الجمهورية الجزائرية الديمقراطية الشعبية
DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
وزارة التعليم العالي و البحث العلمي
Ministery of Higher Education and Scientific Research

École Nationale Polytechnique

Electronics Engineering Department

Final Year Project Dissertation
in partial fulfilment of the requirements for:
Electronics Engineer's Degree

Deep Learning For Predicting Epileptic Seizures Using EEG Signals

Author:
TERBOUCHE Hacène

Presented and defended in September 2nd, 2020 before the members of jury:

President	Cherif	LARBES	Prof.	ENP, Algiers
Supervisor	Mourad	Adnane	Prof.	ENP, Algiers
Co-supervisor	Youcef	Ouadjer	PhD Student.	ENP, Algiers
Examiner	Nesrine	BOUADJENEK	PhD.	ENP, Algiers

ENP 2020

École Nationale Polytechnique (ENP)
10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie
www.enp.edu.dz

الجمهورية الجزائرية الديمقراطية الشعبية
DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
وزارة التعليم العالي و البحث العلمي
Ministery of Higher Education and Scientific Research

École Nationale Polytechnique

Electronics Engineering Department

Final Year Project Dissertation
in partial fulfilment of the requirements for:
Electronics Engineer's Degree

Deep Learning For Predicting Epileptic Seizures Using EEG Signals

Author:
TERBOUCHE Hacène

Presented and defended in September 2nd, 2020 before the members of jury:

President	Cherif	LARBES	Prof.	ENP, Algiers
Supervisor	Mourad	Adnane	Prof.	ENP, Algiers
Co-supervisor	Youcef	Ouadjer	PhD Student.	ENP, Algiers
Examiner	Nesrine	BOUADJENEK	PhD.	ENP, Algiers

ENP 2020

École Nationale Polytechnique (ENP)
10, Avenue des Frères Oudek, Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie
www.enp.edu.dz

ملخص ـ يُعد التنبؤ بنوبات الصرع مشكلة عسيرة تتضمن تحديد بداية النوبة باستخدام إشارات المخطط الكهربائي للدماغ (*EEG*)، إما عن طريق طبيب أعصاب خبير، أو باستخدام تقنيات تعلُّم الآلة. وفي هذا العمل، سوف نستفيد من التطورات الحديثة في تقنيات التعلم العميق باقتراح هندستين: أما الأولى فتعتمد على شبكة عصبية التفافية أحادية البعد (*CNN 1D*) تتعلم التمثيلات الهرمية للتخطيط الكهربائي للدماغ، دون خطوات المعالجة المسبقة، وأما الثانية فتعتمد على شبكة التفافية متكررة طويلة المدى (*LRCN*)، ولها القدرة على تعلُّم تمثيلات مختلفة للهيكل الزماني المكاني لإشارة *EEG* . هذا وتوضح النتائج التجريبية موثوقية النماذج من خلال الحصول على متوسط دالة فاعلية المستقبل (*ROC*) يبلغ 0.848 لنموذج *CNN 1D* و 0.873 لنموذج *LRCN* .

كلمات مفتاحية ـ *EEG* ، توقع نوبة الصرع، التعليم المعمق ، *CNN* ، *LSTM* .

Résumé— La prédiction des crises d'épilepsie est un problème difficile qui consiste à mettre en évidence le début d'une crise à l'aide de signaux d'électroencéphalogramme (EEG), soit par un neurologue expérimenté, soit automatiquement à l'aide de techniques d'apprentissage machine. Dans ce travail, nous allons tirer profit des progrès récents des techniques d'apprentissage profond et proposer deux architectures. La première est basée sur un réseau neuronal convolutif unidimensionnel (1-D CNN) qui apprend les représentations hiérarchiques de l'EEG, sans étapes de prétraitement. La seconde est basée sur un réseau convolutif récurrent à long terme (LRCN), il a la capacité d'apprendre différentes représentations de la structure spatio-temporelle du signal EEG. Les résultats expérimentaux démontrent la fiabilité des modèles en obtenant une moyenne de la fonction efficacité du récepteur (ROC) de 0,848 pour le modèle 1-D CNN, et de 0,873 pour le modèle LRCN.

Mots-clés : EEG, Prédiction de la crise d'épilepsie, Apprentissage approfondie, CNN, LSTM.

Abstract— Epileptic seizure prediction is a challenging problem which consists in identifying a seizure onset using electroencephalogram (EEG) signals, either by an experienced neurologist or automatically using machine learning techniques. In this work, we will take advantage from recent advances of deep learning techniques and propose two architectures. The first model is based on one dimensional convolutional neural network (1-D CNN) architecture that learns hierarchical representations of EEG with no preprocessing steps. The second model is based on long term recurrent convolutional network (LRCN) which has the capacity to learn different representations of the spatiotemporal structure of EEG signal. Experimental results demonstrate the reliability of the models by achieving a mean Area Under Curve (AUC) Receiver Operating Characteristics (ROC) of 0.848 for 1-D CNN model, and 0.873 for LRCN model.
Index-terms : EEG, Epileptic seizure prediction, Deep Learning, CNN, LSTM.

Acknowledgement

First and foremost, I would like to take the opportunity to express my deepest gratitude and thanks to my project supervisor Mr Mourad ADNANE, Professor with Ecole Nationale Polytechnique, for his constant guidance, assistance and support as well as all the knowledge shared during the course of this project

I give special thanks to Mr OUDJER Youcef, a PhD student at Ecole Nationale Polytechnique, for introducing me this topic as well for the support along the way.

I also wish to thank the President of jury Mr Cherif Larbes, Professor at ENP, and the examiner: Ms Nesrine BOUADJENAK, Professor at the ENP, for accepting to be members of the reading committee and for their constructive analysis of the present work

Dedication

I would like to dedicate my work to my parents and loved ones, who always helped me throughout the entire educational pathway. They always stood alongside me during tough moments, offering me their wholehearted support and elating words of encouragement.

I offer this dissertation to all my professors at the Electronics Department at Ecole Nationale Polytechnique, who had spared no effort to teach me, and who have been always supportive and available.

I also dedicate this project to my friends, both from the school and outside the school. All those who have supported me throughout the process. I will always appreciate everything they have done.

Contents

List of Figures

Acronyms

AI Artificial Intelligence. 13

ANN Artificial Neural Network. 31, 76

ApEn Approximate Entropy. 24

AR Auto-regressive. 24

ARMA Auto-Regressive Moving Average. 24

AUC Area Under Curve. 56, 69, 76, 78

CNN Convolutional Neural Network. 14, 15, 43, 67, 68, 78

DL Deep Learning. 15, 41, 67

EEG Electroencephalography. 12, 15, 25

ES Epileptic Seizures. 15, 25

FN False Negative. 29

FP False Positive. 29

FPR False Positive Rate. 29, 56, 74

FT Fourier Transform. 24

GAP Global Average Pooling. 68

HOS High Order Spectra. 24

LLE Largest Laypunov Exponent. 25

LRCN Long-term Recurrent Convolutional Network. 14, 70, 77, 78

LSTM Long Short-Term Memory. 48, 78

MA Moving Average. 24

ML Machine Learning. 13, 15, 30, 41

MSE Mean Square Error. 38

PSD Power Spectrum Density. 24

RFC Random Forest Classifier. 76

RNN Recurrent Neural Network. 15, 46

ROC Receiver Operating Characteristics. 56, 76, 78

SOP Seizure Occurrence Period. 28

SPH Seizure Prediction Horizon. 28

SVM Support Vector Machine. 31, 75, 76

TN True Negative. 29

TP True Positive. 29

TPR True Positive Rate. 56, 74

TSC Time Series Classification. 28

WT Wavelet Transform. 24

Introduction

Epilepsy Epilepsy is a common brain disorder that, according to the World Health Organization (WHO), affects around 50 million people around the world, making it one of the most common neurological diseases globally [1]. In Algeria, it was estimated to have 400.000 people with epilepsy between children and adults [2]. Epilepsy is characterized by recurrent, unprovoked seizures. Seizures are the result of a transient and unexpected electrical disturbance of the brain and excessive neuronal discharge of the electrical activity of the brain. They can vary from the shortest failure of attention to intense and extended convulsions. They can also range in frequency from less than 1 per year to several per day.

Epilepsy is a source of constant anxiety for patients due to the unexpected onset of seizures involving the triggering of manifestations of which they lose consciousness. Moreover, people with epilepsy tend to have physical issues such as break and injuries related to seizures, which decrease the patient's quality of life. In addition, epileptic patients can have reduced access to educational opportunities, a withholding of the opportunity to obtain a driving license, barriers to enter particular occupations, and reduced access to health and life insurance. Currently there is no cure for epilepsy. Many patients' seizures can be controlled, but not cured, with medication. Up to two-thirds of people living with epilepsy could become seizure free with appropriate use of antiseizure medicines [1].

People that remain unresponsive to available medication, may become candidates for surgical intervention. This situation is called drug-resistant epilepsy (DRE) and people suffering of this situation are forced to deal with it in everyday life [3]. Of those unresponsive to anticonvulsant medication, 7% to 8% may profit from epilepsy surgery. However, about 25% of people with epilepsy will continue to experience seizures even with the best available treatment [4]. Besides, for those responsive to medication, many antiepileptic medicines have significant side effects that have a negative impact on quality of life. Consequently, the prediction of epileptic seizures would greatly contribute to improving the quality of life of epileptic patients.

EEG Electroencephalography (EEG) is the electrical recording of the oscillations of potential generated by brain sources. The first EEG recordings were obtained in 1929, but it remains one of the principal techniques for extracting information from the human brain for research and clinical purposes. It is considered as the most powerful diagnostic and analytical tool of epilepsy. Neuroscientists have found that the brain activity of epileptic patients, based on the EEG recordings, can be classified into four different states: *preictal*, which is defined as the time duration prior to seizure, *ictal state* (during the seizure), *postictal state* that is related to the period after seizure and finally the *interictal state* which refers to the time between seizures. As the project objective is to build a seizure prediction system, the key challenge is to discriminate between the *preictal* and *interictal* brain states [5].

Machine Learning Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that covers its statistical part. While AI is the ability of machines to mimic human brain, the aim of machine learning is to understand the structure of data and fit that data into models that can be used for further prediction. The term *machine learning* was invented by *Arthur Samuel* in 1959, an American computer scientist that expressed that ”ML gives computers the ability to learn without being explicitly programmed”. In 1997, Tom M. Mitchell provides a formal definition: ”A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”. Machine learning differs from traditional programming paradigms in that it can learn from data without relying on rules-based programming.

ML techniques can resolve multiple tasks in different domains such as regression, classification, clustering, dimension reduction, anomaly detection, density estimation, recommender systems and many others. Thus, these tasks fit with a wide variety of applications like object detection, face recognition, speech recognition, machine translation, spam email detector, robotics and so on [6], [7].

Machine Learning and Epilepsy In the case of epilepsy, machine learning can be very beneficial if it solves the problem of epileptic seizure prediction. Epileptic seizure prediction algorithms , coupled with surveillance systems, make it possible to envisage three major applications. First, during long-term hospitalized patient EEG recordings, the implementation of a warning system that allows the medical staff to see the onset of a seizure. Secondly, the creation of a mobile device warning the patient of an incoming seizure. By being alerted by a notification, the person has the time to put himself in safety before the start of the seizure in order to avoid dangerous situations. The device can also warn someone else who take care of this patient. In this application, we do not avoid the actual problem (seizure), but it is very valuable to minimize damages. Finally, in the longer term, an anticipation system of seizures coupled with an implantable therapeutic device responds to the control of the seizures by means of implanted stimulators [8]. Neurostimulation is an acceptable way of treating seizures in people with drug-resistant epilepsy. It’s usually considered in people who are not appropriate for epilepsy surgery or when surgery is not helpful [9].

Our work During the last decade, many classical machine learning algorithms (i.e. non-deep learning algorithms) have been proposed for the seizure prediction problem. Since EEG signals are different across patients because of the change in seizure type and location , most seizure prediction methods are as a result patient-specific. These algorithms acquire a diverse range of EEG feature extraction, selection, and classification techniques. However, the principal disadvantage of these methods is the excessive use of manually extracted features. Thus, it is challenging to determine the most discriminative features that best represent each class. In newer tendency, deep learning based seizure prediction algorithms are used, which combine the feature extraction and classification stages into a single automated framework. The aim of this project is to develop an automatic feature learning using deep learning based algorithms that can be applied for all patients with minimum feature engineering and preprocessing.

The main contributions of our work can be summarized as follows:

- Propose a 1-D Convolutional Neural Network (CNN) architecture that can learn different representations of iEEG (intracranial EEG) data efficiently.
- Propose a second architecture based on Long-term Recurrent Convolutional Network (LRCN), with its proper method to pre-process raw EEG data into a form suitable for this model.
- Use testing method, that proves the robustness of the proposed algorithms over different seizures, for performance evaluation and comparison of results.

Organization This dissertation is organized as follows:

- Chapter I provides a background on the context of epileptic seizure prediction, EEG signals and closed-loop systems. It gives an overview of the basic components of DNNs as well as a review of the literature.
- Chapter II describes software and hardware tools used in this project such as dataset, explory data analysis, preprocessing and cross-validation scheme.
- Chapter III presents the proposed models,explains the implementation of the project and all related technical details. Finally, it shows the results, makes some comments and draws the conclusions.
- The conclusion lays out a brief summary of what this dissertation is about. In addition, it explores the challenges that we faced during implementation. It also explores contributions that could be made in future works.

Chapter 1

Background

1.1 Introduction

The aim of this chapter is to provide the reader with the fundamental concepts that are related to the project. We will start by defining *Epilepsy* as a neurological disorder characterized by epileptic seizures and presenting its types. Then, we will discuss the Electroencephalography (EEG) signals, its characteristics, acquisition schemes and related analysis techniques. This allows us to make the relation between EEG signals and epileptic seizure. Since epileptic seizures arise inside the brain, one primary tool that specialist use in order to diagnose epilepsy is EEG. As our project goal is to predict seizures based on EEG data analysis, we will present two notions that may be confused *seizure detection* and *seizure prediction* and show the difference between them. This enables us to introduce the problem definition, where the seizure prediction problem can be treated as a binary classification problem. Classification is the process of predicting the target associated with each sample in the data. Classification task is done usually using Machine Learning (ML) algorithms through a supervised learning approach. However, classical ML-based algorithms used complex handcraft patient-specific feature engineering due to the high complexity of EEG signals. Therefore, we tend to use Deep Learning (DL) models with minimum signal processing treatment. A deep learning review is presented to allow the reader understand the basics of this field and apply them to the problem, starting from the single artificial neuron and the biological analogy to feed-forward, Convolutional Neural Network (CNN) and finally Recurrent Neural Network (RNN). Finally, we proceed to present the state-of-the-art DL-based approaches for Epileptic Seizures (ES) prediction.

1.2 Epilepsy

1.2.1 Definition of Seizure and Epilepsy

The International League Against Epilepsy (ILAE) established both conceptual and operational definitions for seizures and epilepsy.

- Conceptual definitions [10]
  - An epileptic seizure is a transient occurrence of signs and/or symptoms due to abnormal excessive or synchronous neuronal activity in the brain.
  - Epilepsy is a disorder of the brain characterized by an enduring predisposition to generate epileptic seizures, and by the neurobiologic, cognitive, psychological, and social consequences of this condition. The definition of epilepsy requires the occurrence of at least one epileptic seizure.
- Operational (practical) clinical definition of epilepsy [11]
  - Epilepsy is a disease of the brain defined by any of the following conditions:
    - At least two unprovoked (or reflex) seizures occurring > 24h apart
    - One unprovoked (or reflex) seizure and a probability of further seizures similar to the general recurrence risk (at least 60) after two unprovoked seizures, occurring over the next 10 years
    - Diagnosis of an epilepsy syndrome

As a word, *seizure* derives from the Greek meaning *to take hold*. Modern popular terminology uses the word *seizure* for any sudden and severe event (for example, ”he had a heart seizure”). Many physical or psychological sudden events, some of them not even pathological, all resemble epileptic seizures in some ways. To emphasize this usage, typically, we will refer to an *epileptic seizure* or *ES*

1.2.2 Seizure Types and Classification

Classification of a seizure begins with historical elicitation or observation of certain symptoms and signs (sometimes referred to as semiology of seizure) that are known to be associated with common seizures. The key symptoms and signs cannot be matched in one-to-one relationships with seizure types because some symptoms appear in more than one seizure type. Conversely, a seizure type often associates with multiple symptoms.
In 2017, the ILAE released a new seizure classification system that includes both a basic and expanded version [12]. The expanded version provides more subcategories for more detailed characterization of the seizures.

ILAE 2017 Basic Classification of Seizures Types

Figure 1.1 shows the basic classification. Seizures are first categorized by type of onset. Focal-onset seizures are defined as ”originating within networks limited to one hemisphere. They may be discretely localized or more widely distributed. Focal seizures may originate in sub-cortical structures. Generalized from onset seizures are defined as originating at some point within, and rapidly engaging, bilaterally distributed networks. A seizure of unknown onset may still evidence certain defining motor or non-motor characteristics. With further information or further observed seizures, a reclassification of unknown onset seizures into focal or generalized onset categories may become possible. Therefore, ”unknown onset” is not a characteristic of the seizure, but a convenient placeholder for our ignorance. Further classification is optional. Focal seizures may be further classified by level of awareness (aware vs. impaired awareness) and/or motor vs. non-motor onset. Awareness pertains to knowledge of self and environment during a seizure. Generalized onset seizures are divided into motor and non-motor (absence) seizures. Level of awareness is not used as a classifier for generalized seizures, since the large majority (although not all) of generalized seizures are associated with impaired awareness. Seizures of unknown onset can be categorized as motor, including tonic-clonic, non-motor, or unclassified. The term unclassified comprises both seizures with patterns that do not fit into the other categories or seizures presenting insufficient information to allow categorization.

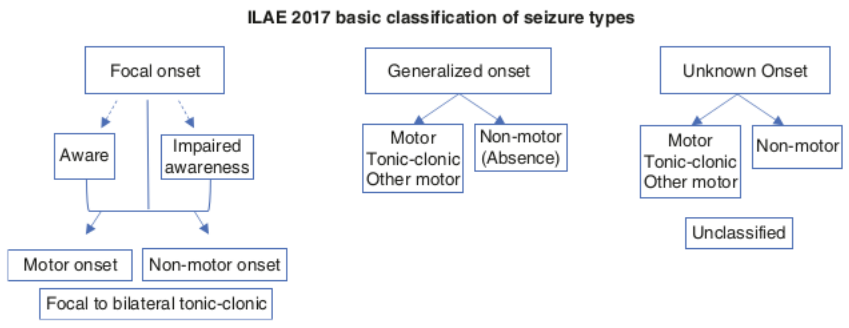


Figure 1.1: ILAE 2017 Basic seizure classification [12]

ILAE 2017 Expanded Classification of Seizures Types

The expanded classification (Fig 1.2) provides another level of seizure names, built on the framework of the basic classification. A focal seizure can be classified as focal aware or focal impaired awareness. Focal aware or impaired awareness seizures can optionally be classified by adding one of the motor onset or non-motor onset terms below, reflecting the earliest prominent sign or symptom other than awareness.

The classification of generalized onset seizures is similar to that of the 1981 classification, with addition of a few new types. Awareness usually is impaired with generalized onset seizures, so level of awareness is not used as a classifier for these seizures. The main subdivision is into motor and non-motor (absence) seizure types. The terms ”motor” and ”non-motor” are present in order to allow characterization of generalized onset motor or non-motor seizures about which nothing else can be said, but ”motor” and ”non-motor” may be omitted if the seizure name is unambiguous, for example, ”generalized tonic seizure”. The word ”generalized” can be omitted for seizures such as absence that present only with generalized onset.

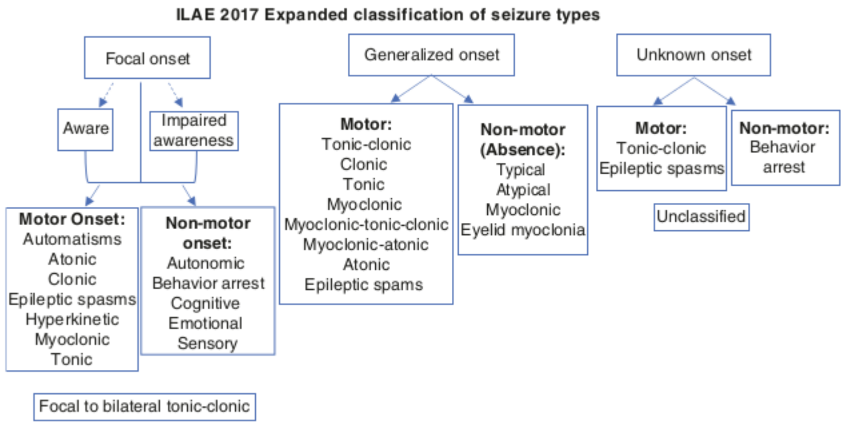


Figure 1.2: ILAE 2017 Expanded seizure classification [12]

1.3 EEG

1.3.1 What is EEG

Electroencephalography (EEG) is a measurement of potentials that reflect the electrical activity of the human brain. It is readily available test that provides evidence of how the brain functions over time. The EEG is widely used by physicians and scientists to study brain functions and to diagnose neurological disorders. The study of the brain’s electrical activity, through the EEG records, is one of the most important tools for diagnoses of neurological diseases, such as epilepsy, brain tumours, head injury, sleep disorders, dementia and monitoring the depth of anaesthesia during surgery [13]. It is also helpful for the treatment of abnormalities, behavioural disturbances (e.g, Autism), attention disorders, learning problems and language delay.
The first EEG recording machine was introduced to the world by *Hans Berger* in 1929. Berger, who was a neuropsychiarist from the University of Jena in Germany, used the German term *”elektrenkephalogramm”* to describe the graphical representations of the electric currents generated in the brain. He suggested that brain currents changed depending upon the functional status of the brain, such as, sleep, anaesthesia and epilepsy. This was a revolutionary idea that helped create a new branch of medical science called neurophysiology.

1.3.2 EEG Acquisition System

During the acquisition phase of EEG, a number of small disks called *electrodes* are placed in different locations on the surface of the scalp with temporary glues. Each electrode is connected to an amplifier (one amplifier per pair of electrodes) and an EEG recording machine. Finally, the electrical signals from the brain are converted into wavy lines on a computer screen to record the results.Figure 1.3 presents an example of how electrodes are placed on the scalp during the recording of EEG signals and EEG signals are displayed on a computer screen [14].
The electrodes detect tiny electrical charges that result from the activity of the brain cells. The charges are amplified and appear as a graph on a computer screen, or as a recording that may be printed out on paper. An expert then interprets the reading. EEG recordings, depending on their use can have from 1 to 256 electrodes recorded in parallel. This is called multi-channel EEG recordings. One pair of electrodes usually makes up a channel. Each channel produces a signal during an EEG recording.

