] [31] - which consisted of **212** columns per signal, a number decided after the largest correlation matrix size found, and used as padding for the rest of the signals.

Thus, the resulted first matrix was of shape *(13136, 5, 212)*, a significant drop from the initial shape of *(13136, 5, 6960)*, with *6960* stemming from the padding of the initial signals for each subject resulting in 696, taken from the largest one, and multiplied by *10*, one for each visual region. Afterwards, a further reduction was applied through a plain Principal Component Analysis (PCA) in order to reach the point of 80 components - features -, a number immediately decided after a thorough experimentation on the AutoLSTM itself. After the extraction of the 80 components, a standardization was applied in order to scale to unit variance using sklearn's *StandardScaler* module.

Hereupon, the AutoLSTM was ready to accept the full matrix of all *13136* samples, *5* time-steps, and *80* features, since I was still transforming the whole dataset before considering splitting feeding it to the final classifier. The best architecture for the AutoLSTM proved to be that of 20 epochs, 150 batch size, a learning rate of 0.03, Adam optimizer, and a sequential 80, 40, 20, and **6** LSTM units, with the final one being the bottleneck layer and the core of the encoder itself, which contains the most compressed version of the initial data, resulting in a matrix of shape ***(13136, 6)*** . Finally, only the encoder was used in order to take this last matrix as the original version of data.

Subsequently, only the bottleneck layer was kept, and the Kera's *predict* was called on it with the matrix

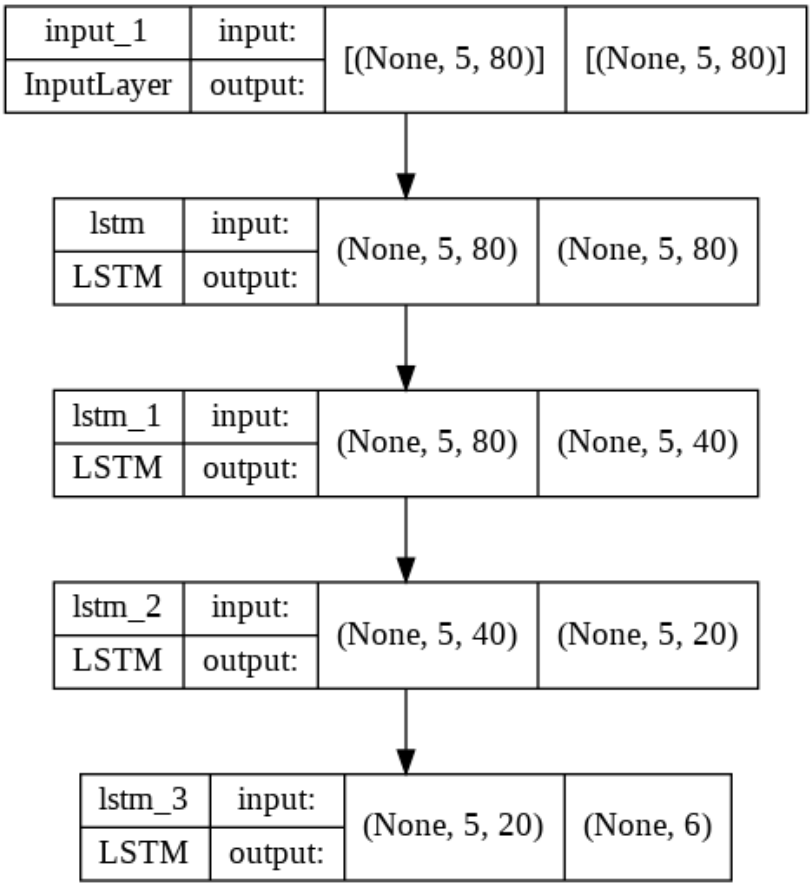


Figure 5.9: Encoder Compression

as input, in order to derive the final compression. Then, a standardization and a simple butter filter of the *signal* module was applied in order to smoothen out the edges of the final signals, and the matrix was ready to be passed on a promising classifier.

With that, I chose to pass the data into a gradient boosting classifier, CatBoost [32], provided by Russia's Yandex. A key component which made me take this decision was CatBoost's embedded mechanisms of preventing overfitting Moreover, it's proven to have quite the potential on heterogeneous data, and BOLD signal tends to acquire that title due to high data redundancy. Finally it's one of the newest gradient boosting modules, thus it was an interesting take to learn from its documentation, and subsequently explore further variables.

