CUFE

Communication and Computer Engineering

Credit Hours System

Spring / 2019

Senior-2 Level

Graduation Project-2

CCEN481

Graduation Project-2

“Biomedical Signal Processing For Seizure Prediction”

Final Report

Submitted by:

Nourhan Ahmed Abdel Fattah

Ingy Alaa Hafez Mostafa

Mostafa Nader Ibrahim

Abdelrahman Magdy Saad

Mohamed Ibrahim Saad

Michael Khalil Malak

Supervised by:

Dr Hassan Mostafa

Acknowledgment

Listed below are the names of the people who provided us with significant help in developing our graduation project in addition to our sponsor National Telecommunications Regulatory Authority (NTRA). To all we extend our sincere thanks.

****

Dr. Hassan Mostafa

Eng. Michael Hany

One Lab

Executive Summary

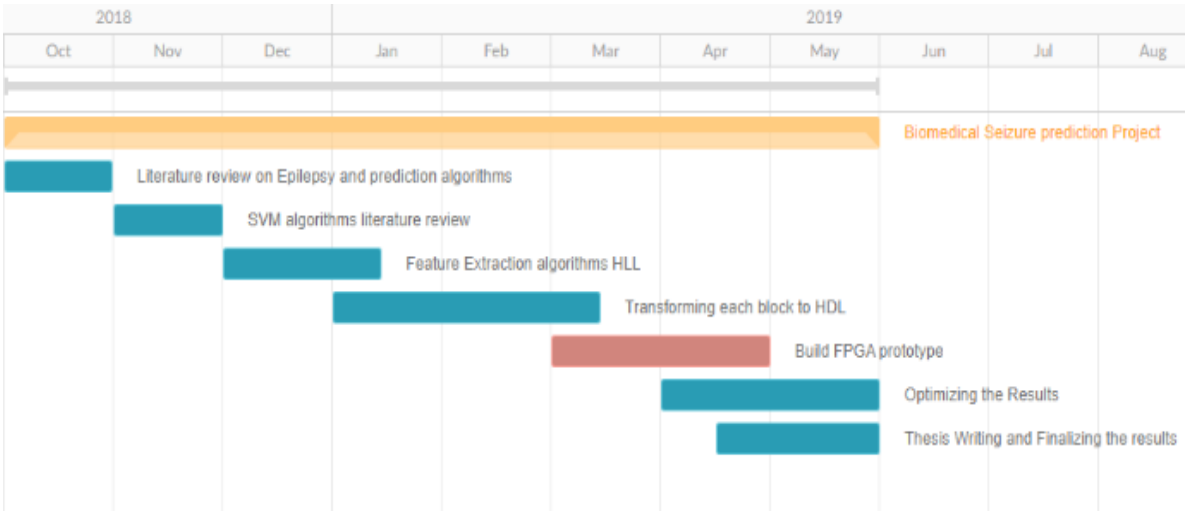
Epileptic seizures occur due to disorder in the brain's functionality which affects the patient's health. In addition, epilepsy is not usually confined to a single area of the brain but rather it is an epileptic network where different areas of the brain interact synchronously causing these pathological spikes or seizures. Prediction of epileptic seizures before the beginning of the onset is quite useful for preventing the seizure without any medical interference.

Although, medication is the most common treatment for people with epileptic seizures, we introduce the idea of feeding our designed chip with a machine learning algorithm with the required decision making to make the prediction and treatment of seizure much more comfortable and end patients' suffering for good.

The proposed work in the thesis is to design a chip for seizure prediction. Support Vector Machine with additional computational methods are used for that purpose. Since our problem is binary classification between preictal and normal periods, determining the right features along with the SVM kernel function is a first step for implementing the chip. Using Electroencephalograms (EEG) signals from EEG CHB-MIT dataset enabled us to simulate different feature-kernel combinations and decide accordingly on the best combination based on prediction performance.

Moreover, hardware implementation was performed after designing the RTL in the aim of quantifying the performance of the algorithm in terms of power consumption and area. Using FPGA and Vivado designing tool, we tested the different combinations of features and SVM kernel in order to decide on the combination that yields the highest prediction performance.

Gantt Chart



The shown plan illustrates the project's time schedule that helped us in distributing the tasks of the project along with time.

First, we tried to know more about the biomedical background regarding epilepsy that helped us in understanding the types, cases and treatments of epilepsy. We had to know the types, measurements and periods of EEG signals that are considered the monitoring method of the patient's brain activities and to know the difference between the detection and prediction algorithms. Since seizure prediction is considered a binary classification between the pre-ictal and the non-pre-ictal periods, we had to differentiate between the algorithms to choose the most suitable one which is SVM algorithm to separate between the two classes. Then, we started the project simulations' phase which is divided into two stages: software and hardware implementation.

In the software simulations, we used Matlab to choose the most suitable features and kernel to provide the best performance in terms of sensitivity, specificity and accuracy. We chose CHB-MIT Scalp's dataset to works with since MIT data provided us with 23 different patients with more than one channel collecting data (EEG signals) from the patients' brains.

In the hardware implementation, we implemented features and classifiers using VHDL and Verilog codes in Modelsim. We used Vivado designing tool to build an FPGA prototype for mapping the signals to I/O ports of the FPGA, synthesize the HDL down into a configuration file or bit stream and calculate the most important aspects of the hardware implementation in terms of power consumption, total delay and the area.