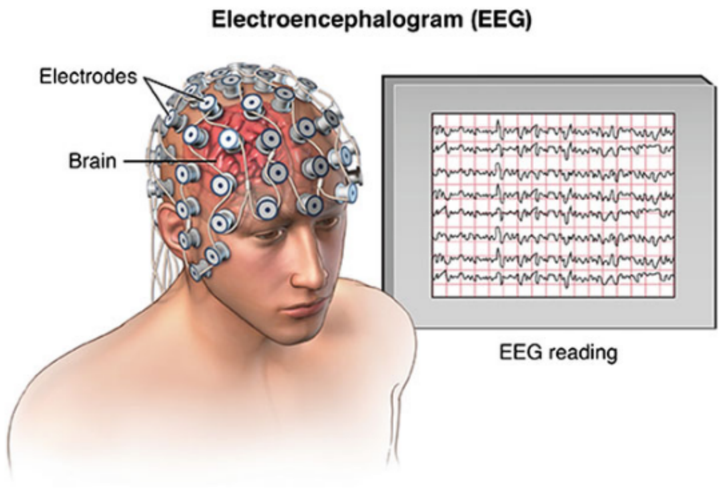


Figure 1.3: An illustration of EEG recording [14]

There are two types of EEGs, depending on where the signal is taken in the head: *scalp* or *intracranial*. For the *scalp* EEG, small electrodes are placed on the scalp with good mechanical and electrical contact. Special electrodes implanted in the brain during surgery result in *intracranial* EEG. On the other hand, the EEG measured directly from the cortical surface using subdural electrodes is called the *electrocorticogram (ECoG)*. The amplitude of an EEG signal typically ranges from 1 to 100 uV in normal adult, and it is approximately 10-20 mV when measured with subdural electrodes.

1.3.3 Placement of electrodes

The question of how to place the electrodes is important, because different lobes of cerebral are responsible for processing different types of activities. The standard method for the scalp electrode localization is the international 10-20 electrode system. The ”10” and ”20” represent actual distances between neighbouring electrodes are either 10 or 20 of the total front-back or right-left of the skull.
The positions are determined by the following two points; *nasion*, which is the point between the forehead and the nose, level with the eyes, and *inon* which is the bony prominence at the base skull on the mid-line at the back of the head.
Figure 1.4 presents the electrode position on the brain according to the international 10-20 system. Each location uses a letter to identify the lobe and a number to identify the hemisphere location. The letters F, T, C, P and O stand for Frontal, Temporal, Central, Parietal and Occipital, respectively. A ”z” refers to an electrode placed on the middle. Even numbers refer to electrode positions on the right hemisphere, whereas odd numbers refer to those on the left hemisphere.

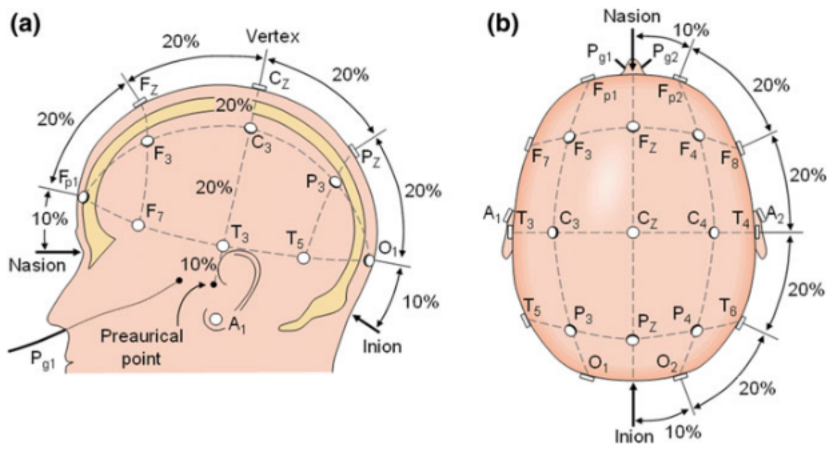


Figure 1.4: The international 10-20 electrode placement system

1.3.4 Common EEG montages

Since an EEG voltage signal represents the voltages at two electrodes, the display of the EEG for the reading EEG machine may be set up in several ways. The placement of electrodes is referred to as a montage. The EEG can be monitored with the following montages:

Bipolar montage

One pair of electrodes usually makes up a channel. Each channel (waveform) represents the difference between two adjacent electrodes. The entire montage consists of a series of these channels[15]. For example, we present a diagram of bipolar montage in Fig 1.5, where the channel ”Fp1-F3” represents the difference in the voltage between the Fp1 electrode and the F3 electrode. The next channel in the montage, ”F3-C3”, represents the voltage difference between F3 and C3, and so on, through the entire array of electrodes.

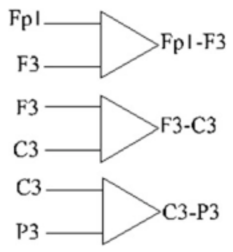


Figure 1.5: An illustration of the bipolar montage [15]

Referential montage

Each channel represents the difference between a certain electrode and a designed reference electrode [15]. In Fig 1.6, electrode A2 is considered as the reference electrode. There is no standard position for this reference. It is, however, at a different position than the ”recording” electrodes. Mid-line positions are often used because they do not amplify the signal in one hemisphere versus the other. Another popular reference is ”linked ears”, which is a physical or mathematical average of electrodes attached to both earlobes and mastoids.

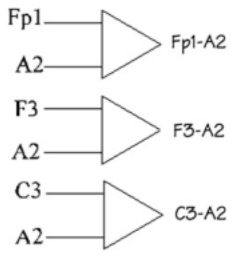


Figure 1.6: An illustration of the referential montage [15]

Average reference montage

The outputs of all of the amplifiers are summed and averaged, and this averaged signal is used as the common reference fro each channel. An illustration of the average reference montage is given by Fig 1.7

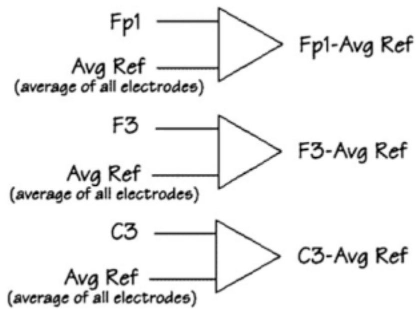

Fp1
Avg Ref
(average of all electrodes)
Fp1-Avg Ref
F3
Avg Ref
(average of all electrodes)
F3-Avg Ref
C3
Avg Ref
(average of all electrodes)
C3-Avg Ref

Figure 1.7: An illustration of the average referential montage [15]

1.3.5 Fundamental of EEG Waves

Since the beginnings of EEG, the study of different brain oscillations and their relationship with different brain functions has attracted the attention of researchers. The brain oscillations are categorized in frequency bands and related with different brain states or functions. In this section, a brief description of EEG frequency bands is provided. A typical example of EEG waves can be seen in Fig 1.8

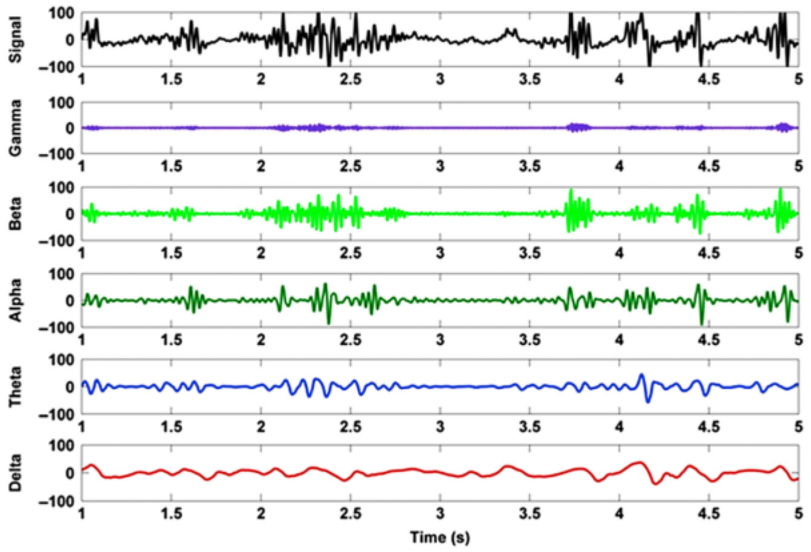


Figure 1.8: EEG signal and corresponding bands

Delta Waves (Up to 4 Hz)

EEG delta waves are high-amplitude brain waves and are associated with deep sleep stages. The delta waves are also associated with different brain functions other than deep sleep, e.g, high frontal delta waves in awake subjects are associated with cortical plasticity. Delta bands are reported as prominent brain waves in cognitive processing especially in event related studies.

Theta Waves (4-8 Hz)

Theta waves are observed in the drowsy state and more common in children than adults. In the awake adult, without doing any attention/cognitive activity , high theta activity is considered abnormal and associated with different brain disorders, e.g, high frontal theta is linked with non-response to antidepressant treatment in depression patients. However, high theta activity plays a significant role in intentional processing and working memory.

Alpha Waves (8-13 Hz)

Alpha waves can be observed spontaneously in normal adults during wakefulness and in relax state, especially when there is no mental activity. During the eye-closed condition, alpha waves are prominent at parietal locations. Attentional processing or cognitive tasks attenuate the alpha waves. Alpha waves are subdivided into lower alpha and upper alpha. It has been observed that alpha activity changes with load during retention of working memory. In addition, individual alpha peak frequency is an indicator of general intelligence factor.

Beta Waves (13-25 Hz)

Beta waves have lower amplitudes than alpha , delta and theta waves. Traditionally, beta waves are subdivided into low beta and high beta. The frontal and central regions of the brain are location where enhanced beta waves can be observed during activeness, anxious thinking, and deep concentration.

Gamma Waves (above 25 Hz)

Gamma waves are fast oscillations and are usually found during conscious perception. Due to small amplitude and high contamination by muscle artifacts, gamma waves are underestimated and not widely studied as compared to other slow brain waves. High gamma activity at temporal locations is associated with memory processes. Research studies reported that gamma activity is involved in attention ,working memory, and long term processes. Gamma activity is also involved in psychiatric disorders such as schizophrenia, hallucination, Alzheimer's disease, and epilepsy.

1.3.6 EEG Analysis Techniques

EEG analysis methods can be classified into the time domain methods, frequency domain methods, time-frequency domain methods, and linear or non-linear methods [16].

Time domain methods

EEG recordings are non-stationary and non-linear functions of time. Linear prediction is a time-domain method in which the output is calculated from the input and earlier outputs. *Principal component analysis* (PCA), *linear discriminant analysis* (LDA) and *independent component analysis* (ICA) are widely used unsupervised time-domain methods to summarize EEG data. PCA is used to transform the high-dimensional data to a low-dimensional data while ICA decomposes high-dimensional data into linear statistically independent components. In EEG data analysis, ICA is most used to remove artifacts. Whereas, LDA is used to reduce dimensions of feature sets by finding linear combinations of feature vectors.

Frequency domain methods

During an epileptic seizure, there is a sudden change in the frequency of EEG signals, which is measurable by applying frequency-domain methods, e.g, using Fourier Transform (FT). One can used either *parametric* or *non-parametric* methods to estimate the power spectrum using FT. Welch (a non-parametric) method, a modified version of widely used *periodogram* method, is generally used for the estimation of Power Spectrum Density (PSD). But this has a disadvantage of spectral leakage and is overcome by employing parametric methods. Parametric methods provide better frequency resolution by assuming the EEG signal is stationary random process. Moving Average (MA), Auto-regressive (AR) and Auto-Regressive Moving Average (ARMA) are commonly applied parametric methods.

Time-frequency domain methods

Above mentioned time-domain and frequency-domain methods have limitations of providing exact frequencies involved in a particular time instant and the information of time moment respectively. To overcome these limitations, Wavelet Transform (WT), a time-frequency based analysis technique, is widely used to obtain multi-resolution decomposed sub-band signals by passing the EEG signal through filter bank.

Non-linear methods

Non-linear analysis methods are applied to detect the coupling among harmonics in signal's spectrum. *High Order Spectra (HOS)*, various measures of entropy e.g, *Approximate Entropy (ApEn)*;

Kolmogorov entropy; *Sample entropy*, along with the *Hurst exponent*, *Largest Laypunov Exponent (LLE)* are widely used non-linear parameters for EEG analysis. Entropy and LLE are commonly used as features for epilepsy classification.

1.4 EEG and Epileptic Seizures

ES are related to the electrical activity of the brain. therefore, EEG is the most effective way to determine the epileptogenic cortex. Since EEG can detect any abnormal brain activity, it is mainly used for epilepsy diagnosis and seizure analysis. Figure 1.9 show the EEG signal of a patient that has experienced a seizure. It is evident that a seizure can be visually distinguished from a normal brain signal [17]

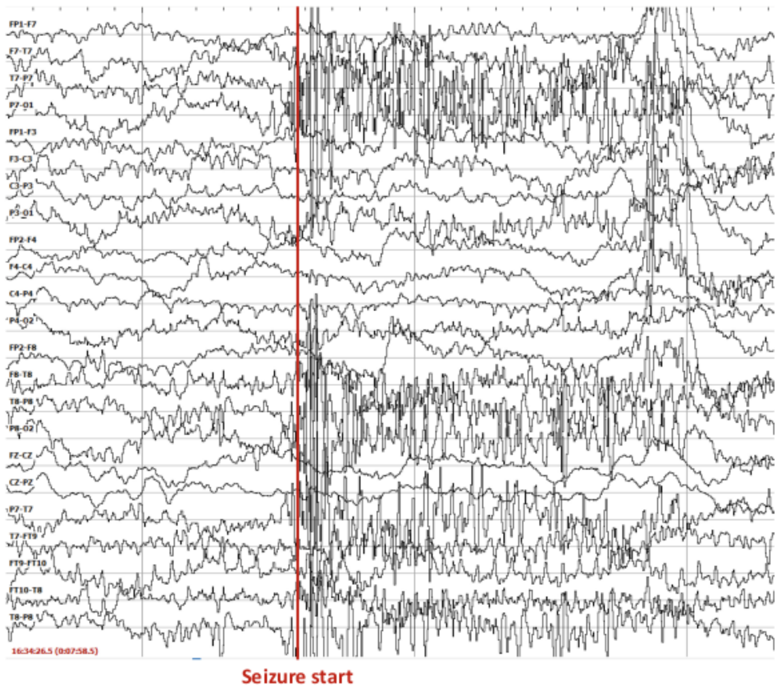


Figure 1.9: Example of normal and seizure EEG recording [17]

1.4.1 EEG in Epilepsy Diagnosis

Epilepsy may develop because of any abnormality in brain wiring, an imbalance of nerve signaling chemicals called neurotransmitters, or some combination of these factors. Figure 1.10 show an image of abnormal electrical impulses during a seizure. Neurons normally generate electrochemical impulses that act on other neurons, glands, and muscles to produce human thoughts, feelings and action. In epilepsy, the normal pattern of neuronal activity becomes disturbed, causing strange sensations, emotions and behaviours [18]. EEG is an essential component for the

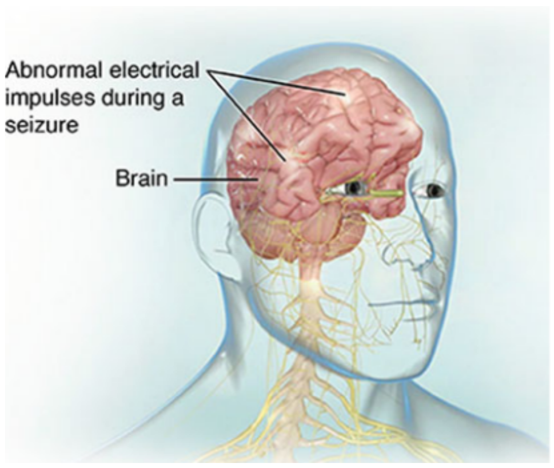


Figure 1.10: An illustration of abnormal electrical impulses during a seizure

diagnosis and analysis of epilepsy [19]. EEG continues to play a central research role in the diagnosis and management of patients with seizure disorders. One of the main reasons behind this is a convenient and relatively inexpensive way to demonstrate the physiological manifestations of abnormal cortical excitability that underlies epilepsy. The seizure can encompass a discrete part of the brain *partial* or the complete cerebral mass *generalized*.

1.4.2 EEG Signal Analysis for Epilepsy

The recorded EEG represents electrical activity produced by firing of neuron within the brain along the scalp. When epilepsy is present, seizure activity will appear as rapid spiking waves on the EEG. Epileptic activity can create clear abnormalities on a standard EEG. Epilepsy leaves its signature in the EEG signals [20]. EEG signals recorded, before and during a seizure, contains characteristics that can be used to identify the different stages of an epileptic seizure, and the pre- and post-seizure periods. These stages are briefly described bellow [21]:

1. **Pre-ictal State**: A pre-ictal state becoms apparent during a said time period before the occurrence of a seizure and does not occur at the rest of the times. It might not necessarily be visually apparent. However, it will reflect changes in the underlying signals and would be predictive of seizures within a specific range of values. For a pre-ictal state to be of use clinically in a warning system, it has to be detected early enough so that the time under false warning is minimised.

2. **Ictal and Interictal State**: The ictal state is a change in EEG signals during a seizure and interictal is the stage between two following seizure onsets. For the same person, the number of epileptogenic neurons, cortical region, and the span of seizure can be altered.

3. **Post-Ictal State**: This state is after the occurrence of a seizure.

The wave pattern may hold valuable information about brain activity. Experienced neurologists can detect disorders by visually observing the EEG signals. However, this procedure is time-consuming and is prone to faulty detection due to high temporal and spatial aspects of the dynamic non-linear EEG data. Therefore, computerized techniques, EEG signal parameters extraction, and analysis can be profoundly beneficial in the diagnostics.

1.5 Seizure Detection and Prediction

Seizure anticipation (or warning) can be classified into two broad categories:

- *Seizure detection*: in which the goal is to use EEG data to identify seizure onset, which typically occurs a few seconds in advance of the observed behavioral changes or during the period of early clinical manifestation of focal motor changes or loss of patient awareness.
- *Seizure prediction*: in which the aim is to detect preictal changes in the EEG signal that typically occur minutes to hours in advance of an impending epileptic seizure.

In seizure detection, since the aim of these algorithms is causally identify an ictal state, the statistical robustness of early detection algorithms is very high [22], [23]. The practical utility of these schemes in the development of an online seizure abatement strategy depends critically on the few seconds of time between the detection of an EEG seizure and its actual manifestation in patients in terms of behavioral changes.
In seizure prediction, the effectiveness of seizure prediction techniques tends to be lower in terms of statistical robustness. This is because the time horizon of these methods ranges from minutes to hours in advance of an impending seizure and because the preictal state is not well-defined state across multiple seizures and across different patients.

1.6 Problem Formulation

The EEG can be conceptualized as a series of numerical values (voltages) over time and space (gathered from multiple electrodes). Such a series is called *a multivariate time series*. Before introducing the problem formulation, let us go through some formal definitions [24].

Definition 1 *A univariate time series $X = [x_1, x_2, ..., x_T]$ is an ordered set of real values. The length of X is equal to the number of real values T.*

Definition 2 *An M-dimensional multivariate time series, $X = [X^1, X^2, ..., X^M]$ consists of M different univariate time series with $X^i \in \mathbb{R}^T$*

Then a seizure prediction methodology can be defined as follows, a preictal duration is pre-selected to categorize EEG signals in preictal and interictal states and a *binary classifier* is trained to exploit differences between the two states. The ictal segments cannot contribute to seizure prediction and are discarded from the analysis; including postictal segments, if they are considered as a separate state. Therefore the problem can be treated as a Time Series Classification (TSC). Let us see another formal definition [24]:

Definition 3 *A dataset* $D = (X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)$ *is a collection of pairs* (X_i, Y_i) *where* X_i *could either be a univariate or multivariate time series with* Y_i *as its corresponding one-hot label vector. For a dataset containing K classes, the one-hot label vector* Y_i *is a vector of length K where each element* $j \in [1, K]$ *is equal to 1 if the class of* X_i *is j and* 0 *otherwise.*

The task of TSC in our case consists of training a classifier (binary classifier) on a dataset D in order to map the space of possible inputs (preictal or interictal segments) to a probability distribution over the class variable values (preictal or interictal states). A general framework for TSC is depicted in Fig 1.11, where K is equal to 2 in our problem.

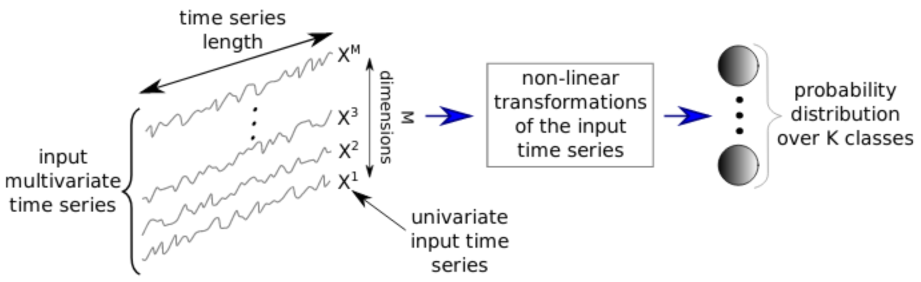

time series length
X^M
X^3
X^2
X^1
M dimensions
input multivariate time series
univariate input time series
non-linear transformations of the input time series
probability distribution over K classes

Figure 1.11: A unified framework for time series classification [24]

As the final goal of a seizure prediction system is to raise an alarm in advance of the seizure onset, it is not trivial to show how this system should be evaluated. A perfect prediction method indicates the exact point in time when a seizure occurs. In practical applications, a predictor anticipates a duration of the high probability of occurrence of seizure. The uncertainty can be considered by use of the Seizure Occurrence Period (SOP), which defined as a time duration in which there is a possibility of seizure. In addition, to permit intervention, a minimum window of time between the alarm raised by the prediction method and the beginning of SOP is essential. This window is called the Seizure Prediction Horizon (SPH). Note that the definition of SOP and SPH used in this work was proposed in [25] as these two parameters are not unifiedly defined in the literature. In Fig 1.12 the concept of SPH and SOP is illustrated.

For a correct prediction, a seizure onset must be is after the SPH and within the SOP. Likewise, a false alarm rises is when the prediction system returns a positive but there is no seizure occurring during SOP. When an alarm rises, it will last until the end of the SOP. Regarding clinical use,

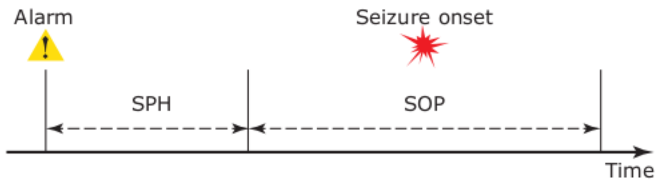


Figure 1.12: Definition of a seizure occurrence period (SOP) and seizure prediction horizon (SPH)

SPH must be long enough to allow sufficient intervention or precautions. SPH is also called intervention time. In contrast, SOP should be not too long to reduce the patient's anxiety.
To recap, the epileptic seizure prediction problem consists of a binary classifier which assigns a preictal/interictal label for each segment of EEG signals, and that given SPH and SOP. Thus, we need to set a sufficient performance and quality check to compare different techniques. Then the evaluation can be done for both *segment-based* and *event-based*:

Segment-based evaluation The classification given to a particular EEG segment can be placed into one of four categories, where the positive class is attributed to the preictal state and the negative class to the interictal state :

- True Positive (TP): an algorithm correctly classifies a preictal segment of an EEG as being in the preictal state
- True Negative (TN): an algorithm correctly classifies an interictal segment of an EEG as being in the interictal state
- False Positive (FP): an algorithm incorrectly classifies an interictal segment of an EEG as being in the preictal state
- False Negative (FN): an algorithm incorrectly classifies a preictal segment of an EEG as being in the interictal state

To estimate the prediction performance based on the definition above, the following metrics can be defined as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{1.1}$$

$$Specificity = \frac{TN}{TN + FP} \tag{1.2}$$

Event-based evaluation Osorio et al. proposed sensitivity and False Positive Rate (FPR) as performance parameters of ES predictors [26]. Sensitivity is defined as the number of successfully predicted seizures by detecting at least one preictal EEG segment and FPR declares the false alarm generation rate as the number of false positives per hour of EEG recordings.