Consequently, the resulting matrix was passed to a CatBoostClassifier model after applying a parameter grid-search for best hyperparameter tuning, after splitting the data at the exact same porportion of the study at [12], thus 0.2 validation split, equal to *10508* train size, and *2628* test size. The list of hyperparameters investigated at the grid-search are listed below:

iterations, which means the number of trees,

l2_leaf_reg, meaning the L2 regularization coefficient of the cost function,

random_strength, which applies the randomness for tree split scoring, and

bagging_temperature, hence the Bayesian Bootstrap optimization (bayesian bootstrap is used for random weight assignment). Higher values result to bolder bagging.

Aside from these, the ***depth*** of the trees was kept at 7, with a ***learning rate*** of .03.

As such, the final proposed method is presented below both as a diagram and a set of rules:

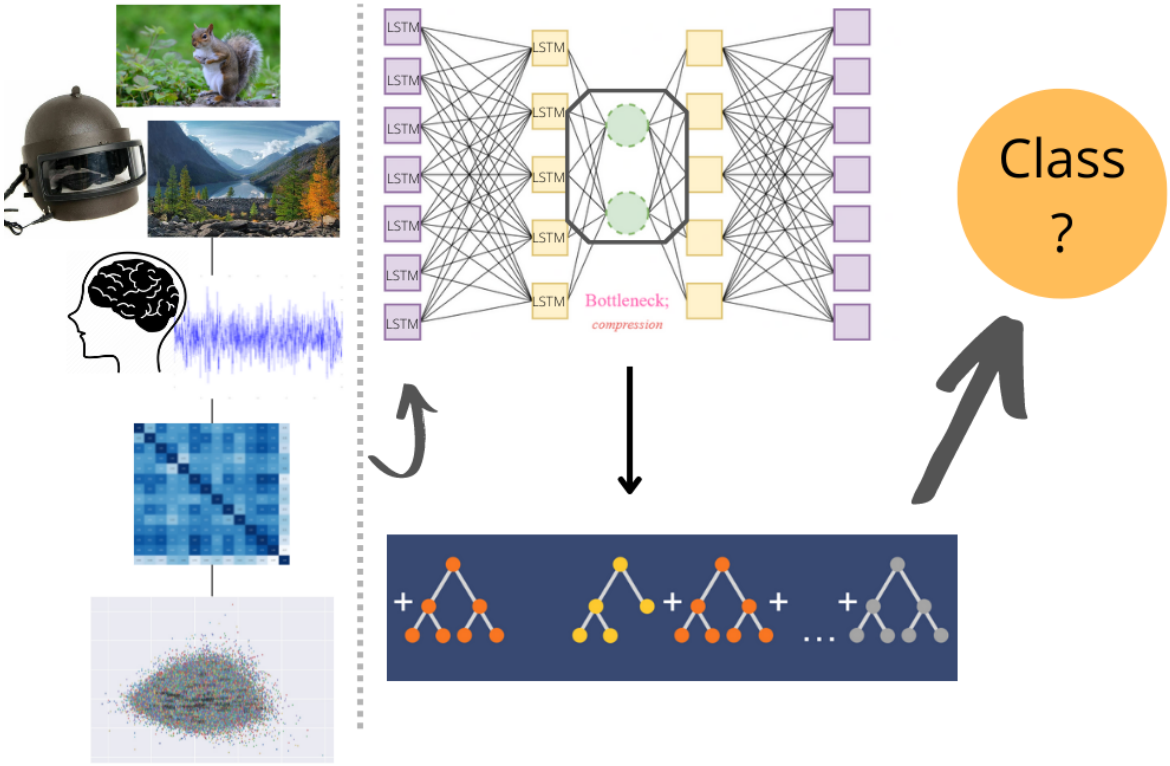


Figure 5.10: Proposed method

- Stimuli is being presented to the subjects, with the main classes expressed as animals, artifacts or scenes.
- Brain activation is manifested and therefore extracted as time-series data.
- A correlation matrix is applied, and the most significant variable are being kept.
- PCA is being applied in order to reduce the dimension of the matrix.
- The data is fed to the optimized Autoencoder-LSTM model, in order to further reduce the dimensions and eliminate the multi-timestep component.
- The data is fed to a gradient boosting classifier.
- The classifier predicts the class.

5.1 Results

The best CatBoost hyperparameters are listed below:

Bagging_temperature	0.5
bootstrap_type	’Poisson’
depth	7
early_stopping_rounds	25
eval_metric	’MultiClass’
grow_policy	’SymmetricTree’
iterations	750
l2_leaf_reg	5
leaf_estimation_iterations	100
learning_rate	0.03
logging_level	’Verbose’
loss_function	’MultiClass’
od_type	’IncToDec’
random_strength	0.15
score_function’	’L2’
task_type	’GPU’

The training lasted about 5 hours, with a *cross-validation* factor of 2, resulting in a modest 64.15% general accuracy, yet it showed a significant increase at the ability of differentiating between the *animal* and *scene* classes, as is presented below:

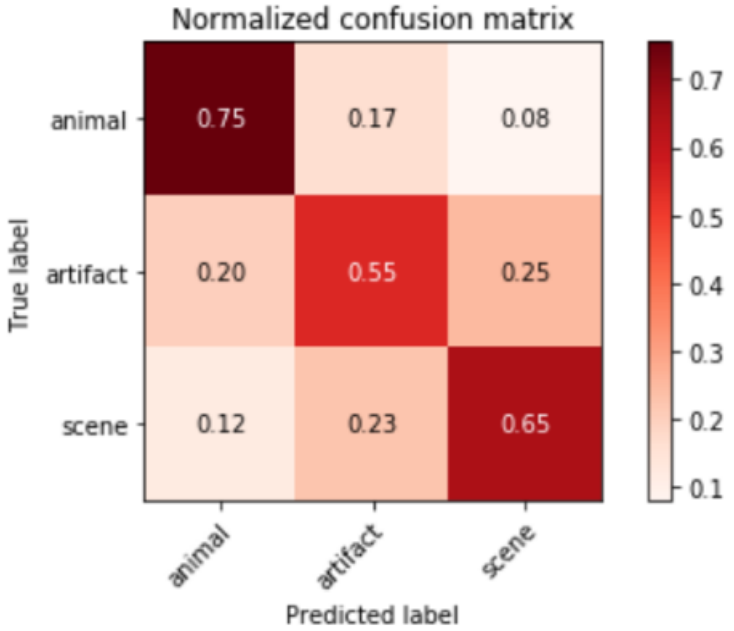


Figure 5.11: Plain LSTM method of [12].

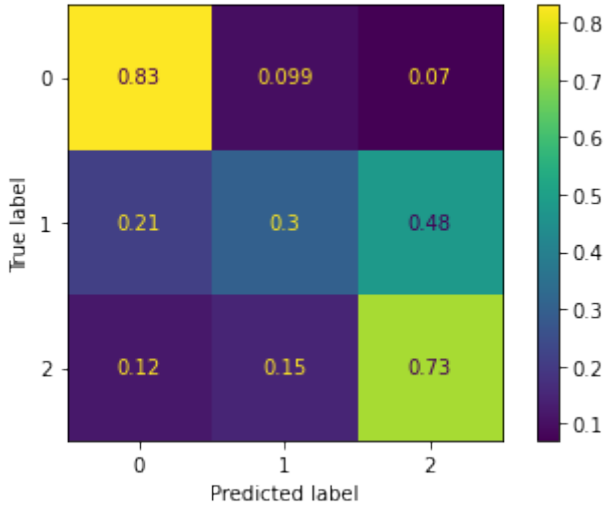


Figure 5.12: Normalized Confusion Matrix results of proposed method.

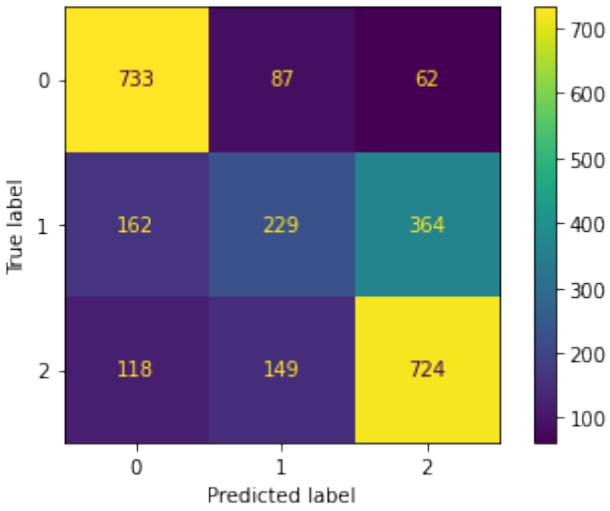


Figure 5.13: Confusion Matrix results of proposed method.