1.7 Machine Learning for Epileptic Seizure Prediction

Machine Learning (ML) has revolutionized the seizure prediction field offering tools to address the high complexity of EEG signals and allow multivariate analysis and feature spaces of higher order to be easily evaluated to distinguish hidden preictal characteristics. Figure 1.13 shows the classical ML methodology for the epilepsy prediction and also highlight the major difference between the use of ML and DL technique. Basically one can give the raw data or minimally processed data (i.e., without extraction of features from the raw data) to a DL model for pattern learning [27]

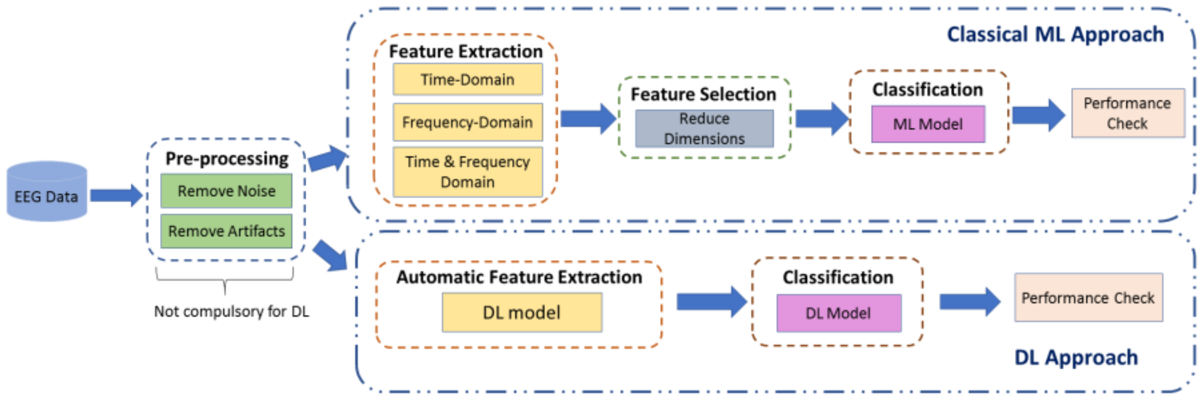


Figure 1.13: Process of epilepsy prediction using EEG data and classification algorithm [27]

Signal Processing Noise and artifact identification is a crucial procedure in raw biomedical signals. To reduce the influence of these artifacts in feature extraction, filtering of these artifacts is needed. Multiple techniques have been employed for filtering e.g. band-pass filter, wavelet filter, finite impulse response filter, and adaptive filter. This processing is also performed to normalize the data to make it comparable with the recording of other patients. There are also many data dropouts or corrupted data in the EEG recording due to limitations of implanted electrodes which lead to the insignificant performance of algorithms. Due to muscle artifacts and environmental noise, there also exist some outliers in data. The presence of these outliers badly influences the extracted features.

Feature Extraction and Selection All prediction models need reliable features, well correlated with preictal and interictal stages. One can categorize these features based on the number of EEG channels as *univariate* (measures taken on each EEG channel separately) and *multivariate* (measures taken on two or more EEG channels) features. Further categorization of each of these is as linear or nonlinear features. Florian et al. compared the performance of univariate and bivariate measures containing both linear and non-linear strategies for ES prediction [28]. They noted that while using univariate measures, preictal variations transpired 5-30 min before ES onset. While bivariate measures performed better by capturing preictal changes at least 240 min before an ES onset. Figure 1.14 shows some of the linear and nonlinear measures used in the literature for ES prediction. Linear measures performed better or sometimes similar to nonlinear measures.

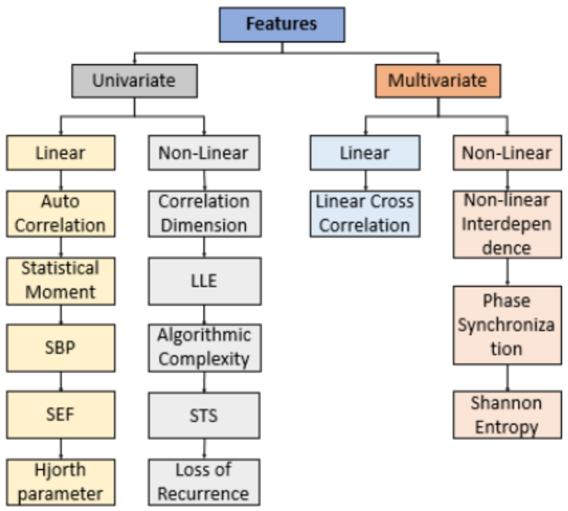


Figure 1.14: Feature classification based on number of channels of EEG data

Classification Identification of preictal and interictal patterns from EEG data is carried out using ML algorithms,e.g, Artificial Neural Network (ANN), k-means clustering, decision trees, Support Vector Machine (SVM), and fuzzy logic. Mostly threshold-based on features values are utilized to make conclusions. However, ML-based studies broadly focused on the extraction of optimized features for prediction.

1.8 Deep Learning Basics

Since we are not interested in using classical machine learning algorithms for this problem, we will discuss the basics of deep learning models to get an intuition about how they work. Deep learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain. It takes the advantage of learning feature hierarchies where features from higher levels of the hierarchy formed by the composition of lower level features. Its architecture is composed of multiple processing layers that allows it to learn representations of data with multiple levels of abstraction.

1.8.1 Artificial Neural Networks

Artificial neural networks, or simply neural networks (NN), are a type of machine learning models that are inspired by human neural networks. The structure of NN is similar to the one of our brain, which has neurons and connections between them to transmit electrical impulses. Each neuron is optimized to receive information from other neurons, process this information in a unique way, and send its result to others cells. Relying on the functional behavior of the biological neuron, we can define a computing unit that is referred to as the artificial neuron. The artificial neuron is the basic unit of what is called neural networks. Mathematically, it can be represented as a function that takes some number of inputs $x_1, x_2, ..., x_m$, each of these inputs have a corresponding weight $w_1, w_2, ..., w_m$. The weighted sum of inputs plus the bias term form the linear part of the transformation producing z, followed by the non-linear part producing the output of the neuron y, which is the application of the activation function. The purpose of bias is to actually shift your activation function to the left and to the right regardless of the inputs to better fit the data.

$$z = \sum_{i=1}^{m} w_i x_i + b \tag{1.3}$$

$$y = f(z) \tag{1.4}$$

Figure 1.15 illustrates the concept of a single neuron. Many activation functions have been proposed in the literature and will be discussed in next paragraphs. Note that in some cases, the activation function may be linear (e.g. regression problems). Using linear algebra, we can rewrite the two equation in a simple way. We can write the inputs $x_1, x_2, ..., x_m$ in a vector X and the weight $w_1, w_2, ..., w_m$ in a vector W . To compute the output, we have to take the dot product of the two vectors and then apply the non-linearity.

$$y = f(w^T x + b) \tag{1.5}$$

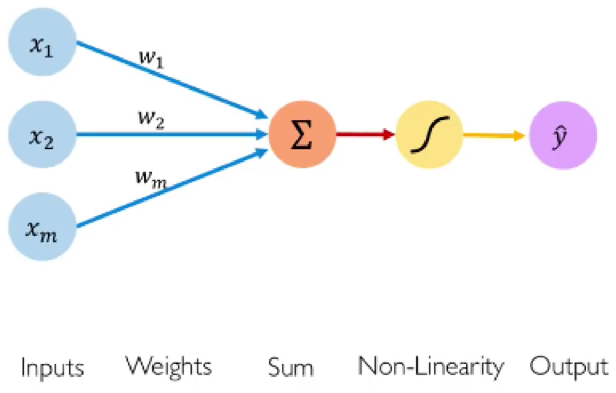


Figure 1.15: Schematic for an artificial neuron

We can now move to understand the neural network architecture, as the idea of a neuron is basic but essential. A simplified version of neural networks is represented as a layered organization of neurons with connections to others neurons. The input layer is represented by the input data, the output layer is the last one that process data and returns results of the entire model, while all the layers in between are called hidden layers. These hidden layers transform the data from a layer to another and send the processed information to the next layer. They are what makes the network deep. This characteristic of the network is what popularised the term "Deep Learning" which is the field of study associated with everything related to ANNs. The following diagram 1.16 represents a typical network's structure.

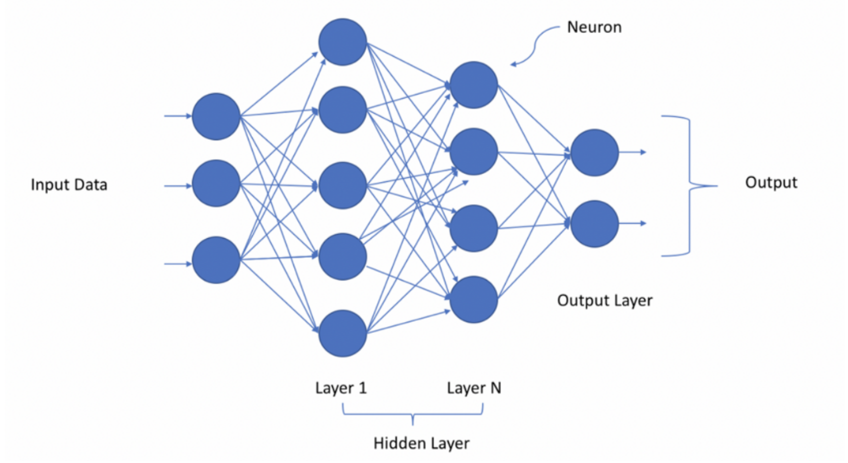


Figure 1.16: Neural Network with N hidden layers [29]

1.8.2 Activation functions

As discussed above, the activation function is one the building blocks of neural networks. We know that it the second transformation of each neuron after the linear part and it is a non-linear function in order to help the model to learn complex patterns. If we use linear activation functions in a deep neural network, no matter how deep our network is, it will be equivalent to just a linear neural network with no hidden layers because those linear activation functions can be combined to form another single linear function. In other words, A neural network without an activation function is essentially just a linear regression model. Thus we use a non linear transformation to the inputs of the neuron and this non-linearity in the network is introduced by an activation function. In this section we will see the most common activation function:

Sigmoid function It is one of the most widely used non-linear activation functions. It has been used for predicting probability based output and has been applied successfully in binary classification problems. It is defined by :

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.6}$$

The sigmoid function translate the input in range $[-\infty + \infty]$ to the range $[0, 1]$. The function is monotonically increasing and continuously differentiable. For small values of x, it is closer to 0. Conversely, for the bigger is x, the closer is to one.

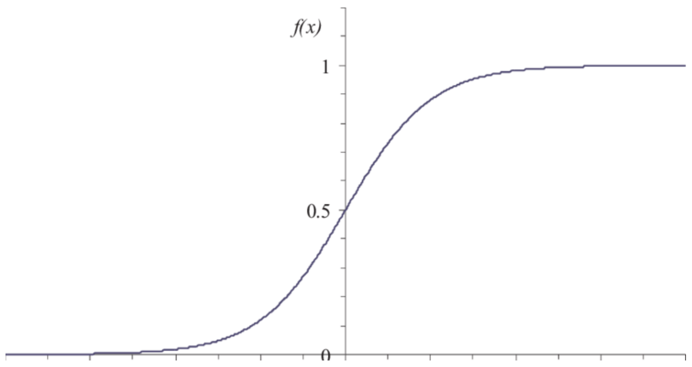


Figure 1.17: Graph of the Sigmoid function .

Tanh Tanh function uses a similar kind of S-shaped non-linearity like the sigmoid function, but instead of ranging from 0 to 1, the output of tanh ranges from -1 to 1. It is defined by the following equation:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.7}$$

The graph of this function is represented in Figure 1.18.

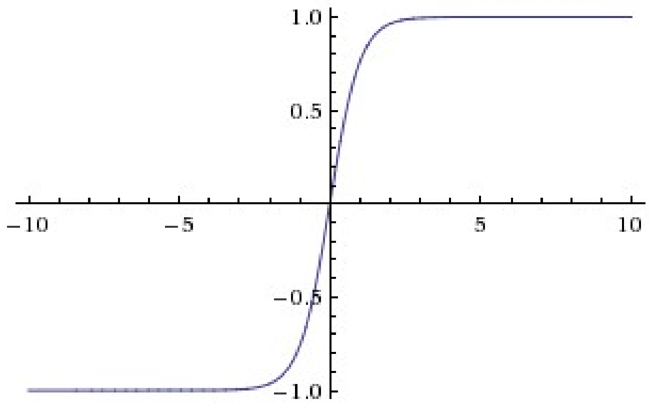


Figure 1.18: Graph of the *tanh* function.

ReLU The ReLU (Rectified Linear Unit) activation function performs a threshold operation to each input element where values less than zero are set to zero and returns the identity output otherwise.

$$f(x) = max(0, x) \tag{1.8}$$

The main advantage of ReLU is the computation power is faster since it does not calculate exponents and divisions, which makes it the neuron of choice for many tasks. Figure 1.19 shows the output of a ReLU neuron as x varies:

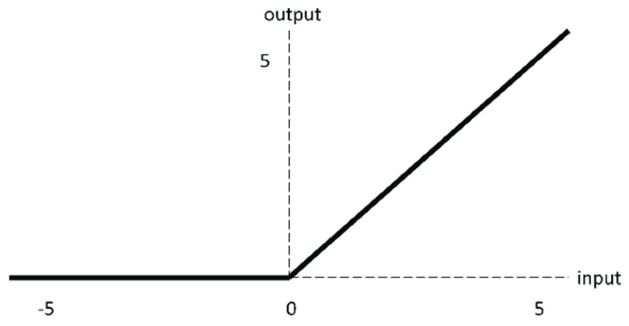


Figure 1.19: Graph of the ReLU function.

Softmax activation function The Softmax activation function outputs a vector representing a probability distribution over a set of mutually exclusive labels. It differs from other activation functions that we have seen before because the output of each individual neuron in the layer associated with it depends on the output of the rest of the neurons of the same layer. This dependency is the result of the law of probability that requires the sum of all outputs to be equal to 1. This is achieved by normalizing z and setting the output of the neuron to:

$$y_{oi} = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{1.9}$$

A good prediction would have one of the entries of the output vector close to 1, whereas the remaining entries are close to 0. Contrariwise, a bad prediction would have multiple possible labels roughly equal to the same value. Therefore, Softmax is used specifically for the output layer of the network.

1.8.3 Feed-forward Neural Networks

Deep feed-forward networks , also called feed-forward neural networks, or dense neural networks, are the core of deep learning models. The goal of a feed-forward network is to approximate some function f. It defines a mapping $y = f(x; \omega)$ and learns the value of parameters ω that result in the best function approximation. They are called feed-forward because information only travels forward in the network (no loops), first through the input nodes, then through the hidden nodes (if present), and finally through the output nodes. The network is also called *fully-connected* because the output of each neuron in one layer is sent as input of each neuron in the next layer, so there is a direct connection between all the neurons belonging to subsequent layers. Figure 1.20 shows an example of a simple version of a feed-forward neural network with one hidden layer, making visible the full interconnection between subsequent layers.

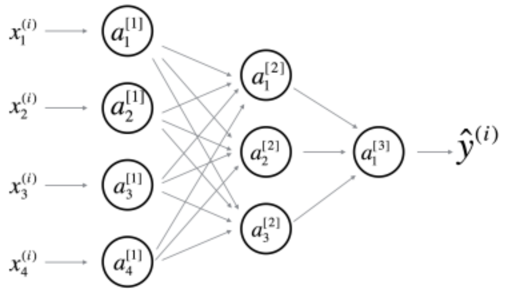


Figure 1.20: Example of dense neural network

In common notations, superscript $[l]$ denotes the l^{th} layer and subscript (k) or (j) denotes the k^{th} node (neuron) in a layer. Thus, between the j^{th} neuron in the l^{th} layer and the k^{th} neuron in the $(l-1)^{th}$ layer, there is a connection that has a weight of $w_{j,k}^{[l]}$ associated with it. The general activation formula for each neuron is bellow, where $f^{[l]}$ is the activation function of the l^{th} layer:

$$a_j^{[l]} = f^{[l]}(\sum_k w_{j,k}^{[l]} a_k^{[l-1]} + b_j^{[l]}) = f^{[l]}(z_j^{[l]}) \tag{1.10}$$

Following these notations and using a vectorized implementation, the feed-forward steps for the network represented in Fig 1.20 are the following equations (x is the input vector):

$$a^{[0]} = x \tag{1.11}$$
$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]} \tag{1.12}$$
$$a^{[1]} = f^{[1]}(z^{[1]}) \tag{1.13}$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \tag{1.14}$$
$$a^{[2]} = f^{[2]}(z^{[2]}) \tag{1.15}$$

1.8.4 Cost Function

Given a data example formed by the pair $(x^{(i)}, y^{(i)})$ where $x^{(i)}$ is the network's input and $y^{(i)}$ is the desired output, and suppose some values (for the moment) for weights and biases (w,b), we can get the forward step using the equations above. The output of the network is the activation of the last layer and it is called the *predicted* value of the model , denoted as $\hat{y}$ or o for output. To train the parameters w and b, we need to define a *cost function* as our goal is to get $\hat{y}^{(i)}$ equal to $y^{(i)}$.
The *loss function* or *error function* measures the discrepancy between the prediction $\hat{y}^{(i)}$ and the desired output $\hat{y}^{(i)}$. Loss function is a method of evaluating "how well your algorithm models your example". If your prediction is far from the desired output, your loss function will output a higher number. Conversely, if it is pretty good, you will get a lower number. Whereas, the cost function is the average of the lost function of the entire training set.

> "The cost function reduces all the various good and bad aspects of a possibly complex system down to a single number, a scalar value, which allows candidate solutions to be ranked and compared[30]."

Generally, cost and loss functions are synonymous and can be used interchangeably, however the cost function can contain regularization terms (see subsection 1.8.7) in addition to loss function.

> "The function we want to minimize or maximize is called the objective function or criterion. When we are minimizing it, we may also call it the cost function, loss function, or error function [31]."

Neural networks are trained using an optimization process that requires a cost function to calculate the model error. Many types have been proposed for different models including regression problems and classification problems (binary classification or multi-class classification).

Regression Problem A regression problem is a model that predicts a real-value quantity, where the output layer is composed of one node with a linear activation unit. The very common cost function for this type is the Mean Square Error (MSE). It is defined as follows, where m is the number of training examples:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2 \tag{1.16}$$

Binary Classification Problem In this case, the model is expected to classify an example as belonging to one of two classes. The model's output layer consists of one node with a sigmoid activation unit. The cost function is called *cross-entropy*, and is defined as follows:

$$\begin{aligned} J(\omega, b) &= \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \qquad (1.17)\\ &= \frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(\hat{y}^{(i)}) + (1 - y^{(i)})log(1 - \hat{y}^{(i)})] \qquad (1.18)\end{aligned}$$

Multi-class Classification Problem This is a problem where the model classifies an example as belonging to one of more than two classes. The last layer is configured such as one node for each class using the softmax activation function. It is just the extension of binary classification problem. For most time, a multi-class cross-entropy loss is used as a cost function for this type. We denote by C the number of classes, which the length of the one-hot encoded output vector y:

$$L(\hat{y}^{(i)}, y^{(i)}) = -\sum_{j=1}^{C} y_j^{(i)} log\hat{y}_j^{(i)} \tag{1.19}$$

$$J(\omega, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) \tag{1.20}$$

1.8.5 Optimizers

We have seen in the last section that the cost function is a measure of the difference between the actual output and the predicted one. Then our goal is to minimize the cost function by finding the optimized value for weights and biases. The minimization process is insured by an optimization algorithm. Optimization is a big part of machine learning. Almost every machine learning algorithm has an optimization algorithm at it's core.
For Optimization is an iterative process where parameters are updated at each step in order to minimize the cost function. Many optimizers have been developed for deep learning's training such as Stochastic Gradient Descent [32], [33], [34], Adagrad Gradient Descent [35]. For this project, the optimizer that has been used for all the models is the *Adaptive Moment Estimation*, known as the *Adam* optimizer [36].

Adam Optimizer Adam is one of the most popular and used optimizers in deep learning. It is an extension to the well-known *stochastic gradient descent* algorithm. The algorithm was presented in 2015 in a paper titled ”Adam: A Method for Stochastic Optimization”. It is based on two other extensions of stochastic gradient descent:

- **Adaptive Gradient Algorithm (AdaGrad)** is an adaptive learning rate method. In Adagrad we adopt the learning rate to the parameters. We perform larger updates for infrequent parameters and smaller updates for frequent parameters.

- **Root Mean Square Propagation (RMSprop)** tries to resolve Adagrad's radically diminishing learning rates by using a moving average of the squared gradient. It utilizes the magnitude of the recent gradient descents to normalize the gradient.

Adam realizes the benefits of both AdaGrad and RMSProp. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients. Therefore, this optimizer stores two moments: the first moment m_t (the mean), which is estimated as the exponentially decaying average of past gradients, and the second moment v_t, which is estimated as the exponentially decaying average of past squared gradients. They are computed as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{1.21}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{1.22}$$

where g_t is the gradient, t the timestep and (β_1, β_2) are optimizer's decay rates. Since m_t and v_t are initialized as zero-vectors, we need to set a bias-correction:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{1.23}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{1.24}$$

Finally, the Adam update rule for model's weights is the following formula:

$$\theta_t = \theta_{t-1} - \alpha . \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{1.25}$$

where θ is the model parameters (weights and biases) and α is the learning rate. According to the paper, the good default settings are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.
Sebastian Ruder publishes a review of different variations of gradient descent optimization algorithms [37]. In his section ”Which optimizer to use ?”, he recommends using Adam:

Insofar, RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances. [. . .] its bias-correction helps Adam slightly outperform RM-

Sprop towards the end of optimization as gradients become sparser. Insofar, Adam might be the best overall choice [37].