The middle class, hence the artifact, proved to be the hardest to predict out of the three, since many factors come to play, with the main one being the possible overlap between scenes and artifacts.

Moreover, the metrics of **precision**, **recall** and **f1-score** are showed below:

Score Results			
↓ **stimuli** | **score** →	f1-score	recall	precision
animal	0.77	0.83	0.72
artifact	0.37	0.30	0.49
scene	0.67	0.73	0.62

Chapter 6: Conclusions

With that, the most meaningful drawn conclusions is that this particular dataset formation might aid the overlap between some *artifact* and *scene* images, thus making the distinction a step more difficult. Although a model like word2vec might put an image to the artifact cluster based on the recorded written descriptions and its title, the perspective of a part of a living organism like a brain is multi-dimensional and thus prone to multiple interpretations of a stimuli based on depth, coloring, physiological and psychological factors, so it fires signals similar to as if it was a scenery, proving once again that each person might view the same but ***comprehend*** differently, resulting even to insubstantial human interaction.

Despite this, the resulted machine learning pipeline managed to acquire decent results, especially for the coupling of *animal* and *scene* based signals, therefore offering a fresh approach to the management of multiple, mass subject experiments. An extension of this work would possibly be an attentive incorporation of scalograms in order to derive further distinguishing features, promising better results and an extended signal apprehension.

6.1 Challenges

fMRI data are interesting, unique, and can be approached via a plethora of ways, whether it's the scans themselves or the extracted signals which manifest as time-series. Therefore, I was in a significant dilemma as for what might be the best approach since the BOLD5000 dataset offer both those modes. If I chose to take advantage of the scans, then I would have to make sure that I correctly applied the masks and the pre-processing which tends to leave the machine learning spectrum, so a mistake on this separate bio-signal preprocessing would possibly lead to problematic classifications later. Even though I was deeply interested at the biomedical aspect of the fMRI scans, the mere size of the total number of files, the uncharted territory of biological knowledge, and the actual challenge of the study at [12] led to the decision to take upon the time-series version of the data.

Moreover, I had to carefully examine the existing study of the corresponding data framing in order to not deviate from the initial idea, yet I had to form a wholly new approach, since I had to keep the same amount of voxels, time-steps and TRs, yet discover a novel sequences of pre-processing for all of them. Even though the AutoLSTM was significant enough on its own, it heavily relied on the steps taken before it, and that exact point of what exactly should be fed into it was a large part of the project itself. Thus, not being able to formulate another version of the data, for example leaving aside certain brain regions or one of the 5 time-steps significantly limited the freedom of movement.

Furthermore, a full time, 6 month internship was significantly consuming, leading to tiredness and less available time, so I had to create a new weekly schedule that included the bibliography study, with the weekends being dedicated to the experiments and writing the project's thesis, so there were moments of substantial psychological and mental tiredness.

Lastly, the training of the gradient boost algorithm was one of the most resource hungry, so I had to get a premium subscription on Google Colab in order to reduce the time needed for training, providing me with both more quick results and time to manage and recontstruct the strategy to produce better results.

6.2 Future Work

The key takeaway of this project was that the brain's visual regions might not participate equally, which they don't, so another set of experiments could be conducted based on single or less than all of the 10 regions, and keep this stream of consciousness in accordance with a varying number of time-steps and TRs used each time.

Subsequently, 1D Convolutional Neural network could be an interesting take in order to examine its abilities on extracting temporal information of time-series sequential bio-signals. Although the proposed framework managed a good outcome on the animal and scene classes, there still has to be gained a better understanding of any voxel information that I couldn't get via this pipeline, so that the artifact class can become more easily distinguishable, and by extension reveal the correlated regions that can be attributed with those activation, or prove that they simply don't have anything else to offer, thus signifying that these might not be the correct views of the brain for this particular class.