1.8.6 Neural Network's training process

During this learning phase, the network learns by adjusting the parameters so as to be able to predict the correct class label of input samples. The model parameters, in this case, refer to the weights and biases of each layer. First of all, the parameters are initialized. Many initialization schemes have been proposed in the literature [38] [39]. One common initializer that is used in this project is called *Xavier Initializer* [40]. Then, the feed-forward neural network is used to accept an input x and produce an output $\hat{y}$, information flows forward through the network as explained in section (1.9.3). The input x provides the initial information that then propagates up to the hidden units at each layer and finally produces $\hat{y}$. This is what is called *forward propagation*. With these outputs $\hat{y}$, and knowing the actual value of the output y, the cost function can be computed as described in section (1.9.4), which produces a scalar cost $J(\omega, b)$. After that, the error measured by the cost function is propagated through each layer in reverse order to compute the error contribution from each neuron's connection, until it reaches the input layer. This step allows the information from the cost to then flow backward through the network to compute the gradient. This phase is called the *back-propagation algorithm*, often simply called *backprop*. The model parameters are then updated by an optimizer algorithm such as Adam described above, using the gradients.

> Actually, back-propagation refers only to the method for computing the gradient, while another algorithm, such as stochastic gradient descent, is used to perform learning using this gradient [31].

When all the weights and biases have been successfully updated, the whole process is repeated for another incoming batch, until all the training dataset has been entirely traversed, in which case, one epoch of training is achieved. The back-propagation algorithm represents the core of the learning process of a neural network, since it is the procedure to update the model's learnable parameters. It uses the *Chain Rule* of calculus to compute the derivatives of the cost function with respect to different neurons of each layer in an efficient way.
Training a neural network with back-propagation involves choosing a number of components and hyperparameters:

- **Cost function** The function used to estimate the performance of a model with a specific set of weights on examples from the training dataset.
- **Weight Initialization** The procedure by which the initial small random values are assigned to model weights at the beginning of the training process.

- **Batch size** The number of examples used to estimate the error gradient before updating the model parameters.
- **Learning Rate** The amount that each model parameter is updated per cycle of the learning algorithm.
- **Epochs** The number of complete passes through the training dataset before the training process is terminated.

1.8.7 Regularization

The central challenge in ML as well as DL is that our algorithm must perform well on *new*, previously *unseen* inputs, not just those on which the model was trained. The ability to perform well on unobserved inputs is called *generalization*. In the evaluation process of a machine learning model, first we have access to the *training error*, the error measured on the training set. We try to minimize this error as possible using an optimization algorithm. However, what separates machine learning from optimization is that we want the *generalization error*, also called the *test error*, to be low as well. Deep learning models generally have a huge amount of parameters that leads sometimes to the overfitting of training set. Overfitting occurs when the gap between the training error and test error is too large. To solve this problem, there exist several techniques called *regularization methods* [41]. The regularization techniques used in this project are the dropout and the early stopping.

Dropout Dropout is probably the most common regularization technique in the deep learning era. According to Wikipedia [42], the term "dropout" refers to dropping out units (both hidden and visible) in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections. Thus, this unit is not considered during a particular forward and backward pass [43].
Dropout regularization is a computationally cheap and a simple way to regularize a deep neural network. During the training process, we go through each of the layers of the NN and set some probability p of eliminating a node. The probability p represents the dropout rate, that is the fraction of nodes to ignore. Figure 1.21 shows a dropout neural network model. The intuition behind dropout is that the model can't rely on any one feature, because if an essential neuron is ignored, then it is not able to perform well. Thus, the model has to spread out weights. Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. This technique randomly knocks out units in your network, so it is as if on every iteration you are working with a smaller network which is less prone to overfitting.

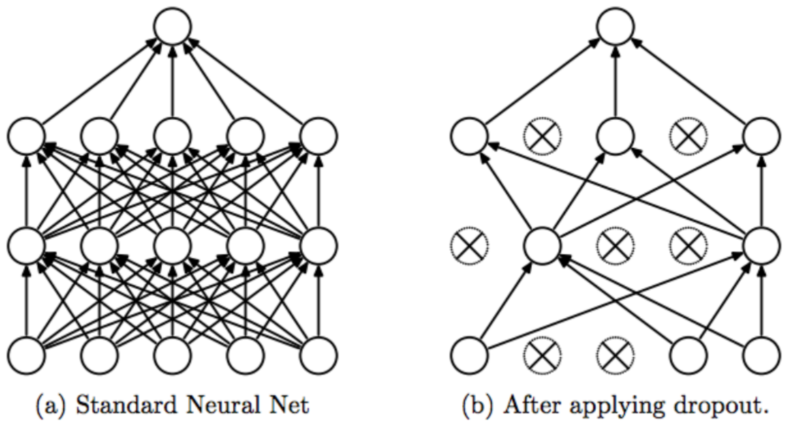


Figure 1.21: Illustrated example of dropout [43]

Early Stopping The number of epochs is a training's parameter that should be fixed as mentioned is section (1.9.6). Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that allows you to specify an arbitrarily large number of training epochs and stop training once the model performance stops improving on data outside of the training set. Figure 1.22 illustrates this behaviour.

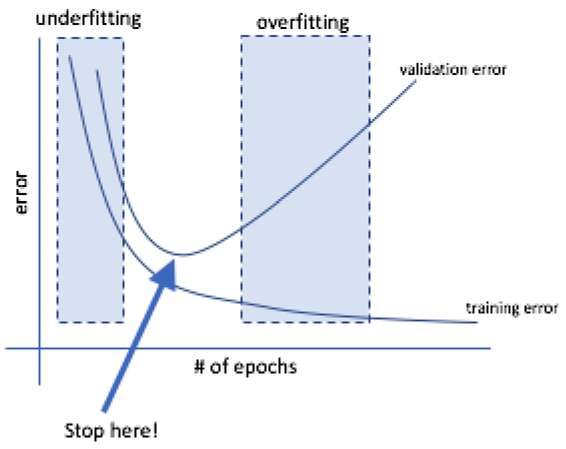


Figure 1.22: Illustration of early stopping

This means we can obtain a model with better validation set error (and thus, hopefully better test set error) by returning to the parameter setting at the point in time with the lowest validation set error [44].

1.8.8 Convolutional Neural Networks

Convolutional Neural Network (CNN), also known as ConvNet, are a class of deep, feed-forward (not recurrent) artificial neural networks that are applied to analyzing data that has a known grid-like topology. Examples include time series in 1-D, image data in 2-D, video data in 3-D. In 1995, the concept of Convolutional Neural Networks was introduced by Yann LeCun et al. [45]. The name "convolutional neural network" indicates that the network employs a mathematical operation called *convolution*. CNN have been demonstrated very high performance on complex computer vision and natural language processing tasks. They work very well on visual problems such as image classification [46], object detection [47], [48], neural style transfer [49], [50]. The CNN building blocks are convolution layer, pooling layer and fully-connected layer.

Convolution layer Convolutional layer is the major building blocks used in convolutional neural networks. It is used fundamentally for feature extraction. It is generally computed using a linear part (convolution operation) followed by an activation function (non-linear part). A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a *feature map*, indicating the locations and strength of a detected feature in an input, such as an image.

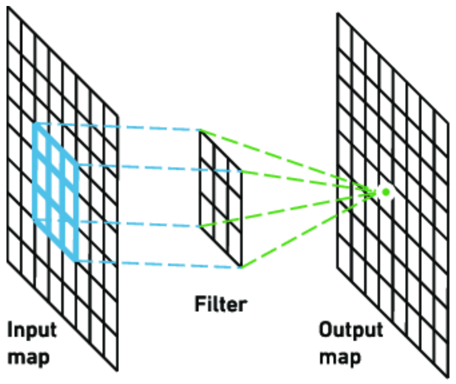


Figure 1.23: An illustration of a convolution operation

Pooling Layer Another important element of a convolutional neural network is the pooling layer. A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. The goal of a pooling layer is to subsample the input in order to reduce its dimensionality. This has the effect of making the resulting down sampled feature maps more robust to changes in the position of the feature in the image, referred to by the technical phrase *local translation invariance*. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively. Figure 1.24 depicts an example of a matrix that get down-sampled by pooling its elements using either max pooling or average pooling:

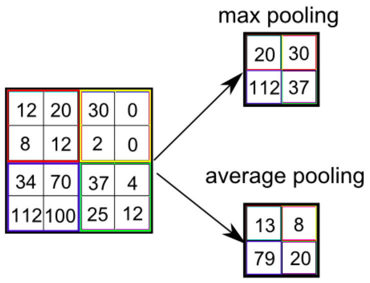


Figure 1.24: Top-right: max pooling, bottom-right: average pooling.

Fully-connected Layer This layer forms the last block of the CNN architecture. This is essentially a Fully connected Simple Neural Network, consisting of two or three hidden layers and an output layer. It can be viewed as the final learning phase, which maps extracted visual features to desired outputs. fully-connected layers are used to create final non-linear combinations of features and for making predictions by the network. It is usually adaptive to classification/encoding tasks. Also, it is common to pass this layer through softmax to represent the confidence of classification.

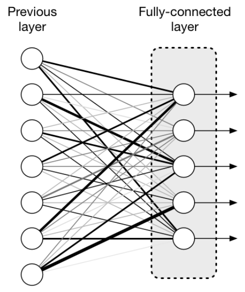


Figure 1.25: An illustration of a fully-connected layer

Batch Normalization Batch normalization is one of the reasons why deep learning has made such outstanding progress in recent years. It has the effect of dramatically accelerating the training process of a neural network, and in some cases improves the performance of the model via a modest regularization effect. It solves a problem called *internal covariate shift*, which occurs when there is a change in the input distribution of feature maps. This can cause the learning algorithm to forever chase a moving target. Sergy Ioffe et al. [51] proposed this technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch, which has the effect of stabilizing the training process and reducing the number of learning epochs. During training time, the batch normalization layer performs the following:

1. Calculate the batch mean μ_B and batch variance σ_B^2 of the input tensor with m elements:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

2. Normalize the input tensor's elements using the previously calculated batch statistics:

$$\bar{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

   where ϵ is a small constant used to avoid division by zero (usually 10^{-5})

3. The output tensor elements of the layer are shifted and scaled versions of the input tensor elements:

$$y_i = \gamma \bar{x}_i + \beta$$

   γ and β are learned during training along with the other parameters of the model.

Putting it together Now that we know about the building blocks discussed in the above subsections, we can hang them together to form a simple convolutional network. The way the layers are connected to each other differs from a model to another depending on its use. Very common architecture that is used for image classification task is the following, the input image go through a series of convolutional layers, pooling layers, fully connected layers, and in the end, a softmax function is applied to classify an object by delivering a probability vector. Figure 1.26 illustrates a typical CNN architecture designed for images classification. It comprises a feature extraction block composed of convolutional layers and a classification block that contains fully connected layers.

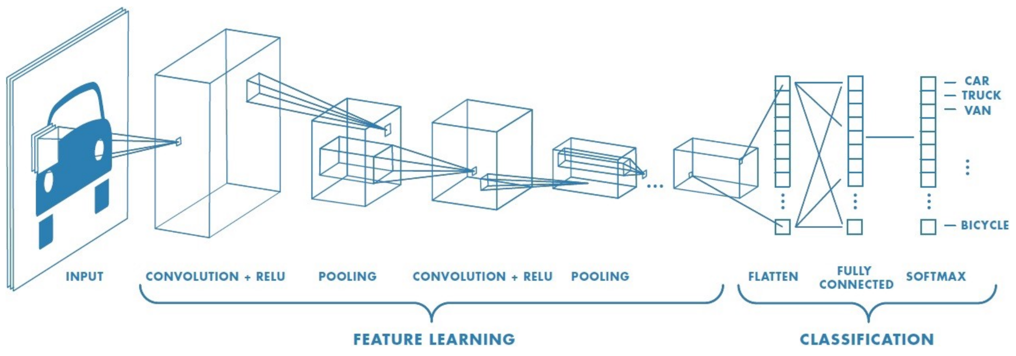


Figure 1.26: Convolutional neural network architecture .

1.8.9 Recurrent Neural Network

Recurrent Neural Network (RNN) are a type of neural networks for processing sequential data. They are called *recurrent* not *feed-forward* because the outputs from previous time steps are fed as input to the current time step. RNN have the advantage of processing sequences of variable length. Many complex problems have been solved using RNN such as machine translation [52], speech recognition [53], image captioning [54] and music generation [55].
RNN are very effective for time series classification and other sequence tasks because they have "memory". They can read inputs $x^{\langle t \rangle}$ (such as words) one at a time, and remember some information/context through the hidden layer activations that get passed from one time-step to the next. This allows a *unidirectional* RNN to take information from the past to process later inputs. A *bidirectional* RNN can take context from both the past and the future. Figure 1.27 illustrates the forward propagation for a simple recurrent neural network. At each time t, each neuron receives both the input $x^{\langle t \rangle}$ as well as the entire layer's output $y^{\langle t-1 \rangle}$. In this example, the RNN model has equal length for both inputs and outputs $T_x = T_y$, which is not necessary.

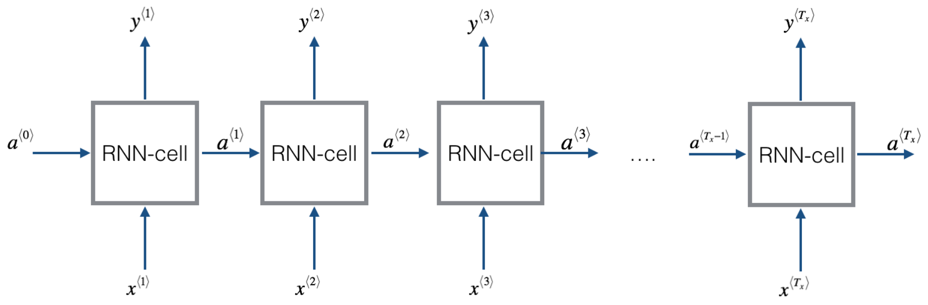


Figure 1.27: Basic RNN model

One of the most relevant features of recurrent neural networks is the the ability to handle arbitrary length inputs and outputs. This flexibility allows us to define a large variety of architectures to solve different problems. Figure 1.30 shows general architectures used for various sequence learning tasks. Note that for simplicity, we will continue presenting RNN models where $T_x = T_y$.

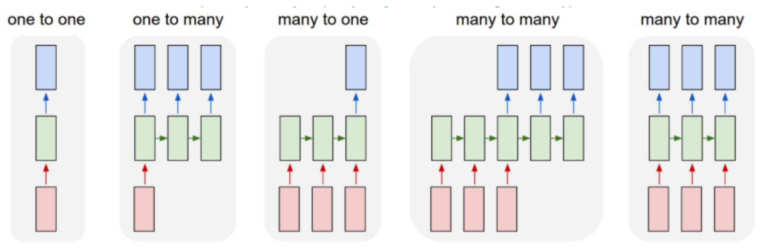


Figure 1.28: Different types of RNNs

RNN Cell A recurrent neural network can be seen as the repeated use of a single cell. Figure 1.29 describes the operations for a single time-step of an RNN cell. The activation $a^{\langle t \rangle}$ that is passed to the RNN from one time step to another is called a *hidden state*. Each cell takes two inputs at each time step:

- $a^{\langle t-1 \rangle}$ the hidden state from the previous cell.
- $x^{\langle t \rangle}$: The current time-step's input data.

It has two outputs at each time step:

- A hidden state ($a^{\langle t \rangle}$)
- A prediction ($y^{\langle t \rangle}$)

The hidden state is computed with tanh activation using the following equation, where W_{aa} and W_{ax} are weight matrices, and b_a is a bias vector:

$$a^{\langle t \rangle} = \tanh(W_{aa}a^{\langle t-1 \rangle} + W_{ax}x^{\langle t \rangle} + b_a) \tag{1.26}$$

The new hidden state $a^{\langle t \rangle}$ can be used for different connections depending on the model architecture. In this case, it is used to compute the output prediction as follows, where W_{ya} is a weight matrix and b_y is a bias vector:

$$\hat{y}^{\langle t \rangle} = softmax(W_{ya}a^{\langle t \rangle} + b_y) \tag{1.27}$$

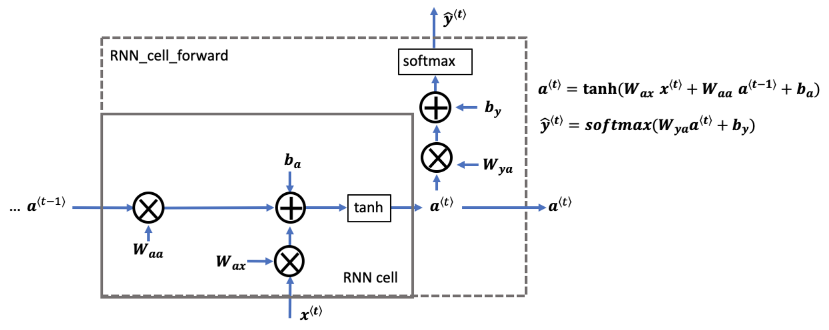

$\hat{y}^{\langle t \rangle}$
RNN_cell_forward
softmax
b_y
W_{ya}
$a^{\langle t \rangle} = \tanh(W_{ax}\,x^{\langle t \rangle} + W_{aa}\,a^{\langle t-1 \rangle} + b_a)$
$\hat{y}^{\langle t \rangle} = softmax(W_{ya}a^{\langle t \rangle} + b_y)$
b_a
... $a^{\langle t-1 \rangle}$
tanh
$a^{\langle t \rangle}$
$a^{\langle t \rangle}$
W_{aa}
W_{ax}
RNN cell
$x^{\langle t \rangle}$

Figure 1.29: Basic RNN cell

RNN Network A recurrent neural network is a repetition of the RNN cell that we have just discussed. If our input sequence of data is 10 time steps long, then we will re-use the RNN cell 10 times. A very important characteristic of RNN models is that the weights and biases

$(W_{aa}, b_a, W_{ax}, b_x)$ are re-used each time step. Figure 1.30 depicts an illustration of a basic RNN models, where the input sequence $x = (x^{\langle 1 \rangle}, x^{\langle 2 \rangle}, ..., x^{\langle T_x \rangle})$ is carried over T_x time steps. The network outputs $y = (y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, ..., y^{\langle T_x \rangle})$.

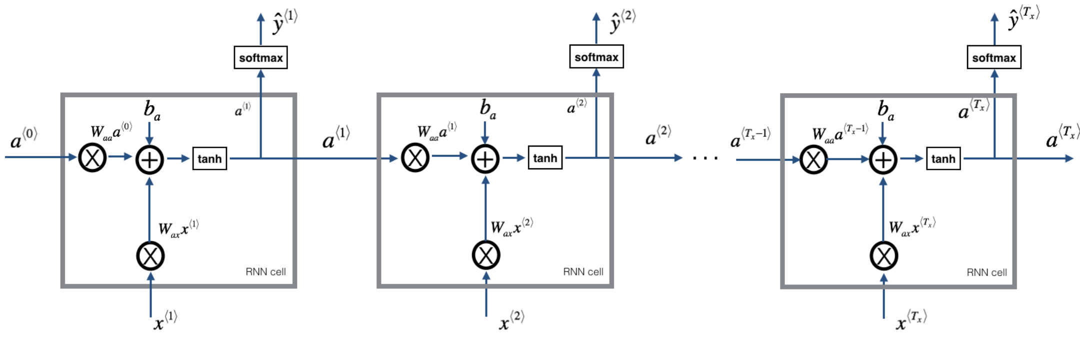

$\hat{y}^{\langle 1 \rangle}$
softmax
$a^{\langle 0 \rangle}$
$W_{aa}a^{\langle 0 \rangle}$
b_a
tanh
$a^{\langle 1 \rangle}$
$W_{ax}x^{\langle 1 \rangle}$
RNN cell
$x^{\langle 1 \rangle}$
$\hat{y}^{\langle 2 \rangle}$
$a^{\langle 2 \rangle}$
$x^{\langle 2 \rangle}$
$a^{\langle T_x - 1 \rangle}$
$\hat{y}^{\langle T_x \rangle}$
$a^{\langle T_x \rangle}$
$x^{\langle T_x \rangle}$

Figure 1.30: Basic RNN model

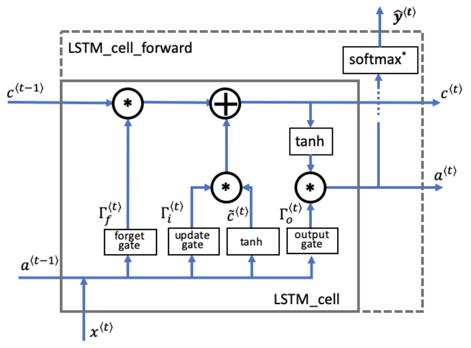
1.8.10 LSTM-based RNN

The standard RNN model presents some drawbacks. Fisrt, when the network is trained for long sequences, it suffers from the vanishing [56] /exploding [57] gradient problems, leading to convergence issues in the training procedure. Moreover, the RNN works best when each output $\hat{y}^{\langle t \rangle}$ can be estimated using *local* context. Local context refers to information that is close to the prediction's time step t. We could consider the RNN to have a short-term memory, which is not optimal when we need to predict events based on long sequence of time steps. To address these problems, we introduce more complex model called Long Short-Term Memory (LSTM) network [58]. The LSTM will be better able to remember a piece of information and keep it saved for many timesteps.

LSTM cell Figure 1.31 shows the operations of an LSTM cell. While the standard RNN cell contains only a single computational unit as shown in fig 1.29, the LSTM cell contains four interacting blocks (or layers), each one have a well-defined purpose. These units are discussed bellow:

- **Forget gate $\boldsymbol{\Gamma}_f$** The forget gate is a tensor containing values that are between 0 and 1. If a unit in the forget gate has a value close to 0, the LSTM will forget the stored state in the corresponding unit of the previous cell state $\mathbf{c}^{\langle t-1 \rangle}$. If a unit in the forget gate has a value close to 1, the LSTM will mostly remember the corresponding value in the stored state $\mathbf{c}^{\langle t-1 \rangle}$. It is computed as follows:

$$\boldsymbol{\Gamma}_f^{\langle t \rangle} = \sigma(\mathbf{W}_f[\mathbf{a}^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_f)$$



$\hat{y}^{\langle t \rangle}$
LSTM_cell_forward
softmax*
$c^{\langle t-1 \rangle}$
$c^{\langle t \rangle}$
tanh
$a^{\langle t \rangle}$
$\Gamma_f^{\langle t \rangle}$
$\Gamma_i^{\langle t \rangle}$
$\tilde{c}^{\langle t \rangle}$
$\Gamma_o^{\langle t \rangle}$
forget gate
update gate
tanh
output gate
$a^{\langle t-1 \rangle}$
LSTM_cell
$x^{\langle t \rangle}$

Figure 1.31: LSTM cell

- **Candidate value $\tilde{\mathbf{c}}^{\langle t \rangle}$** The candidate value is a tensor containing information from the current time step that *may* be stored in the current cell state $\mathbf{c}^{\langle t \rangle}$. Which parts of the candidate value get passed on depends on the update gate. The candidate value is a tensor containing values that range from -1 to 1 , and computed by the following equation:

$$\tilde{\mathbf{c}}^{\langle t \rangle} = \tanh\left(\mathbf{W}_c[\mathbf{a}^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_c\right) \tag{1.28}$$

- **Update gate $\mathbf{\Gamma}_i$** We use the update gate to decide what aspects of the candidate $\tilde{\mathbf{c}}^{\langle t \rangle}$ to add to the cell state $c^{\langle t \rangle}$. The update gate decides what parts of a *candidate* tensor $\tilde{\mathbf{c}}^{\langle t \rangle}$ are passed onto the cell state $\mathbf{c}^{\langle t \rangle}$. It is a tensor containing values between 0 and 1. When a unit in the update gate is close to 1, it allows the value of the candidate $\tilde{\mathbf{c}}^{\langle t \rangle}$ to be passed onto the hidden state $\mathbf{c}^{\langle t \rangle}$. When a unit in the update gate is close to 0, it prevents the corresponding value in the candidate from being passed onto the hidden state. The update gate has the following equation:

$$\mathbf{\Gamma}_i^{\langle t \rangle} = \sigma(\mathbf{W}_i[a^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_i) \tag{1.29}$$

- **Cell state $\mathbf{c}^{\langle t \rangle}$** The cell state is the *memory* that gets passed onto future time steps. The new cell state $\mathbf{c}^{\langle t \rangle}$ is a combination of the previous cell state and the candidate value. The following equation is used to compute the cell state, where we denote by $*$ the element-wise product :

$$\mathbf{c}^{\langle t \rangle} = \mathbf{\Gamma}_f^{\langle t \rangle} * \mathbf{c}^{\langle t-1 \rangle} + \mathbf{\Gamma}_i^{\langle t \rangle} * \tilde{\mathbf{c}}^{\langle t \rangle} \tag{1.30}$$

- **Output gate $\mathbf{\Gamma}_o$** The output gate decides what gets sent as the prediction (output) of the time step. It is like the other gates. It contains values that range from 0 to 1, computed by:

$$\mathbf{\Gamma}_o^{\langle t \rangle} = \sigma(\mathbf{W}_o[\mathbf{a}^{\langle t-1 \rangle}, \mathbf{x}^{\langle t \rangle}] + \mathbf{b}_o) \tag{1.31}$$

- **Hidden state $\mathbf{a}^{\langle t \rangle}$** The hidden state gets passed to the LSTM cell's next time step. It is used to determine the three gates $(\mathbf{\Gamma}_f, \mathbf{\Gamma}_u, \mathbf{\Gamma}_o)$ of the next time step. The hidden state is also used for the prediction $y^{\langle t \rangle}$. The hidden state's equation is:

$$\mathbf{a}^{\langle t \rangle} = \mathbf{\Gamma}_o^{\langle t \rangle} * \tanh(\mathbf{c}^{\langle t \rangle}) \tag{1.32}$$

- **Prediction $\mathbf{y}_{pred}^{\langle t \rangle}$** The prediction in this use case is a classification, so we'll use a softmax. The equation is

$$\mathbf{y}_{pred}^{\langle t \rangle} = \text{softmax}(\mathbf{W}_y \mathbf{a}^{\langle t \rangle} + \mathbf{b}_y) \tag{1.33}$$

LSTM Network As discussed before in the basic RNN model, the LSTM neural network has the form of a chain of repeating modules of LSTM cells. Using the above equations, the LSTM model is able to preserve important information through time and understand which are the essential inputs to keep and what to discard. Figure 1.32 represents a basic architecture of an LSTM network, where always $T_x = T_y$.

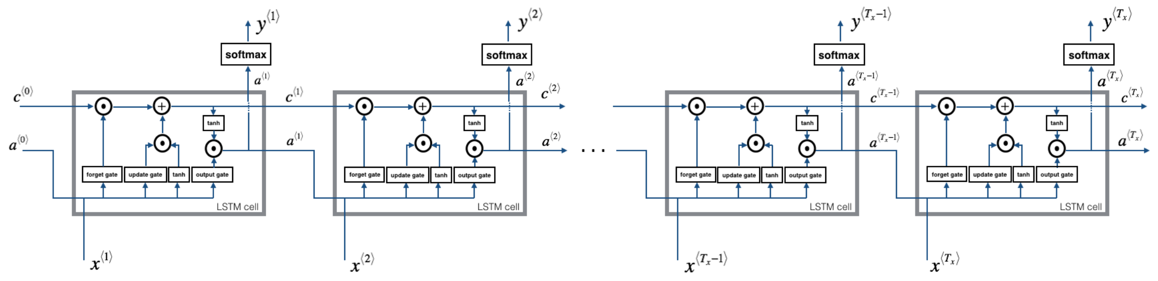


Figure 1.32: LSTM Network

1.9 State-of-the-art DL-based approaches for ES Prediction

ML models for epileptic prediction apply heavily handcraft feature extraction and/or carefully tailored feature engineering to each patient to achieve very high sensitivity and low false prediction rate for a particular dataset. Features are derived from traditional signal processing methods, which are given as input to a classification algorithm such as SVM to make prediction. Hence, it is difficult to produce a generalized automatic system across different patients and different datasets. On the other hand, DL algorithms proposed a generalized framework and patient-specific seizure prediction systems, with more robust features. Table 1.1 represents a summary of DL methods used for epileptic seizure prediction that have been proposed in the literature.

Table 1.1: Summary of DL methods used for ES prediction

Year	Ref	Predictive characteristics	Database	EEG Type	No. of Patients	No. of Seizure per Patient	Prediction Time	Sensitivity
CNN								
2017	Haider et al. [59]	Wavelet Transform	MSSM CHB-MIT	Scalp	47	2.78	8 min 6 min	87.8%
2018	Truong et al. [60]	STFT	Freiburg CHB-MIT American Epilepsy Society	iEEG/ Scalp	28 humans 5 canines	Not mentioned	5 min	81.4% 81.2% 75%
2019	Ramy Hussain et al. [61]	STFT	Melbourne seizure prediction competition dataset	iEEG	3	380	5 min	87.8%
LSTM								
2018	Tsiouris et al. [62]	Various time and frequency features	CHB-MIT	Scalp	24	7.7	15-120 min	99.28%
DCAE + Bi-LSTM								
2019	Hisham et al. [63]	Raw data	CHB-MIT	Scalp	8	5.37	1 hr	99.72%

1.10 Conclusion

This chapter presented general information regarding epileptic seizure prediction based on deep learning using EEG signals. It gives the reader the required basics to understand the seizure prediction systems, starting with epilepsy and EGG, and going through the problem formulation, we gave also the basic foundations of deep learning. In addition, a review of the literature has been presented. To recap, our goal is build a binary classifier based on deep learning models that accurately detect the preictal state in an EEG recording, and satisfy the state-of-the-art performance with minimum preprocessing and feature extraction steps.
The next chapter will present all methods and materials that was used in this project.

Chapter 2

Materials and Methods

2.1 Introduction

Machine learning strategies are ways of analysing a machine learning problems that will point us in the direction to the most promising thing to do next. We need them because they allow us to improve machine learning performance more efficiently by making the correct decision at the right time. For example, should we train a bigger model or use regularization techniques if we had a variance problem ?
In this chapter, we describe the environment in which the models were trained and justify the strategies that were used. The chapter begins by a presentation of hardware tools available in this project and needed software tools. Then, we presents the iEEE dataset from the American Epilepsy Society Seizure Prediction Challenge. It is followed by a brief analyzing of the data with some visual methods. Next, data reduction is proposed using resampling technique in order to decrease the memory and computational resources. Furthermore, EEG clips are split into small segment, with overlapping for the preictal state in order to overcome the imbalanced classification problem. Finally, we will discuss the cross-validation technique used to evaluate the models and the learning rate tuning using the TensorBoard toolkit.

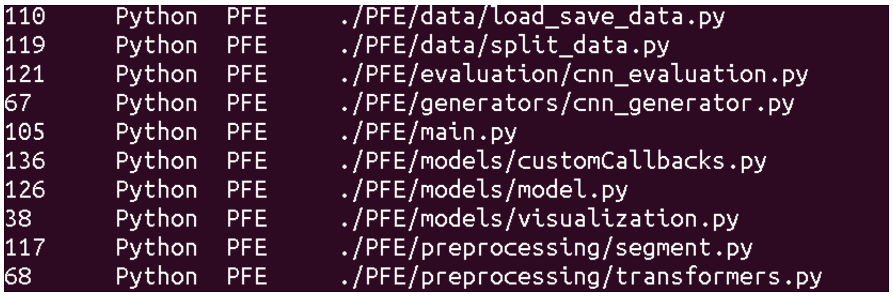
2.2 Tools

2.2.1 Hardware Tools

The only available hardware to use for training models in this project was my personal computer *ThinkPad T460s*. It uses an Intel processor "Intel® Core$^{\text{TM}}$ i5-6200U". This processor contains two cores (physical CPU) and four threads (virtual CPU). It operates at the base frequency of 2.30 GHz and maximum turbo frequency of 2.80 GHz. It also features 3 MB cache memory. *Ubuntu 18.04* has been used as the operating system for its flexibility in installing others packages. Another important specification that need to be mentioned is the RAM memory, we only have 8 GB.

2.2.2 Software Tools

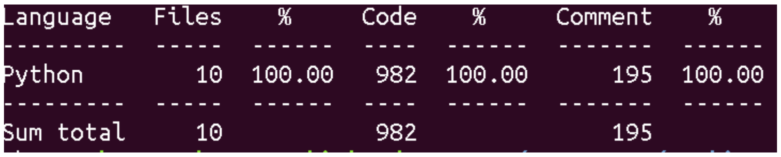
For software implementation, this subsection will present the most used tools. Essentially, we used Python as our programming language combined with its data science and machine learning libraries like NumPy, SciPy, Matplotlib, sklearn and TensorFlow. For readability and scalability, we used modular programming paradigm. Each module contains related tasks in order to accomplish the full pipeline of seizure prediction. These modules are then called in the main function. Figure 2.1 presents a list of python files and the number of code lines in each.



```
110     Python  PFE     ./PFE/data/load_save_data.py
119     Python  PFE     ./PFE/data/split_data.py
121     Python  PFE     ./PFE/evaluation/cnn_evaluation.py
67      Python  PFE     ./PFE/generators/cnn_generator.py
105     Python  PFE     ./PFE/main.py
136     Python  PFE     ./PFE/models/customCallbacks.py
126     Python  PFE     ./PFE/models/model.py
38      Python  PFE     ./PFE/models/visualization.py
117     Python  PFE     ./PFE/preprocessing/segment.py
68      Python  PFE     ./PFE/preprocessing/transformers.py
```

Figure 2.1: List of line counts in each python file

For the whole project, Figure 2.2 shows summary of code lines and comment lines.



```
Language   Files    %      Code    %      Comment    %
---------  -----  ------   ----  ------   -------  ------
Python        10  100.00    982  100.00       195  100.00
---------  -----  ------   ----  ------   -------  ------
Sum total     10            982               195
```

Figure 2.2: Summary of source code

Python [64] The programming language that has been used along this project is Python v3.7. Python is an interpreted, object-oriented, high-level programming language with dynamic typing. The most advantage of python is that it is easy to learn, clear to read, and simple to write in. Moreover, python is a general purpose programming language that is used in web development (server side), mathematical computing and even embedded systems. In the last few years, python is becoming the default choice and the most used in the field of data science.

NumPy [65] NumPy is an open source python library and a very powerful tool in scientific computing. It is based on multidimensional arrays through matrix representation and functions that operates on arrays. The array object called *ndarray* is up to $50x$ faster that traditional Python list, where most low level operations are written in C. Numpy was used in all the project to handle and manipulate EEG data;

SciPy [66] SciPy is an open source Python-based library, dedicated for scientific computing, mathematics, engineering and science. It is a library that uses NumPy arrays as the basic data structure and proposes modules for different tasks including signal processing, image processing and calculus. It was used in the preprocessing step such as resampling.

Matplotlib [67] Matplotlib is a plotting library for creating 2-D plots that can be saved as figures. Its core module is called *Pyplot* which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB with the advantage of being free and open-source. All the plots in this project have been created using this library.

Scikit-learn [68] Sklearn is the most common general machine learning library. It is built on the library NumPy for high-performance linear algebra and array operations, and also uses SciPy and Matplotlib libraries. It features various tools for machine learning related methods including data preprocessing, feature extraction, feature selection, regression, classification and clustering.

TensorFlow [69] The deep learning framework that was used to build models is TensorFlow V2.0.1. TensorFlow is an end-to-end open source platform and one of the most popular frameworks. It lays out a full ecosystem of tools, libraries and community resources that ease building machine learning models. TensorFlow has APIs available in many languages, but Python API is the most complete and approved. It was first developed in 2015 by the deep learning artificial intelligence research team at Google, Google Brain, and then released under the Apache License 2.0 on November 9, 2015. With the new release version 2.0 of TensorFlow, the high level neural networks API ”Keras” has became the official API for building deep learning models. Keras was originally a separate library from TensorFlow and designed for fast experimentation with deep neural nets and is thus simpler [70]. This choice can be convinced by the fact that I master this library.

2.3 Dataset

In this project we consider the dataset from the "American Epilepsy Society Seizure Prediction Challenge", an international competition organized by National Institutes of Health, the Epilepsy Foundation, and the American Epilepsy Society [71]. Its goal was to identify the best model for discriminating preictal and interictal iEEG clips. The iEEG data for the competition was recorded from 5 dogs with the naturally occurring epilepsy. EEG was sampled from 16 electrodes at 400 Hz, and recorded voltages were referenced to the group average. These recordings span multiple months up to a year and contain up to a hundred seizures in some dogs. A seizure advisory system, shown in Fig 2.3, is based on an implantable device (labeled as Figure 1A). It acquires 16 channels of iEEG (Figure 1B) and wirelessly transmits the data to personal advisory device, which stores it on a flash drive and uploads weekly via the internet to a central data storage site [72].

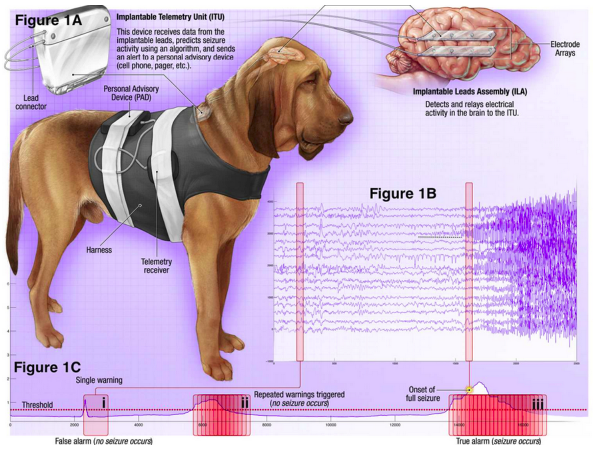


Figure 1A
Implantable Telemetry Unit (ITU)
This device receives data from the implantable leads, predicts seizure activity using an algorithm, and sends an alert to a personal advisory device (cell phone, pager, etc.).
Lead connector
Personal Advisory Device (PAD)
Electrode Arrays
Implantable Leads Assembly (ILA)
Detects and relays electrical activity in the brain to the ITU.
Harness
Telemetry receiver
Figure 1B
Figure 1C
Single warning
Threshold
i
ii
iii
Repeated warnings triggered (no seizure occurs)
Onset of full seizure
False alarm (no seizure occurs)
True alarm (seizure occurs)

Figure 2.3: Seizure advisory system in canines with epilepsy [72]

Seizures are known to occur in clusters, which implies little benefit from forecasting follow-on seizures. Thus, only leading seizures are included, defined as seizures occurring four hours or more after another seizure. Interictal data segments were chosen randomly from the whole iEEG recording so that they are at least at one week before and after any seizure, to avoid contamination with preictal or postictal signals.
The dataset is organized into ten minute long clips of preictal and interictal activity. These clips are grouped into one hour sequences (each six 10-min clips form a sequence), and numbered sequentially. Preictal data segments are provided covering one hour prior to seizure with a five minute seizure horizon. (i.e. from 1:05 to 0:05 before seizure onset) as shown in Figure 2.4. This pre-seizure horizon ensures that seizures could be predicted with enough warning to allow administration of fast-acting medications.

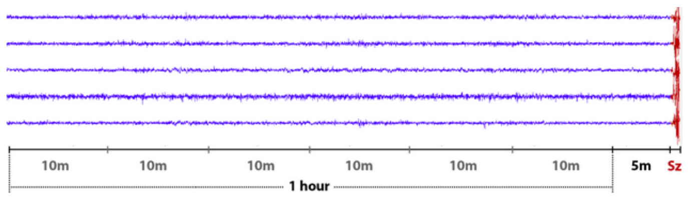


Figure 2.4: Example of iEEG signal (only 5 channels) showing 1 hour and 5-min before the seizure

2.4 Performance Evaluation

The required output was a probability of the preictal state for a given EEG segment. To evaluate performance, organizers used a single evaluation metric called Area Under Curve (AUC).
The Receiver Operating Characteristics (ROC) graph is a technique for visualizing, organizing and selecting classifiers based on their performance [73]. It is a two-dimensional graph in which True Positive Rate (TPR) is plotted on the Y axis and False Positive Rate (FPR) is plotted on the X axis for different thresholds of classification. A ROC graph depicts relative trade-offs between benefits (true positives) and costs (false positives). The AUC is the area under the (ROC) curve. It is a metric that ranges from 0 to 1. It can interpreted as the probability that a classifier ranks a randomly chosen positive instance higher than a randomly chosen negative one. A random classifier has an AUC of 0.5, whereas the maximum is 1 (see Fig. 2.5).

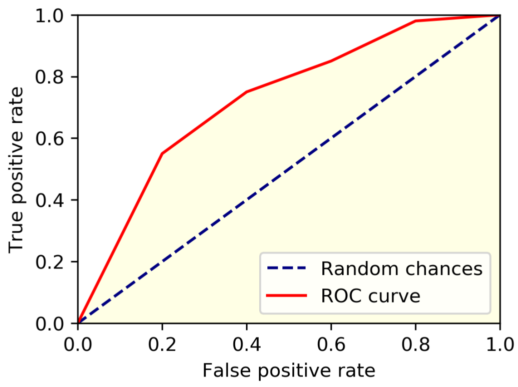


Figure 2.5: An example of a ROC curve

FPR and TPR are defined by the following equations where FP, FN, TP and TN were defined in Chapter 1.

$$FPR = \frac{FP}{TN + FP} \tag{2.1}$$

$$TPR = \frac{TP}{TP + FN} \tag{2.2}$$

2.5 Exploratory Data Analysis

In this section, we will start investigating the EEG data used in our project in order to get some insights or discover hidden patterns or anomalies. This is not an extensive quantitative analysis of the data, but it is recommended to have at least a look to the data before working on models. Here's an advice from Andrej Karpathy, the director of artificial intelligence and Autopilot Vision at Tesla.

> "The first step to training a neural net is to not touch any neural net code at all and instead begin by thoroughly inspecting your data. This step is critical. I like to spend copious amount of time (measured in units of hours) scanning through thousands of examples, understanding their distribution and looking for patterns. [74]"

We start by a global view of the data, let us see the number of 10-min clips for each subject. This allows us to see the data distribution over the two classes. Figure 2.6 show a bar plot that presents the number of data clips in each state. It indicated clearly that the dataset is highly imbalanced, where there is much more interictal clips than preictal ones. This may be justified by the fact that a seizure is rare event compared to the baseline state. Thus, we have less access to the preictal state.

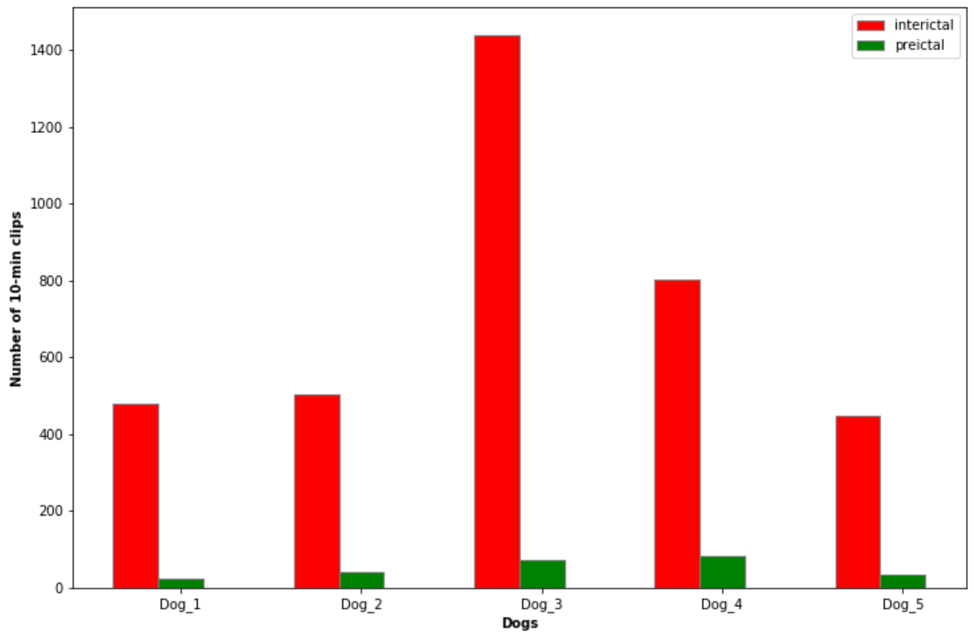


Figure 2.6: A bar plot: the number of preictal/interictal data clips

Table 2.1 presents a summary of the used dataset, where for each subject we give the number of seizures, the number of preictal clips, the number of interictal clips and the proportion of interictal clips compared to the preictal clips to demonstrate that the imbalanced aspect of the data.

Table 2.1: Summary of EEG dataset

Dog	**No. of seizures**	**Preictal clips**	**Interictal Clips**	**Imbalance Ratio**
Dog 1	4	24	480	1:20
Dog 2	7	42	504	1:12
Dog 3	12	72	1440	1:20
Dog 4	14	84	804	1:9.57
Dog 5	6	36	450	1:12.5

Next, as mentioned above, it is useful to visualize some examples to see visual differences if there are. Figure 2.7 presents a visualisation of 10-min preictal clip, whereas Fig 2.8 shows a 10-min interictal clip. The two clips belong to subject one, and clearly can be distinguished visually.