Lastly, multiple classifiers could be used for each **separate** class with this pipeline attached serving as the guide, or even incorporate alien datasets as boost for the comprehension of these four subjects, and reversely aiding the generalized approach for any brain, giving us the opportunity to speak with more diligence and certainty for the human's brain activations.

6.2.1 Incorporation of Scalograms

Due to time and resource limitations, I couldn't explore the extracted scalograms to their full extent. Scalograms tend to be in a go-to method for biosignal data, so their capability might prove to be exceptional for this problem too.

Due to their mechanism, the dataset should not be padded based on the brain regions, so it won't use the same establishment as [12], so in this case, a direct comparison would not be possible, but it would help the reasearch of voxel-to-scalogram conversion and whether a transfer learning application could catch information not previously gathered, since there's a plethora of predefined models like VGG or ResNet, which interestingly enough were trained on ImageNet, the same image set that was used as stimuli for the subjects.

Chapter 7: Bibliography

References

[1] Guoan Cheng, Ai Matsune, Qiuyu Li, Leilei Zhu, Huaijuan Zang, Shu Zhan, ”Encoder-Decoder Residual Network for Real Super-resolution”, Department of CSIE, Hefei University of Technology, 230000, Hefei, China, April 2020.

[2] Jinghui Chen, Saket Sathe, Charu Aggarwal, Deepak Turaga, ”Outlier Detection with Autoencoder Ensembles”, June 2017

[3] Neda Tavakoli, Sima Siami-Namini, Mahdi Adl Khanghah, Fahimeh Mirza Soltani, Akbar Siami Namin, ”Clustering Time Series Data through Autoencoder-based Deep Learning Models”, April 2020.

[4] Kai Qiao, Jian Chen, Linyuan Wang, Chi Zhang, Lei Zeng, Li Tong, Bin Yan, ”Category Decoding of Visual Stimuli From Human Brain Activity Using a Bidirectional Recurrent Neural Network to Simulate Bidirectional Information Flows in Human Visual Cortices”, PLA Strategic Support Force Information Engineering University, Zhengzhou, China, July 2019

[5] Ming-Chou Ho, Hsin-An Shen , Yi-Peng Eve Chang, Jun-Cheng Weng, ”A CNN-Based Autoencoder and Machine Learning Model for Identifying Betel-Quid Chewers Using Functional MRI Features”, June 2021

[6] Ioannis Livieris, Niki Kiriakidou, Stavros Stavroyiannis, P. E. Pintelas, ”An advanced CNN-LSTM model for cryptocurrency forecasting”, January 2021

[7] Po-Hsuan Chen, Xia Zhu, Hejia Zhang, Javier S. Turek, Janice Chen, Theodore L. Willke, Uri Hasson, Peter J. Ramadge, ”A Convolutional Autoencoder for Multi-Subject fMRI Data Aggregation”, August 2016

[8] K. O. Gupta, P. N. Chatur, ”Gradient self-weighting linear collaborative discriminant regression classification for human cognitive states classification”, March 2020

[9] Dong Wen, Zhenhao Wei, Yanhong Zhou, Guolin Li, Xu Zhang, Wei Han, ”Deep Learning Methods to Process fMRI Data and Their Application in the Diagnosis of ognitive Impairment: A Brief Overview and Our Opinion”, April 2018

[10] João R. Sato, Jorge Moll, Sophie Green, John F.W. Deakin, Carlos E. Thomaz, Roland Zahn, ”Machine learning algorithm accurately detects fMRI signature of vulnerability to major depression”, July 2015

[11] Vijay Rowtula, Subba Reddy Oota, Manish Gupta, Raju S. Bapi, ”A Deep Autoencoder for Near-Perfect fMRI Encoding”, September 2018

[12] Arash Jamalian, Rafi Ayub, Faraz Fadavi, ”Categorization of seen images from brain activity using sequence models”, Stanford University, Fall 2019

[13] Nadine Chang, John A. Pyles, Abhinav Gupta, Michael J. Tarr, Elissa M. Aminoff, ”BOLD5000 A public fMRI dataset of 5000 images”, September 2018

[14] Piyawat Saengpetch, Luepol Pipanmemekaporn, Suwatchai Kamolsantiroj, ”Visual Representation Model for fMRI-based Brain Decoding”, Department of Computer and Information Science King Mongkut’s University of Technology North Bangkok, Bangkok, Thailand 10800, April 2019