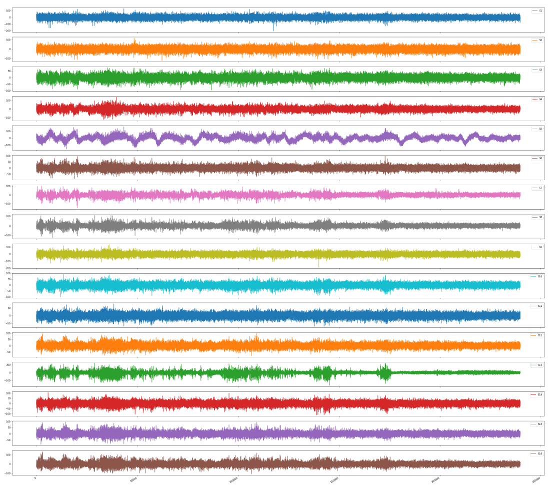


Figure 2.7: An illustration of dog 1's preictal clip

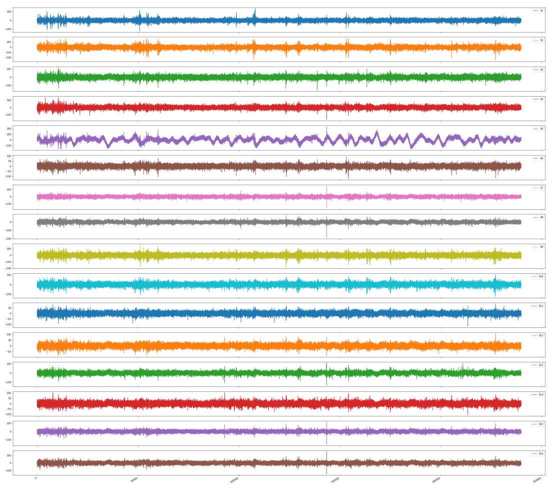


Figure 2.8: An illustration of dog 1's interictal clip

As we discussed earlier, machine learning based algorithms focus on extracting features from EEG signals. Some of these features are statistical such as mean and variance. Therefore, let's see if preictal and interictal clips are statistically different, and if the answer is yes, then we can use their statistical features to differentiate between them. The best tool to use in answering this question is descriptive statistics, where we use boxplots to visualize these results. A boxplot is a standardized way of displaying the distribution of data based on a five number summary (minimum, first quartile (Q1), median, third quartile (Q3), and maximum). It can be interpreted using the following quantities:

- **Median** the middle value of the dataset, represented by the horizontal line in the center of the box
- **Range** the range describes the spread of the data, represented by the vertical distance between the minimum and the maximum values
- **Inetrquartile Range (IRQ)** this is the box plot showing the middle 50% of scores (i.e., the range between the first quartile and the third quartile).

Figure 2.9 shows a comparative boxplot for preictal and interictal iEEG data for subject 02. First, we observe that the median values for all 16 channels for both preictal and interictal data are around zero. Moreover, for all sensors, the range and interquartile range of the interictal clips are greater than those of the preictal ones. In addition, we notice that both types display potential *outliers* at both sides.

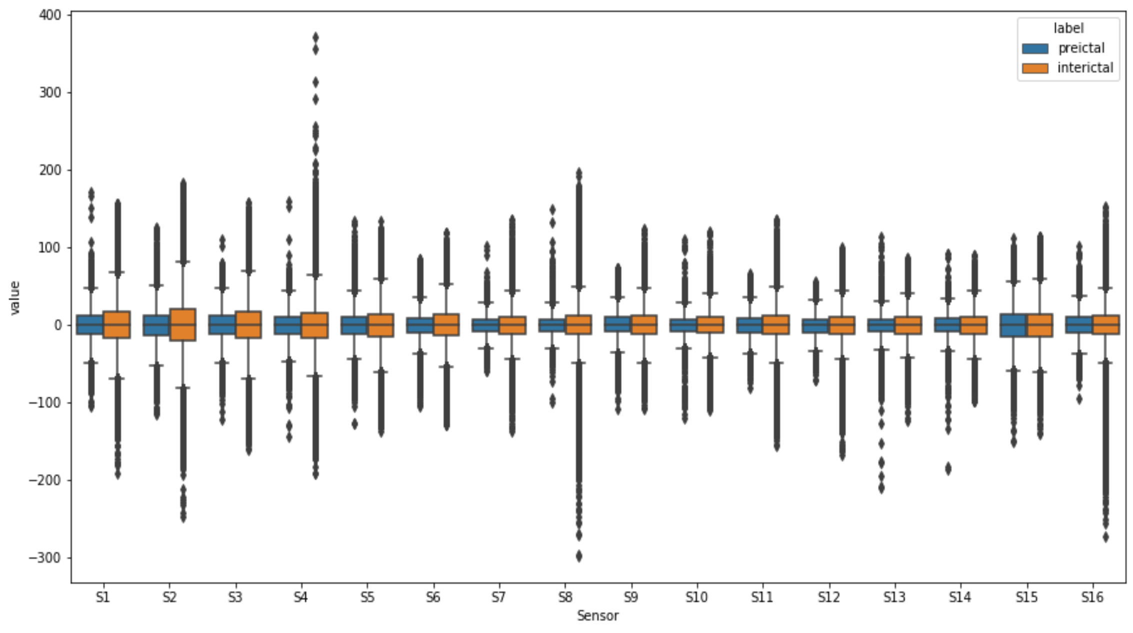


Figure 2.9: Boxplots of dog 2's data

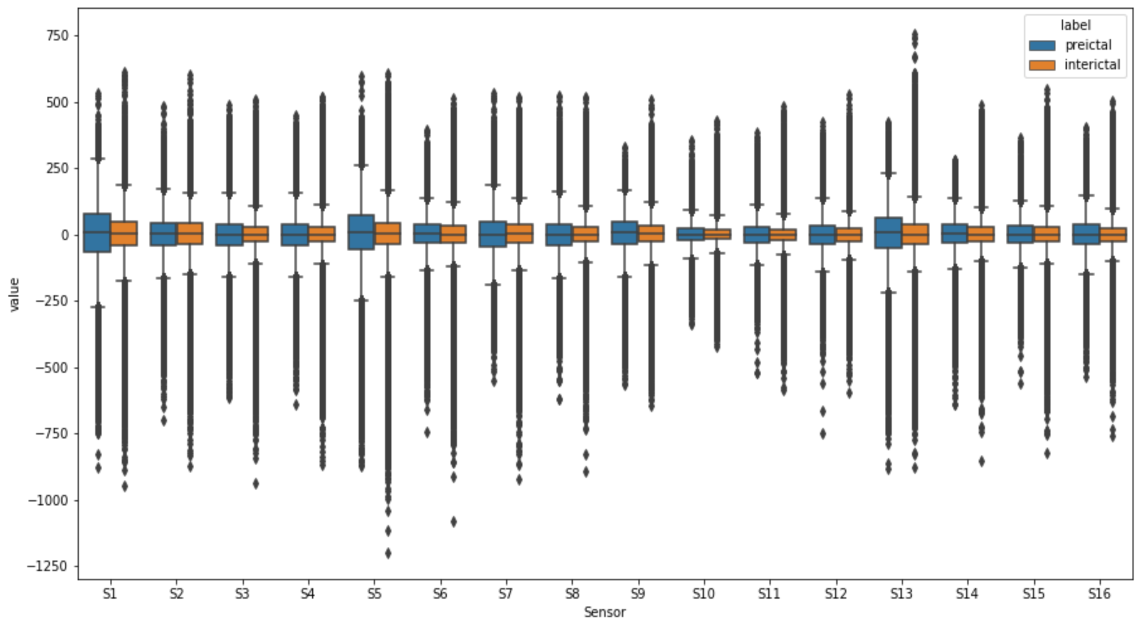


Figure 2.10: Boxplots of dog 4's data

Even with the fact that the preictal and interictal clips seem to have different variability for this subject, which implicate that they are statistically different, we can not rely on these features as they are subject-specific. Figure 2.10 depicts that the distribution of interictal and preictal data for subject 4 are different from those found in subject 2, where the interquartile of preictal clips is much greater than those of the interictal ones in all channels. Based on these observations for different subjects, we suggest that there is no typical trend in our data and we conclude that these features can be only significant for building patient-specific feature extraction algorithms, which is far from our goal.

2.6 Data Reduction

As described in the previous section, our dataset had a total of 3936 EEG clips (3678 interictal and 258 preictal). Each clip has 10-min recording, sampled at the exact frequency of 399.61, which gives 239766 timesteps $(10min \times 60sec \times 399.609756097561Hz)$ for each clip. Reducing data size will be beneficial for both memory and computational resources.
Figure 2.11 presents an example of the frequency spectra of interictal and preictal EEG clips, where for each sub-figure, we draw the spectrum of one channel (blue for preictal and orange for interictal) and a vertical line at $f = 128$ Hz. We can notice that, for both interictal and preictal

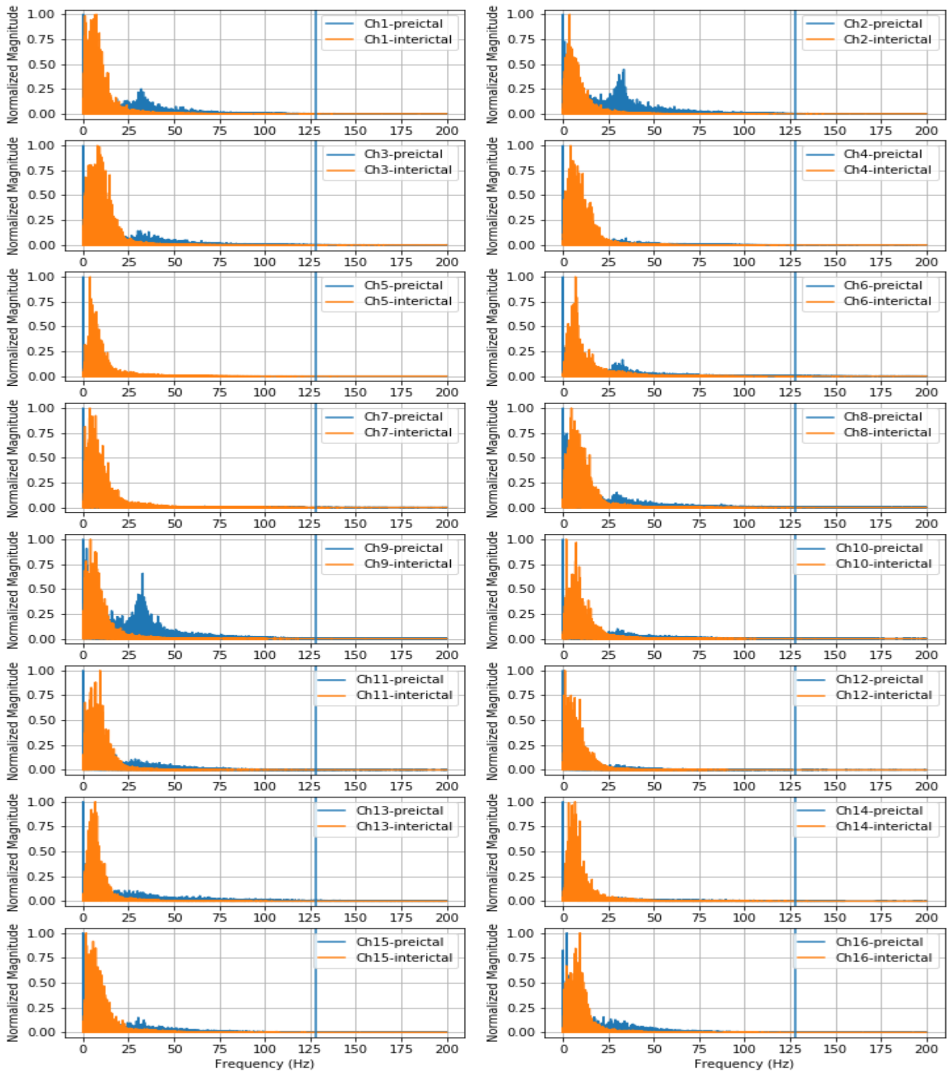


Figure 2.11: Frequency spectra of interictal and preictal dog 1's EEG signals collected by the 16 implanted electrodes

EEG, an important quantity of information is concentrated at low-frequency band $(< 128Hz)$, and almost there is no information at high-frequency band $(> 128Hz)$. Note that we can choose much less limit for low-frequency band but it is useful to add a secure region and insure that this operation does not distort data. Also, for many acquisition systems in embeded systems used the frequency of 256Hz. According to the Nyquist rule, we resample EEG data at $256Hz$ $(2 \times 128Hz)$

instead of $400Hz$.This operation is called *down-sampling* in the field of signal processing. Thus, the data dimension is reduced by a factor of 1.5625 and that implies that the data storage is also decreased by the same factor. Now each clip has 153600 timesteps $(10min \times 60sec \times 256Hz)$.
To demonstrate the effect of this operation on EEG signal, we choose one channel for illustration as shown in Fig. 2.12. Figure 2.12(a) shows the original univariate time series, Fig. 2.12(b) shows its spectrum along the frequency range $0 - 128Hz$, Fig.2.12(c) shows the down-sampled version in time domain and Fig.2.12(d) shows its frequency spectrum.

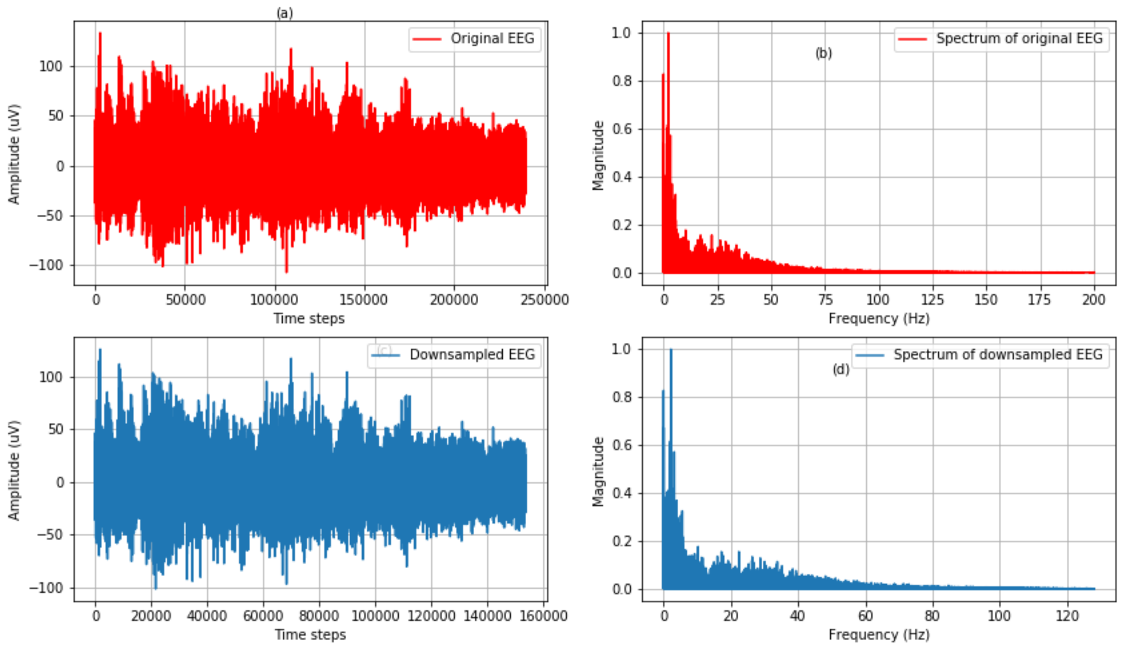


Figure 2.12: Time-series EEG signals and their corresponding spectra

2.7 Data Segmentation and Imbalanced Dataset

As seen previously , our dataset is heavily imbalanced. Consequently a classification algorithm will perform well by just classifying all examples as interictal. Many strategies have been proposed to attack this problem [75]. Imbalanced dataset classification techniques can be categorized in four types:

- **Data Sampling Algorithms** They change the composition of the training dataset including data oversampling, data undersampling and combined oversampling and undersampling.

- **Cost-Sensitive Algorithms** They are modified versions of machine learning algorithms designed to take the differing costs of misclassification into account when fitting the model on the training dataset.
- **One-Class Algorithms** They used for outlier detection and anomaly detection can be used for classification tasks.
- **Probability Tuning Algorithms** Predicted probabilities can be improved in two ways: calibrating probabilities or tuning the classification threshold.

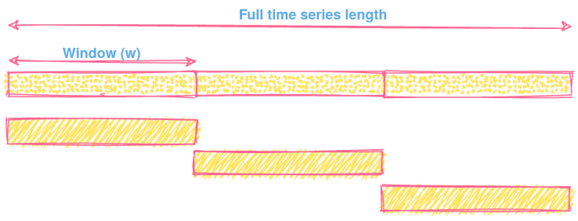


Figure 2.13: Interictal EEG segmentation

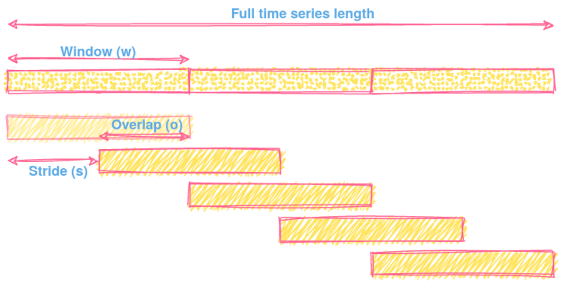


Figure 2.14: Preictal EEG segmentation

Time-series segmentation is a method of time-series analysis, used mainly to partition the full time series data into smaller segments. For interictal data, 10-min clips are divided into $W - sec$ length segments with no overlapping between adjacent segments, where W is the length of each window in seconds as shown in Fig. 2.13. Whereas for preictal data, in order to overcome the imbalanced classification problem, we generate more preictal segments by using overlapping technique during training phase. In particular, we create extra preictal samples for training by sliding a W-sec window along time axis at every step S-sec over preictal time-series EEG signals, where W is the length of the window in seconds, and S is the stride length in seconds that can be replaced by the overlapping window O (see Fig. 2.14). Those parameters (W,S) and can be learned in the training process through grid search optimization, but due to the limitation in computational resources they are set according to the most commonly used values.
Each segment is then considered as a training sample and takes the same label as the full clip for both preictal and interictal EEG time series. A main advantage of this operation is that it outputs a large number of labeled samples needed for training deep learning models. For example, let us assume that ($W = 1min$,$S = 30sec$), this results into 10 non-overlapping 1-minute interictal EEG samples, and 19 overlapping 1-minute preictal EEG samples; each sample has 15360 timesteps. This results in an increase in the total number of training (interictal and preictal) EEG samples. Thus this method belongs to minority-class data oversampling techniques.

2.8 Cross Validation Technique

In Chapter 1, we made the distinction between the *test error* and the *training error*. The training error is easily calculated by applying the algorithm to the observations used in the training set. In contrast, we need a technique to get an unbiased estimate of the test error. Cross-validation (CV) is one of the most widely used methods to produce robust measurements of model performance known as *model assessment*. Many cross-validation techniques have been used, two of the most common types are hold-out cross-validation and k-fold cross-validation [76], [77].
It is common in image classification tasks to use the hold-out method as a cross-validation technique. First, samples in the dataset are shuffled. Second, we split the dataset into a subset called the training set, and another subset called the test set. Then, it is usual to have some hyper-parameters to optimize such as the size of a filter or the number of units in a layer. Therefore, we split the training set into a training subset and a validation set. The model is trained on the training subset and the parameters that minimize error on the validation set are chosen. Finally, the test dataset provides the benchmark standard for model evaluation.

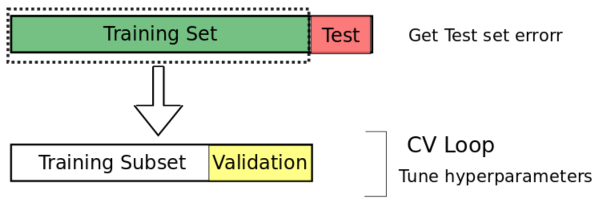


Figure 2.15: Visualization of the data splits .

Two problems arise when we try to apply this method to our problem. First, the time series data (EEG signals) is often strongly correlated along the time axis (EEG data segments within the same 10-min clip). The randomization will make it likely that for each sample in the test set, many correlated samples exist in the training set. Second, the dataset provides a small number of seizures and to ensure robustness and generalization of the proposed models we evaluate the models on all seizures not only some of them as even seizures of the same subject are different due to the non-stationarity nature of EEG signals. So, we used the Leave-one-out cross validation (LOOCV) technique as the evaluation method for all of our proposed models. If a subject had N seizures, $(N-1)$ seizures are involved in the training process, and the remaining one is used for testing. The process is then repeated N times by changing the seizure under test, so that all seizures were used for testing once and the samples related to the tested seizures are unseen during the training. Interictal segments are also split into N parts while respecting the temporal order, where $(N-1)$ parts were used for training and the remaining part were used for testing. The performance for one subject is the average across N trials and the overall performance is the average across all subjects. In addition, 80% of the training data is assigned to the training set while 20% is assigned to the validation set (as a rule of thumb). This last split is done by carefully considering the temporal order. We use the validation set to monitor the loss function on the validation set as the criteria for early stopping, where after each training epoch, the loss is calculated with respects to the validation data to check if the network starts to over-fit the training set. This process of early-stopping is used to avoid overfitting and was discussed earlier.

2.9 Hyper-parameters Optimization

Every deep learning model has a certain number of parameters that we need to define before the training process, often called *hyper-parameters*. Hyper-parameters refers to parameters that cannot be updated during the training of machine learning models. They can be categorized into two groups: those used for training like the learning rate and the optimizer, and those used for model design such as the number of neurons , filter size and the dropout rate. The process of designing the ideal model architecture with an optimal hyper-parameter configuration is named *hyper-parameter tuning*. Manual testing is a traditional way to tune hyper-parameters, the main disadvantage of which is time-consuming, complex models and a large number of hyperparameters. Consequently, this prompts an increased search for automatic optimization techniques for hyper-parameters, called hyper-parameter optimization (HPO). Many hyper-parameter optimization techniques have been developed, the well-known of which are Grid Search, Random Search and Bayesian search [78],[79].
The more hyperparameters that we need to tune, the slower the tuning process. Therefore, we will only tune the learning rate as it is the most important parameter as mentioned bellow:

> ”The learning rate is perhaps the most important hyper-parameter. If you have time to tune only one hyper-parameter, tune the learning rate” [31].

To do so, we used *TensorBoard*, a TensorFlow’s visualization toolkit for providing the measurements and visualizations needed during the machine learning workflow such as tracking experiment metrics like loss and accuracy, visualizing the model graph. Hyper-parameter Optimization is insured by *HParams*, a dashboard in TensorBoard that provides several tools to help with the process of identifying the best sets of hyper-parameters.
Figure 2.16, shows an example of using this tool. For simplicity, we use a grid search over the search space $[10^{-5} - 10^{-1}]$ to tune the learning rate. This will run experiments for each value and presents the results in three different ways: the *table view* lists the runs, their hyperparameters, and their metrics, the *the parallel coordinates view* shows each run as a line going through an axis for each hyper-parameter and metric, the *scatter plot view* shows plots comparing each hyper-parameter/metric with each metric.

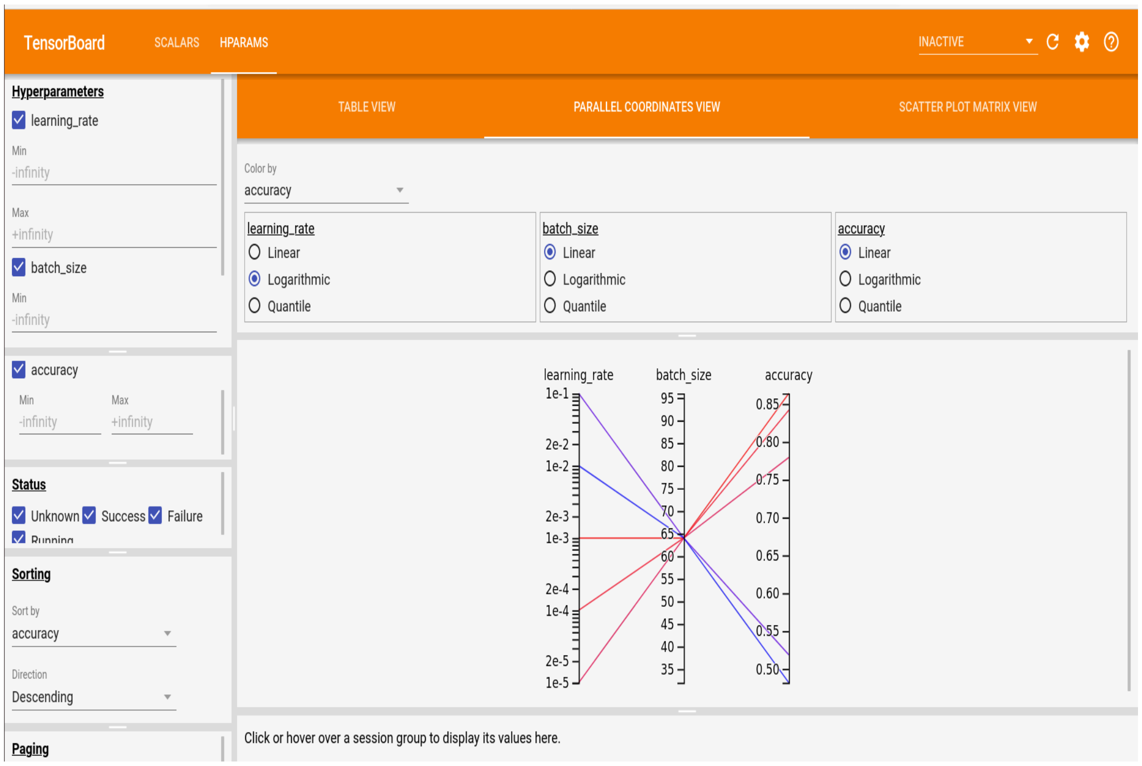


Figure 2.16: An illustration of HPO using TensorBoard

2.10 Conclusion

This chapter gave the reader the main strategies that were used to build and train the proposed models. We have discovered the dataset, the evaluation metric as well as the evaluation process by the LOOCV method.
We have also seen two of the building blocks of our algorithms; data reduction and data segmentation. In the next chapter, we will move to the data preparation step needed to every model.

Chapter 3

Implementation

3.1 Introduction

Recently, Deep Learning (DL) models have shown great capability in many practical applications. State-of-the-art results have been achieved in a variety of problems such as image recognition, object detection and text processing, by automatically learning features from the raw data. In the era of big data, we tend to use an end-to-end deep learning approach, in which a lot of intermediate steps are bypassed and less hand-designing of components is needed. Moreover, DL models have even surpassed human-level performance in several problems like online advertising, product recommendation and loan approval. These examples are learning from structured data. In contrast, it is a bit harder for a machine to surpass human-level performance on natural perception tasks like computer vision, voice and natural language processing [80].
Most common deep learning models are convolutional neural networks and recurrent neural networks. A CNN network is generally composed of chain of convolution layers and pooling layers interchangeably for extracting low-level spatial features in first layers to high-level features in deep ones, followed by dense layers for prediction. Whereas, LSTM networks are more suitable for sequence modeling in which the data is time ordering.
In this chapter, we present the two proposed models for classifying EEG segments into preictal and interictal states as well as the data preparation steps for each model. The first model is 1-D Convolutional Neural Network (CNN), whereas the second model takes the advantages of both CNN and LSTM. We will present our findings and compare them to other works found in the literature.

3.2 Convolutional Neural Network Model

Convolutional Neural Networks have demonstrated high performance in visual related tasks and are bridging the gap between the capabilities of humans and machines despite the fact that the human visual system is fast and accurate. This is due to the ability of CNN to automatically extract relevant spatial features that best represents the data from its raw form without any pre-processing or human decision in selecting these features. Those models accept a two-dimensional input representing an image's pixels and color channels in a process called feature learning. This motivates researchers to use these models for other applications than computer vision. In this section, we propose a 1-D convolutional neural network to differentiate between preictal and interictal segments. Note that CNNs work the same way whether they are 1-D, 2-D or 3-D. The only difference is the structure of the input data and how the filter scan the data to produce the feature maps.

3.2.1 Data Preparation

In the previous chapter, we have seen that EEG signals are split into small segments of length W, and with overlapping O for preictal state. For this model, we choose $W = 5 - sec$ and $O = 0.5$, which results in segments of 1280 time steps $(5 - sec \times 256Hz)$. Since the acquisition system has 16 electrodes, thus there are a total of 16 variables for each time step. Each sample will have then the shape $[1280, 16]$. The choice of W and O is done based on what we have seen in many other works [63], [62]. Note that a large window leads to high-dimensional input data. Those numbers can be tuned through the hyper-parameter optimization process as done in [81], but this need more resources.
The $5 - sec$ segments form the input, as raw data with no feature engineering, of our model and are used in the training process, in which the batch shape is $[samples, timesteps, features] = [\#samples, 1280, 16]$. The model outputs a probability distribution over the two classes for each segment. Please note that the training is subject-specific but the architecture is the same across all subjects.

3.2.2 Architecture

Figure 3.1 presents the proposed 1-D CNN architecture. It consists of three convolution blocks. Each basic block contains three operations: a convolution layer followed by a batch normalization layer whose results is passed to a ReLU activation function. These three blocks acts as a feature extractor. The output of the third convolution block is fed to the Global Average Pooling (GAP) layer which averaged it over the whole time dimension. Finally, a sigmoid function is fully connected to the GAP layer's output.

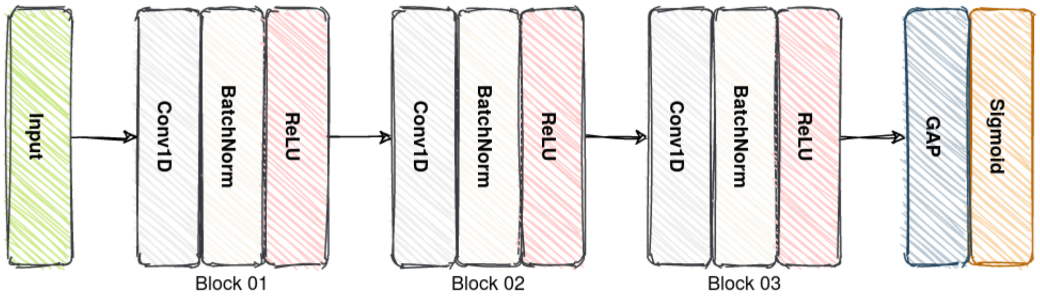


Figure 3.1: The proposed CNN architecture

The convolution layers are realized using 1-D kernels with a stride equal to 1 and a zero padding to preserve the exact length of the time series after the three convolution blocks. The first convolution contains 128 filters with a kernel size equal to 8, the second convolution comprises 256 filters with a kernel size equal to 5, and the final convolutional layer composed of 128 filters, each one with a length equal to 3.
This CNN do not include any local pooling operation, which means that the number of time steps does not change through the three convolution blocks. In addition, we used a global average pooling layer instead of a fully connected layer, which reduces the number of weights [82]. This strategy is also used in the very common ResNet architecture [83] to prevent overfitting. Therefore, this type of architectures are called *fully convolutional neural networks*.

3.2.3 Results

This model was trained and tested following the cross-validation method described earlier. For each subject, the AUC is averaged across the N trials, where N is the number of seizures. Table 3.1 presents the results for each subject as well as the average. This model achieves testing area under the operation characteristic curve (AUC) of 0.813 on average.

Table 3.1: Summary of CNN results

Subject	**No. of seizures**	**Interictal Duration (h)**	**Preictal Duration (h)**	**AUC**
Dog 1	4	80	4	0.651
Dog 2	7	84	7	0.933
Dog 3	12	240	12	0.930
Dog 4	14	134	14	0.772
Dog 5	6	75	6	0.958
Average	8.6	-	-	0.848

3.3 Long-term Recurrent Convolutional Network Model

Since we are manipulating time series data, it is reasonable to use recurrent neural networks either with basic RNN cells or LSTM cells. However, our multivariate data have also a spatial structure and cannot be easily treated with standard LSTM. The major drawback of the Vanilla LSTM in handling spatio-temporal data is that it has to flatten the inputs to 1-D vectors before processing and, as a result, all the spatial information will be lost during the process. For more details, return to section 1.8.9 where $x^{\langle t \rangle}$ the t time-step's input data is 1-dimensional vector. Therefore, we prefer to use a CNN-LSTM recurrent neural network. This type of architectures was originally referred to as Long-term Recurrent Convolutional Network (LRCN) [84] . They involve using CNN layers for feature extraction on spatial input data and then fed to an LSTM network to support sequence modeling. They were developed for visual recognition and description tasks such as activity recognition, image captioning and video description. They are appropriate for problems that have spatial structure in their input such as an image, as well as a temporal structure such as the order of images in a video.

> LRCNs are a class of models that is both spatially and temporally deep, and has the flexibility to be applied to a variety of vision tasks involving sequential inputs and outputs [84].

3.3.1 Data Preparation

The data preparation step for this model involves also EEG signals segmentation into small sequences. The LRCN model will read sub-sequences of the main sequence in as blocks, extract features from each block in a parallel way using the same CNN, then feed the LSTM network with the features extracted from each block with time ordering. One approach to implementing this model is to choose $W = 1min$ and $O = 0.75$, which results in sequences of 15360 time steps $(60 - sec \times 256Hz)$. Each sequence of 1-min is split into 12 sub-sequences of $5 - sec$, and thus each sub-sequence contains 1280 time steps $(5 - sec \times 256Hz)$.
The spatial feature extraction for this model is done by a 2-D CNN. Therefore, we need to transform each multivariate time series sub-sequence into image-like format. This was done by a simple algorithm inspired from [85]. Raw signals are stacked row-by-row into a signal matrix based on Algorithm 1. In the signal matrix, every channel signal has the chance to be adjacent to every other channel, which enables CNN to extract hidden correlations between neighboring channels. For the sake of simplicity, Figure 3.2 illustrates this process for the case where the number of channels is $N_s = 5$. Figure 3.2(a) presents the raw signals that are fed to Algorithm 1 to produce the signal matrix that has 11 rows. Finally, we transform the signal matrix to a gray-scale image by just considering the voltage values as the gray level (we only change the software format). Thus, the signal image is interpreted as a gray-scale image that is passed to the

CNN network.

Algorithm 1: Raw Signals $\longrightarrow$ Signal Image

```
Notations:
  • Signal Matrix (SM): a 2D array to store permutated raw signals.
  • Signal Index String (SIS): a string to store signal indices, whose length is $N_{SIS}$
input : $N_s$ signal sequences stacked horizontally in a matrix $RS$, each signal has a
        sequence label
output: SM
$i \leftarrow 1$ ;
$j \leftarrow i + 1$ ;
$SM \leftarrow RS(i)$ ;
$SIS \leftarrow$ 'i' ;
$N_{SIS} \leftarrow 1$;
while $i \neq j$ do
    if $i \neq j$ then
        $j \leftarrow 1$ ;
    else if $'ij' \not\subset SIS$ && $'ji' \not\subset SIS$ then
        Append $RS(j)$ to the bottom of SM ;
        Append 'j' to SIS ;
        $N_{SIS} \leftarrow N_{SIS} + 1$ ;
        $i \leftarrow j$ ;
        $j \leftarrow i + 1$ ;
    else
        $j \leftarrow j + 1$ ;
// When $N_s$=5 as in Fig. 3.2, the final SIS is '12345135241' and $N_{SIS} = 11$
```

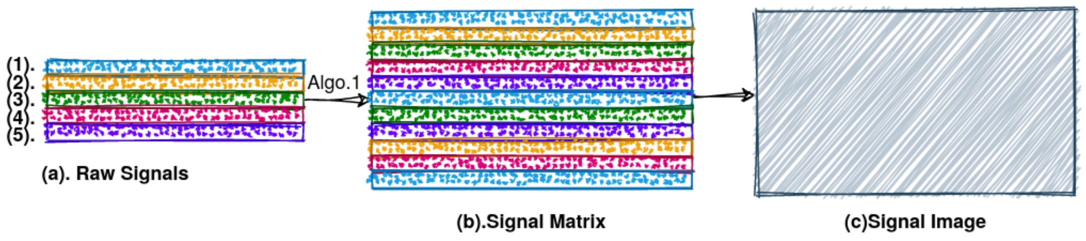


Figure 3.2: Flowchart to generate an image-like format from time series data

For our case, we have 16 channels for every sub-sequence and thus Algorithm 1 will produce an image of size 114×1280 pixels. We have split each sequence into 12 sub-sequences and thus the number of steps in the LSTM network is 12 (number of LSTM cells is 12). The batch shape is then $[samples, n\_steps, rows, columns, depth] = [\#samples, 12, 114, 1280, 1]$, where depth is the number of channels in the image, equals to 1 as it is gray-scale.

3.3.2 Architecture

The proposed architecture is presented in Fig. 3.3. It has a sequential input $[x^{\langle 1 \rangle}, x^{\langle 2 \rangle}, ..., x^{\langle Tx \rangle}]$, each $x^{\langle t \rangle}$ is an image produced as described above, and a static output y that returns a probability distribution over the two classes for each sequence. The number of sub-sequences Tx was chosen to be 12 in the training phase as mentioned above, but it can be any other chosen number. It can be also tuned through the hyper-parameter optimization process. This model works by passing each image $x^{\langle t \rangle}$ through feature transformation using the 2D-CNN to produce a fixed-length vector representation. These vectors are then passed into a recurrent sequence learning module using the LSTM network. Each LSTM cell has 32 units. Finally, a fully connected layer of 16 neurons is used to interpret the features extracted by the LSTM hidden layer, before a final sigmoid layer is used to make predictions. Both the CNN and LSTM weigths are shared across time, resulting in a representation that scales for an arbitrary length sequences.

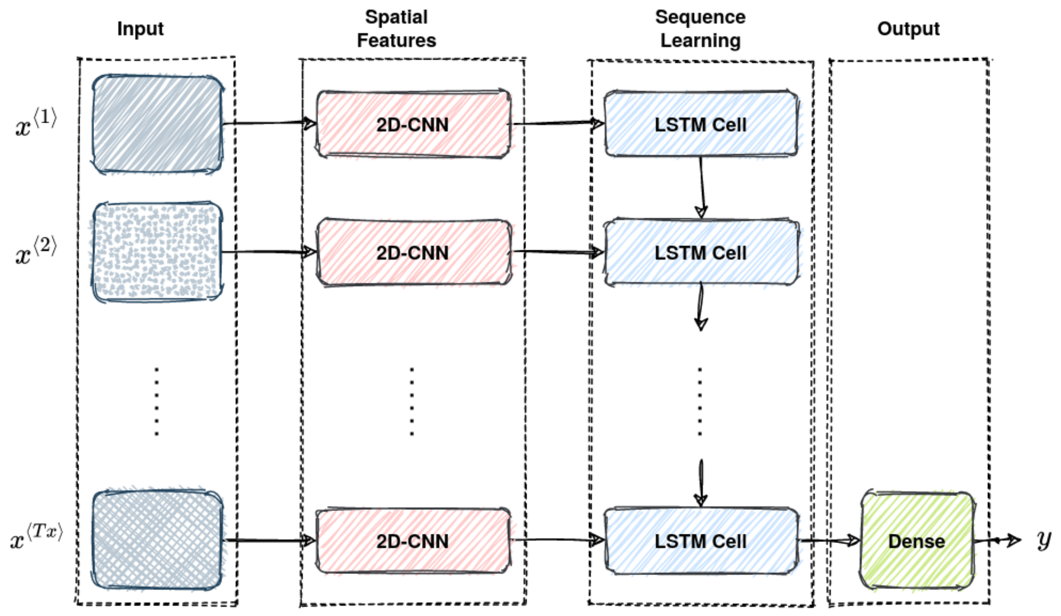


Figure 3.3: Architecture of the proposed CNN-LSTM recurrent neural network

Figure 3.4 zooms in the CNN architecture to present its details. It is classical CNN composed of three convolutional blocks. Each block consists of three operations: convolution layer, followed by a ReLU activation function that is fed to a MaxPooling layer. The output of the last block is flattened to be passed as input to the LSTM cell. The convolution layers are realized using 2-D kernels of size $(3,3)$ with a stride equal to $(1,1)$ and a zero padding. Each block has $16, 32, 32$ filters respectively. MaxPooling layers have pool sizes of $[(2,2),(3,3),(4,4)]$, and strides values of $[(2,2),(2,2),(3,3)]$, respectively

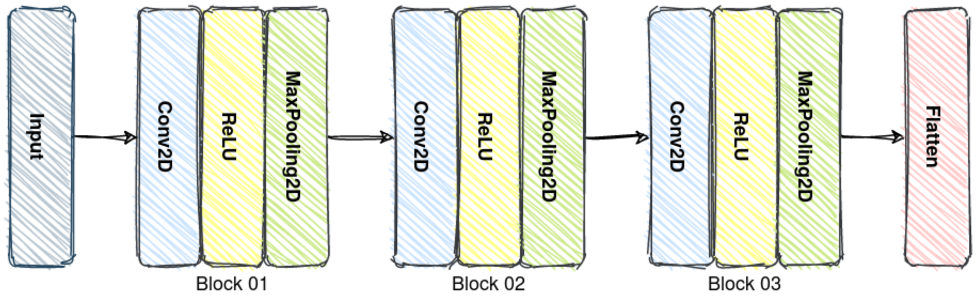


Figure 3.4: Architecture of the 2-D CNN

3.3.3 Results

Similar to the previous model, this second model was trained and tested in subject-specific way. For each subject, we compute the mean of N trials, where N is the number of seizures. Table 3.2 presents the results for each subject as well as the average. This model achieves testing area under the operation characteristic curve (AUC) of 0.873 on average.

Table 3.2: Summary of LCRN results

Subject	**No. of seizures**	**Interictal Duration (h)**	**Preictal Duration (h)**	**AUC**
Dog 1	4	80	4	0.603
Dog 2	7	84	7	0.999
Dog 3	12	240	12	0.962
Dog 4	14	134	14	0.803
Dog 5	6	75	6	0.999
Average	8.6	-	-	0.873

3.4 Discussion

First, we remind that the overall evaluation metric is the area under ROC (AUROC) as a single score. This metric is very used in medical applications as it gives us the flexibility to make a trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR), such that changing the threshold of classification will change the balance of predictions towards improving the TPR at the expense of FPR, or the reverse case. Therefore, if the built model is deployed in a mobile device, we will have the possibility to configure it paying more attention to the sensitivity or the false prediction rate, or both of them. Another advantage that makes it a popular diagnostic tool for classifiers on balanced and imbalanced binary prediction problems, as it is not biased to the majority or minority class.

> ″ROC analysis does not have any bias toward models that perform well on the majority class at the expense of the minority class—a property that is quite attractive when dealing with imbalanced data″ [86].

The first step in analysing our results is to compare the results given by the two models using the AUC scores. Per-subject AUC scores of the two proposed models as well as the average AUC are shown in Table 3.3.

Table 3.3: Per-subject AUC scores of the proposed models.

Subject	Proposed Model 1 (1-D CNN)	Proposed Model 2 (CNN-LSTM)
Dog 1	0.651	0.603
Dog 2	0.933	0.999
Dog 3	0.930	0.962
Dog 4	0.772	0.803
Dog 5	0.958	0.999
Average	0.848	0.873

We notice that the CNN-LSTM model outperforms the CNN model in all subjects except the first one. In addition the AUC of the second model is slightly better the first model's AUC with a difference of 0.025. Thus, we argue that spatio-temporal characteristic of the second models helps it to best model the data and get more meaningful features. Moreover, the two models do not perform well on subject 1's data, with AUCs of 0.651 , 0.603 respectively. More extensive analysis should be done to explain the reasons. We should verify if its data come from the same distribution as it can vary over time. A potential reason for that is that seizures were recorded at time far away from each other. During this time, the patient may be influenced by medications. Also, we recommend doing an advanced hyper-parameter optimization to find the ones that improves performance. Figure 3.5 depicts a bar plot of the AUC scores obtained by the two models as well as the average.

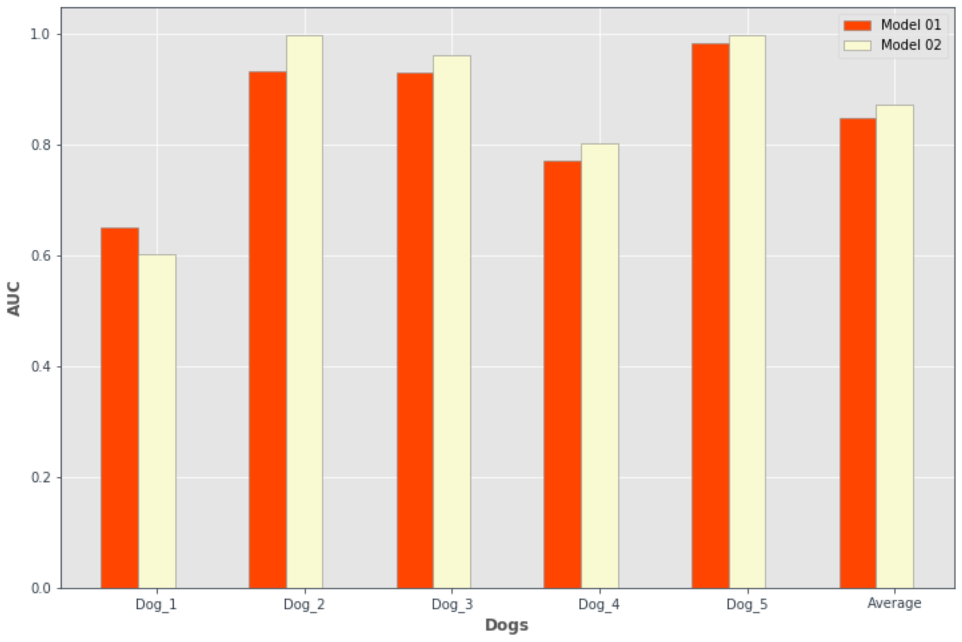


Figure 3.5: Bar Plot: Per-subject and average AUC scores of the proposed models

For reasonable evaluation of our proposed methods, we compare our achieved experimental results with previous works that have used the same dataset as shown in Table 3.4. Benjamin H. Brinkmann et al. [87] use frequency correlations as well as univariate spectral power in 11 adjacent frequency bands as features of the iEEG data, which are fed to an Support Vector Machine (SVM) classifier. The achieved mean area under the curve (AUC) was 0.72. Our two models yields superior results.