[15] Sunao Yotsutsuji, Miaomei Lei, Hiroyuki Akama, ”Evaluation of Task fMRI Decoding With Deep Learning on a Small Sample Dataset”, February 2021

[16] Barry Devereux, Colin Kelly, Anna Korhonen, ”Using fMRI activation to conceptual stimuli to evaluate methods for extracting conceptual representations from corpora”, 6 June 2010

[17] Yue Bai, ”Deep-Learning based Analysis of fMRI data: A Visual Recognition Study”, Northeastern University, Boston, Massachusetts, December 2019

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, ”U-Net: Convolutional Networks for Biomedical Image Segmentation”, Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany, May 2015

[19] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, Olaf Ronneberger, ”3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation”, June 2016

[20] Haxby, J V; Gobbini, M I; Furey, M L; Ishai, A; Schouten, J L; Pietrini, P, ”Distributed and overlapping representations of faces and objects in ventral temporal cortex”, University of Zurich, 2001

[21] Sun Zhuochen, Yang Fenglei, Song Kexin, ”High-resolution brain fMRI reconstruction via double cooperative network learning”, Shanghai University, Shanghai China, July 2020

[22] Michele Svanera, Mattia Savardi, Sergio Benini, Alberto Signoroni, Gal Raz, Talma Hendler, Lars Muckli, Rainer Goebel, Giancarlo Valente, ”Transfer learning of deep neural network representations for fMRI decoding”, October 2019

[23] Ziya Yu, Chi Zhang, Linyuan Wang, Li Tong, Bin Yan, ”A Comparative Analysis of Visual Encoding Models Based on Classification and Segmentation Task-Driven CNNs”, PLA Strategy Support Force Information Engineering University, Zhengzhou 450001, China, August 2020

[24] Yeong-Hyeon Byeon, Sung-Bum Pan, Keun-Chang Kwak, ”Intelligent Deep Models Based on Scalograms of Electrocardiogram Signals for Biometrics”, Department of Control and Instrumentation Engineering, Chosun University, Gwangju 61452, Korea, February 2019

[25] Mohammed I. Al-Hiyali, Norashikin Yahya, Ibrahima Faye, Zia Khan, ”Classification of BOLD FMRI Signals using Wavelet Transform and Transfer Learning for Detection of Autism Spectrum Disorder”, Khaled AlsaihLaboratoireHubert CurienUniversite deLyonSaint- ´Etienne, France, April 2021

[26] Samiul Based Shuvo, Shams Nafisa Ali, Soham Irtiza Swapnil, Taufiq Hasan, Mohammed Imamul Hassan Bhuiyan, ”A Lightweight CNN Model for Detecting Respiratory Diseases from Lung Auscultation Sounds using EMD-CWT-based Hybrid Scalogram”, September 2020

[27] Jinlong Hu, Yuezhen Kuang, Bin Liao, Lijie Cao, Shoubin Dong, Ping Li, ”A Multichannel 2D Convolutional Neural Network Model for Task-Evoked fMRI Data Classification”, December 2019

[28] Subba Reddy Oota, rowtula Vijay, Manish Gupta, Bapi Raju Surampudi, ”StepEncog: A Convolutional LSTM Autoencoder for Near-Perfect fMRI Encoding”, Centre for Cognitive Science International Institute of Information Technology Hyderabad - 500 032, INDIA, July 2019

[29] Zhuangwei Shi, Yang Hu, Guangliang Mo, Jian Wu, ”Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction”, August 2021

[30] Chunlei Shi, Jiacai Zhang, Xia Wu, ”An fMRI Feature Selection Method Based on a Minimum Spanning Tree for Identifying Patients with Autism”, School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China, December 2020

[31] Ceyhun Can Ülker, Tevfik Aytekin, ”Improving the Performance of Active Voxel Selection in the Analysis of fMRI Data Using Genetic Algorithms”, Bahcesehir University Computer Engineering Department Ciragan Caddesi 34353, Besiktas Istanbul, Turkey, September 2013

[32] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, Andrey Gulin, ”CatBoost: unbiased boosting with categorical features”, Yandex, Moscow, Russia, Moscow Institute of Physics and Technology, Dolgoprudny, Russia, January 2019