Table 3.4: Benchmark of recent seizure prediction approaches and this work

Year	**Ref**	**Predictive characteristics**	**Classification algorithm**	**AUC**
2015	B.Brinkmann et al. [87]	Univariate and multivariate frequency-related features	SVM	0.72
2016	Zisheng Zhang et al. [88]	Spectral power features Cross-correlation coefficients	AdBoost RBF SVM ANN	0.7603 0.8472 0.8884
2017	Yogatheesan Varatharajah et al. [89]	Univariate spectral power in band (PIB) Time domain correlations (TMCO) Spectral coherence (SPCO)	ANN SVM RFC	0.83
2020	Ours	Raw Data	CNN	0.848
2020	Ours	Raw Data	CNN-LSTM	0.873

In [87], Zisheng Zhang et al. proposed a seizure prediction algorithm based on a pipeline of three steps. First, they extract two different feature sets, spectral features including relative spectral powers and spectral power ratios and correlation features using cross-correlation coefficients. The two feature sets are then subjected to feature selection independently by classification and regression tree (CART). Three classifiers are used and tested on the selected features, which include AdaBoost, radial basis function kernel support vector machine (RBF-SVM), and artificial neural netwroks (ANN). The combined best results which use patient-specific feature sets achieve a mean AUC of 0.7603, 0.8472, and 0.8884 for AdaBoost, SVM, and ANN, respectively. The ANN model achieves better results than ours, but this work not only uses hand-crafted features but also the feature selection is patient-specific which is not practical for real-life application (more computational resources, latency, need domain-knowledge).
Yogatheesan Varatharajah et al. [89] computed three types of features to characterize EEG signals, univariate spectral power in band (PIB) features (1104-dimensional vector), time domain correlations (TMCO) features (120-dimensional vector) and spectral coherence (SPCO) between different pairs of channels (1820-dimensional vector). Then, they used PCA to maintain 90% of the variance. They used three types of classification algorithms Artificial Neural Network (ANN), Support Vector Machine (SVM) and Random Forest Classifier (RFC).They achieved an average Area Under ROC curve of 0.83. Besides the fact that we got better performance, their algorithm need to compute a large number of features.
In addition, we compare our results to those of the winning solutions in the "American Epilepsy Society Seizure Prediction Challenge" . The Receiver Operating Characteristics (ROC) Area Under Curve (AUC) was the metric used for ranking various solutions. The winning team used a blend of 11 models and more than 4000 hand-crafted iEEG features, and achieved an average AUC of 0.839. It is, however impractical to use such a computationally-intensive process for real-time applications. More, it needs a good level of signal processing domain-knowledge to choose theses features. Our models obtain a slightly higher seizure prediction AUC score, but with much faster in obtaining the results. Thus, it is more suitable.

Table 3.5: AUC scores for the proposed method and top finishing contestants in the competition.

Top 10 Competition Scores (2014) [71] min-max	Proposed Model 1 (CNN)	Proposed Model 2 (CNN-LSTM)
0.785-0.839	0.848	0.873

3.5 Conclusion

Along this chapter, we presented the first model based on 1-D fully convolutional neural network for both feature extraction and classification segments into preictal and interictal. The data preparation and the evaluation results have been also presented. This model achieves a mean AUC ROC of 0.848.
A second model that has been implemented, combines the advantages of CNN in spatial features extraction and LSTM in sequence modeling., known as CNN-LSTM or Long-term Recurrent Convolutional Network (LRCN) for consistency. This model is better than first one, on modelling the sequential structure of the data, and improves the classification performance to a mean AUC ROC of 0.873.
Finally, we have shown that our models achieve the state-of-the-art performance, with minimum preprocessing and no manual feature extraction. For more robust evaluation, we suggests to train and evaluate these models on a larger dataset.

Conclusion

Seizure forecasting systems have the promise to help patients with epilepsy and improve their quality of life. That would be beneficial especially with patients that have drug-resistant epilepsy. The goal would be to create a wearable EEG device which would alert patients, family members, and doctors to imminent seizures. This has the potential to enhance the safety of patients and perhaps to allow some patients to take medications only when needed and not chronically. In order for EEG-based seizure forecasting systems to work effectively, we initiated this project to build a deep learning model that can detect the preictal state.

Along this project, we tried to exploit the advances in deep learning domain to build an efficient model that can distinguish between preictal and interictal states. We have proposed a first model based on Convolutional Neural Network (CNN) architecture that uses 1-D kernels. The CNN model we proposed has an efficient fully convolutional network architecture that automatically learns distinctive iEEG features. This model used the raw data without any preprocessing step or feature extraction. A second model that has been proposed is based on Long-term Recurrent Convolutional Network (LRCN), also called as CNN-LSTM for simplicity. It takes the advantages of both Convolutional Neural Network and Long Short-Term Memory for better modeling of the spatiotemporal structure of the EEG signals. Using Leave-One-Out cross-validation technique to test the proposed models proves the robustness and generality of our models against variation across various seizure types. Our experimental results and the comparison with previous work demonstrate that the proposed models are efficient and suitable for real-time application of seizure prediction. This is by achieving a mean Area Under Curve (AUC) Receiver Operating Characteristics (ROC) of 0.848 for the first model, and 0.873 for the second one.

Future work The implemented models in this dissertation showed some promising results. However, they need to be extensively tested on bigger dataset for more fair evaluation, though only a few iEEG datasets are available. For example, the EPILEPSIAE database that contains more than 2500 epileptic seizures for 250 patients [90], but it is not open-source now. Moreover, it would be interesting if we test these models on scalp EGG signals, which are more prone to noise but easier to obtain.

Furthermore, it is encouraging to perform an advanced hyper-parameters optimization to find those that suit well models. The length of the window, the amount of overlapping, the number of sequential inputs and many other parameters have not been tuned. Thus, bad results of these hyper-parameters can lead to low performance.

We should mention that it is strictly mandatory to have high performance computers to be able to carry out these experiments as in our case very poor hardware was available. During the training process, we faced a serious problem of slow computational speed as well as a very limited memory where the data of one subject can not be uploaded once in the available memory.

Bibliography

[1] World Health Organization (WHO), *Epilepsy*, [accessed 13-August-2020]. [Online]. Available: `https://www.who.int/news-room/fact-sheets/detail/epilepsy`.

[2] Wassila Benhamed, Journal Elmoudjahid, *Journée mondiale de l'épilepsie : 400.000 cas en algérie*, [accessed 13-August-2020]. [Online]. Available: `http://www.elmoudjahid.com/fr/actualites/90338`.

[3] P. Kwan, A. Arzimanoglou, A. T. Berg, M. J. Brodie, W. Allen Hauser, G. Mathern, S. L. Moshé, E. Perucca, S. Wiebe, and J. French, "Definition of drug resistant epilepsy: Consensus proposal by the ad hoc task force of the ilae commission on therapeutic strategies: Definition of drug resistant epilepsy," *Epilepsia*, vol. 51, no. 6, 1069–1077, 2009, ISSN: 00139580, 15281167. DOI: `10.1111/j.1528-1167.2009.02397.x`.

[4] J. F. Annegers, "United states perspective on definitions and classifications," *Epilepsia*, vol. 38, no. s11, S9–S12, 1997. DOI: `10.1111/j.1528-1157.1997.tb06137.x`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1528-1157.1997.tb06137.x`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1528-1157.1997.tb06137.x`.

[5] C. E. Stafstrom and L. Carmant, "Seizures and epilepsy: An overview for neuroscientists," *Cold Spring Harbor Perspectives in Medicine*, vol. 5, no. 6, a022426–a022426, 2015, ISSN: 2157-1422. DOI: `10.1101/cshperspect.a022426`.

[6] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1–6.

[7] S. Angra and S. Ahuja, "Machine learning and its applications: A review," in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, 2017, pp. 57–60.

[8] G. K. Bergey, "Neurostimulation in the treatment of epilepsy," *Experimental Neurology*, vol. 244, 87–95, 2013, ISSN: 00144886. DOI: `10.1016/j.expneurol.2013.04.004`.

[9] Epilepsy Foundation, *Neurostimulation in the treatment of epilepsy, may 31st 2017*, [accessed 13-August-2020]. [Online]. Available: `https://www.epilepsy.com/article/2017/5/neurostimulation-treatment-epilepsy`.

[10] R. S. Fisher, W. v. E. Boas, W. Blume, C. Elger, P. Genton, P. Lee, and J. Engel, "Epileptic seizures and epilepsy: Definitions proposed by the international league against epilepsy (ilae) and the international bureau for epilepsy (ibe)," *Epilepsia*, vol. 46, no. 4, 470–472, 2005, ISSN: 0013-9580, 1528-1167. DOI: `10.1111/j.0013-9580.2005.66104.x`.

[11] J. G. Thundiyil, T. E. Kearney, and K. R. Olson, "Evolving epidemiology of drug-induced seizures reported to a poison control center system," *Journal of Medical Toxicology*, vol. 3, no. 1, 15–19, 2007, ISSN: 1556-9039, 1937-6995. DOI: `10.1007/BF03161033`.

[12] R. S. Fisher, J. H. Cross, C. D'Souza, J. A. French, S. R. Haut, N. Higurashi, E. Hirsch, F. E. Jansen, L. Lagae, S. L. Moshé, and et al., "Instruction manual for the ilae 2017 operational classification of seizure types," *Epilepsia*, vol. 58, no. 4, 531–542, 2017, ISSN: 00139580. DOI: `10.1111/epi.13671`.

[13] N. Hazarika, J. Z. Chen, A. C. Tsoi, and A. Sergejew, "Classification of eeg signals using the wavelet transform," *Signal Processing*, vol. 59, no. 1, 61–72, 1997, ISSN: 01651684. DOI: `10.1016/S0165-1684(97)00038-8`.

[14] Y. Z. a. Siuly Siuly Yan Li, *EEG Signal Analysis and Classification: Techniques and Applications*, 1st ed., ser. Health Information Science. Springer International Publishing, 2016, ISBN: 978-3-319-47652-0, 978-3-319-47653-7. [Online]. Available: `http://gen.lib.rus.ec/book/index.php?md5=b2e245c0701c1e31410ee6d4e1b3ab5f`.

[15] *Electroencephalography: basic principles, clinical applications, and related fields*, 5th ed. Lippincott Williams Wilkins, 2005, ISBN: 9780781751261.

[16] D. P. Subha, P. K. Joseph, R. Acharya U, and C. M. Lim, "Eeg signal analysis: A survey," *Journal of Medical Systems*, vol. 34, no. 2, 195–212, 2010, ISSN: 0148-5598, 1573-689X. DOI: `10.1007/s10916-008-9231-z`.

[17] K. Blinowska and P. Durka, "Electroencephalography (eeg)," in *Wiley Encyclopedia of Biomedical Engineering*, M. Akay, Ed. John Wiley Sons, Inc., 2006, ebs0418, ISBN: 9780471740360. DOI: `10.1002/9780471740360.ebs0418`. [Online]. Available: `http://doi.wiley.com/10.1002/9780471740360.ebs0418`.

[18] S. Supriya, S. Siuly, and Y. Zhang, "Automatic epilepsy detection from eeg introducing a new edge weight method in the complex network," *Electronics Letters*, vol. 52, Jul. 2016. DOI: `10.1049/el.2016.1992`.

[19] T. N. Alotaiby, S. A. Alshebeili, T. Alshawi, I. Ahmad, and F. E. Abd El-Samie, “Eeg seizure detection and prediction algorithms: A survey,” *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, p. 183, 2014, ISSN: 1687-6180. DOI: `10.1186/1687-6180-2014-183`.

[20] S. Siuly and Y. Li, “Discriminating the brain activities for brain–computer interface applications through the optimal allocation-based approach,” *Neural Computing and Applications*, vol. 26, no. 4, 799–811, 2015, ISSN: 0941-0643, 1433-3058. DOI: `10.1007/s00521-014-1753-3`.

[21] U. R. Acharya, S. Vinitha Sree, G. Swapna, R. J. Martis, and J. S. Suri, “Automated eeg analysis of epilepsy: A review,” *Knowledge-Based Systems*, vol. 45, 147–165, 2013, ISSN: 09507051. DOI: `10.1016/j.knosys.2013.02.014`.

[22] I. Osorio, M. G. Frei, S. Sunderam, J. Giftakis, N. C. Bhavaraju, S. F. Schaffner, and S. B. Wilkinson, “Automated seizure abatement in humans using electrical stimulation,” *Annals of Neurology*, vol. 57, no. 2, 258–268, 2005, ISSN: 0364-5134, 1531-8249. DOI: `10.1002/ana.20377`.

[23] M. Saab and J. Gotman, “A system to detect the onset of epileptic seizures in scalp eeg,” *Clinical Neurophysiology*, vol. 116, no. 2, 427–442, 2005, ISSN: 13882457. DOI: `10.1016/j.clinph.2004.08.004`.

[24] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: A review,” *Data Mining and Knowledge Discovery*, vol. 33, pp. 917–963, 2019.

[25] T. Maiwald, M. Winterhalder, R. Aschenbrenner-Scheibe, H. U. Voss, A. Schulze-Bonhage, and J. Timmer, “Comparison of three nonlinear seizure prediction methods by means of the seizure prediction characteristic,” 2004.

[26] I. Osorio, M. G. Frei, and S. B. Wilkinson, “Real-time automated detection and quantitative analysis of seizures and short-term prediction of clinical onset,” *Epilepsia*, vol. 39, no. 6, 615–627, 1998, ISSN: 0013-9580, 1528-1167. DOI: `10.1111/j.1528-1157.1998.tb01430.x`.

[27] K. Rasheed, A. Qayyum, J. Qadir, S. Sivathamboo, P. Kwan, L. Kuhlmann, T. J. O’Brien, and A. Razi, “Machine learning for predicting epileptic seizures using eeg signals: A review,” *IEEE reviews in biomedical engineering*, vol. PP, 2020.

[28] F. Mormann, T. Kreuz, C. Rieke, R. G. Andrzejak, A. Kraskov, P. David, C. E. Elger, and K. Lehnertz, “On the predictability of epileptic seizures,” *Clinical Neurophysiology*, vol. 116, no. 3, 569–587, 2005, ISSN: 13882457. DOI: `10.1016/j.clinph.2004.08.025`.

[29] Cyrille Kone, *A layman's guide to deep neural networks*, [accessed 13-August-2020]. [Online]. Available: `https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb`.

[30] B. R. Reed and R. J. MarksII, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 1999, `https://mitpress.mit.edu/books/neural-smithing`.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[32] K. Chandra, E. Meijer, S. Andow, E. Arroyo-Fang, I. Dea, J. George, M. Grueter, B. Hosmer, S. Stumpos, A. Tempest, and S. Yang, "Gradient descent: The ultimate optimizer," *ArXiv*, vol. abs/1909.13371, 2019.

[33] J. Duda, "SGD momentum optimizer with step estimation by online parabola model," *CoRR*, vol. abs/1907.07063, 2019. arXiv: `1907.07063`. [Online]. Available: `http://arxiv.org/abs/1907.07063`.

[34] A. Botev, G. Lever, and D. Barber, "Nesterov's accelerated gradient and momentum as approximations to regularised update descent," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1899–1903.

[35] N. Zhang, D. Lei, and J. Zhao, "An improved adagrad gradient descent optimization algorithm," *2018 Chinese Automation Congress (CAC)*, pp. 2359–2362, 2018.

[36] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015, 2014. [Online]. Available: `http://arxiv.org/abs/1412.6980`.

[37] S. Ruder, "An overview of gradient descent optimization algorithms," *ArXiv*, vol. abs/1609.04747, 2016.

[38] S. K. Kumar, "On weight initialization in deep neural networks," *ArXiv*, vol. abs/1704.08863, 2017.

[39] M. Skórski, "Revisiting initialization of neural networks," *ArXiv*, vol. abs/2004.09506, 2020.

[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[41] J. Kukacka, V. Golkov, and D. Cremers, "Regularization for deep learning: A taxonomy," *ArXiv*, vol. abs/1710.10686, 2017.

[42] Wikipedia, *Dilution (neural networks)*, [accessed 13-August-2020]. [Online]. Available: `https://en.wikipedia.org/wiki/Dilution_(neural_networks)`.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, 1929–1958, Jan. 2014, ISSN: 1532-4435.

[44] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, 289–315, 2007, ISSN: 0176-4276, 1432-0940. DOI: 10.1007/s00365-006-0663-2.

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[46] F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional neural network," *CoRR*, vol. abs/1905.03288, 2019. arXiv: 1905.03288. [Online]. Available: http://arxiv.org/abs/1905.03288.

[47] X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, 2020.

[48] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *CoRR*, vol. abs/1807.05511, 2018. arXiv: 1807.05511. [Online]. Available: http://arxiv.org/abs/1807.05511.

[49] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *CoRR*, vol. abs/1508.06576, 2015. arXiv: 1508.06576. [Online]. Available: http://arxiv.org/abs/1508.06576.

[50] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song, "Neural style transfer: A review," *CoRR*, vol. abs/1705.04058, 2017. arXiv: 1705.04058. [Online]. Available: http://arxiv.org/abs/1705.04058.

[51] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. arXiv: 1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167.

[52] A. Garg and M. Agarwal, "Machine translation: A literature review," *CoRR*, vol. abs/1901.01122, 2019. arXiv: 1901.01122. [Online]. Available: http://arxiv.org/abs/1901.01122.

[53] M. A. Anusuya and S. K. Katti, "Speech recognition by machine, A review," *CoRR*, vol. abs/1001.2267, 2010. arXiv: 1001.2267. [Online]. Available: http://arxiv.org/abs/1001.2267.

[54] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *CoRR*, vol. abs/1810.04020, 2018. arXiv: 1810.04020. [Online]. Available: http://arxiv.org/abs/1810.04020.

[55] J. Briot, G. Hadjeres, and F. Pachet, "Deep learning techniques for music generation - A survey," *CoRR*, vol. abs/1709.01620, 2017. arXiv: 1709.01620. [Online]. Available: http://arxiv.org/abs/1709.01620.

[56] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, pp. 107–116, Apr. 1998. DOI: 10.1142/S0218488598000094.

[57] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *CoRR*, vol. abs/1211.5063, 2012. arXiv: `1211.5063`. [Online]. Available: `http://arxiv.org/abs/1211.5063`.

[58] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[59] H. Khan, L. V. Marcuse, M. C. Fields, K. Swann, and B. Yener, "Focal onset seizure prediction using convolutional networks," *IEEE Transactions on Biomedical Engineering*, vol. 65, pp. 2109–2118, 2018.

[60] N. D. Truong, A. D. Nguyen, L. Kuhlmann, M. R. Bonyadi, J. Yang, and O. Kavehei, "A generalised seizure prediction with convolutional neural networks for intracranial and scalp electroencephalogram data analysis," *ArXiv*, vol. abs/1707.01976, 2017.

[61] R. Hussein, M. O. Ahmed, R. K. Ward, Z. J. Wang, L. Kuhlmann, and Y. Guo, "Human intracranial eeg quantitative analysis and automatic feature learning for epileptic seizure prediction," *ArXiv*, vol. abs/1904.03603, 2019.

[62] K. M. Tsiouris, V. C. Pezoulas, M. E. Zervakis, S. Konitsiotis, D. D. Koutsouris, and D. I. Fotiadis, "A long short-term memory deep learning network for the prediction of epileptic seizures using eeg signals," *Computers in biology and medicine*, vol. 99, pp. 24–37, 2018.

[63] H. G. Daoud and M. Bayoumi, "Efficient epileptic seizure prediction based on deep learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, pp. 804–813, 2019.

[64] *Python*, `https://www.python.org/`, [accessed 13-August-2020].

[65] *Numpy*, `https://numpy.org/contribute/`, [accessed 13-August-2020].

[66] *Scipy*, `https://www.scipy.org/`, [accessed 13-August-2020].

[67] *Matplotlib*, `https://matplotlib.org/`, [accessed 13-August-2020].

[68] *Sckitlearn*, `https://scikit-learn.org/`, [accessed 13-August-2020].

[69] *Tensorflow python api*, `https://www.tensorflow.org/api_docs/python/tf`, [accessed 13-August-2020].

[70] *Keras*, `https://keras.io/`, [accessed 13-August-2020].

[71] *American epilepsy society seizure prediction challenge*, https://www.kaggle.com/c/seizure-prediction/overview.

[72] J. Howbert, E. Patterson, S. Stead, B. Brinkmann, V. Vasoli, D. Crepeau, C. Vite, B. Sturges, V. Ruedebusch, J. Mavoori, K. Leyde, W. Sheffield, B. Litt, and G. Worrell, "Forecasting seizures in dogs with naturally occurring epilepsy," English (US), *PLoS One*, vol. 9, no. 1, Jan. 2014, ISSN: 1932-6203. DOI: `10.1371/journal.pone.0081920`.

[73] T. Fawcett, "Roc graphs: Notes and practical considerations for data mining researchers," *ReCALL*, vol. 31, pp. 1–38, Jan. 2004.

[74] A. Karpathy, *A recipe for training neural networks*, http://karpathy.github.io/2019/04/25/recipe/.

[75] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," *CoRR*, vol. abs/1305.1707, 2013. arXiv: `1305.1707`. [Online]. Available: `http://arxiv.org/abs/1305.1707`.

[76] , "Cross-validation," in *Encyclopedia of Machine Learning and Data Mining*, 2010.

[77] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *CoRR*, vol. abs/1811.12808, 2018. arXiv: `1811.12808`. [Online]. Available: `http://arxiv.org/abs/1811.12808`.

[78] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *ArXiv*, vol. abs/2007.15745, 2020.

[79] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *ArXiv*, vol. abs/2003.05689, 2020.

[80] A. Ng, *Structuring machine learning projects online course*, https://www.deeplearning.ai/deep-learning-specialization/.

[81] B. Abbaszadeh and M. Yagoub, "Optimum window size and overlap for robust probabilistic prediction of seizures with ieeg," *2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pp. 1–5, 2019.

[82] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2014.

[83] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: `1512.03385`. [Online]. Available: `http://arxiv.org/abs/1512.03385`.

[84] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *CoRR*, vol. abs/1411.4389, 2014. arXiv: `1411.4389`. [Online]. Available: `http://arxiv.org/abs/1411.4389`.

[85] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," ser. MM '15, Brisbane, Australia: Association for Computing Machinery, 2015, 1307–1310, ISBN: 9781450334594. DOI: `10.1145/2733373.2806333`. [Online]. Available: `https://doi.org/10.1145/2733373.2806333`.

[86] *Imbalanced Learning: Foundations, Algorithms, and Applications*, Haibo He. Wiley-IEEE Press, 2013, p. 27, ISBN: 9781118074626,9781118646106.

[87] B. Brinkmann, E. Patterson, C. Vite, V. Vasoli, D. Crepeau, M. Stead, J. Howbert, V. Cherkassky, J. Wagenaar, B. Litt, and G. Worrell, “Forecasting seizures using intracranial eeg measures and svm in naturally occurring canine epilepsy,” *PLoS ONE*, vol. 10, 2015.

[88] Z. Zhang and K. Parhi, “Seizure prediction using long-term fragmented intracranial canine and human eeg recordings,” *2016 50th Asilomar Conference on Signals, Systems and Computers*, pp. 361–365, 2016.

[89] Y. Varatharajah, R. Iyer, B. M. Berry, G. Worrell, and B. Brinkmann, “Seizure forecasting and the preictal state in canine epilepsy,” *International journal of neural systems*, vol. 27 1, p. 1 650 046, 2017.

[90] *The european epilepsy database*, `http://epilepsy-database.eu/`, [accessed 13-August-2020].