Last, we documented the results that yielded from the software simulation and the hardware implementation.

Abbreviations

RTL **R**egister **T**ransfer **L**evel

EEG **E**lectro**e**ncephalo**g**ram

FPGA **F**ield **P**rogrammable **G**ate **A**rray

ILAE **I**nternational **L**eague **A**gainst **E**pilepsy

AED **A**nti-**E**pileptic **D**rugs

DBS **D**eep **B**rain **S**timulation

VNS **V**agus **N**erve **S**timulation,

NREM **N**on-**R**apid **E**ye **M**ovement

ICA **I**ndependent **C**omponent **A**nalysis,

MIT **M**assachusetts **I**nstitute of **T**echnology

CHB **C**hildren's **H**ospital **B**oston

EDF **E**uropean **D**ata **F**ormat

SVM **S**upport **V**ector **M**achine

ANN **A**rtificial **N**eural **N**etwork

GA **G**radient **A**scent

SGD **S**tochastic **G**radient **D**escent

KKT **K**arush–**K**uhn–**T**ucker

RBF **R**adial **B**asis **K**ernel

QP **Q**uadratic **P**roblem

TP **T**rue **P**ositive

TN **T**rue **N**egative

FP **F**alse **P**ositive

FN **F**alse **N**egative

CL **C**oast **L**ine

AM **A**bsolute **M**ean

SD **S**tandard **D**eviation

RMS **R**oot **M**ean **S**quare

RSP **R**elative **S**pectral **P**ower

VC **V**ariation **C**oefficient

HDL **H**ardware **D**escription **L**anguage

CRA **C**arry **R**ipple **A**dder

ASIC **A**pplication **S**pecific **I**ntegrated **C**ircuit

CPU **C**entral **P**rocessing **U**nit

IDE **I**ntegrated **D**esign **E**nvironment

CLB **C**onfigurable **L**ogic **B**lock

CUFE **C**airo **U**niversity **F**aculty of **E**ngineering

JAC **J**apan-**A**frica **C**onference

CRC **Cr**yptography **C**ompetition

AES **A**dvanced **E**ncryption **S**tandard

NTRA **N**ational **T**elecom **R**egulatory **A**uthority

Table of Content

Table of Figures

Table of Tables

1

BIOMEDICAL BACKGROUND

1.1 Introduction

1.1.1 Motivation

More than 50 million globally are not able to lead a normal life since they suffer from epilepsy. [2] Epilepsy, simply put, is an abnormality in the central nervous system that leads to unplanned-for seizures. Medicine is only successful with 66% of the cases. Surgery is the preceding option. [3] However, not all epileptic cases are eligible for surgery as the epileptogenic zone needs to be localized. Surgery at the temporal lobe is a mere 60% success rate and a 35% success rate in case of operation at the extratemporal lobe. [4]

Near 15.5 billion dollars per day are paid by American patients for their treatment. [5] 80% of epileptic patients are living from developing world according to the world health organization. [6] With the low success rate of traditional treatment methods for epilepsy and high economic demand -that can fund research in that area- for a viable treatment option. A possible option would be data processing for seizure prediction using the right Seizure detection algorithms along with minimal hardware design to conserve battery use.

1.1.2 Proposed work

Our project's main aim is to develop a hardware chip (FPGA prototype) that can be either implanted in the human's brain or embedded in a head-cap to predict seizure periods by the help of machine learning algorithms (Support Vector Machine) to distinguish the pre-ictal (pre-seizure) periods from other periods.

Our product is designed to be sensitive, accurate, affordable, and a low power consumer to prolong the need for patients undergoing complex and expensive surgeries to change the device's battery as much as possible. The algorithms for feature extraction and classification is presented in this thesis that ultimately results in a true positive seizure prediction 95% of the time with a 30 minutes' window before the seizure.

1.1.3 Organization of the thesis

The thesis is organized as follows: Chapter-1: gives an overview as background knowledge on epilepsy and EEG. Chapter-2: expands on our approach in the planning, tool, and processes for the graduation project. Chapter-3: gives an overview of the machine learning algorithms and techniques relevant to seizure prediction. Chapter-4: explains the methodology and decisions taken in our software simulation phase. Chapter-5: expands on the to-be-implemented RTL design and the results of the hardware simulation phase. Chapter-6: Concludes the overall project's results and presents possible future work preceding this project.

1.2 Literature Review

This chapter aims to provide the necessary background knowledge that will facilitate the scope of the presented work. Section 1.2.1 briefly gives an overview of epilepsy, its causes, types and the traditional methods used to treat it. Section 1.2.2 expands on the type of signal measures from the brain, EEG signal that is a major tool for detecting and predicting epilepsy.

1.2.1 Epilepsy

If two or more unprovoked seizures occur to a person over a time span of more than 24 hours, then that person is diagnosed as an epilepsy patient. [2] About 1% of the global population are epileptic patients with an estimate of their third are not respondent to medicine. Even for those who use a medicine, their quality of life is affected tragically as the seizure occurs unexpectedly, potentially causing a life-threatening experience.

Structural malfunctions of the brain cause the electrical surge that causes a seizure. [7] These malfunctions cause frequent spontaneous seizures and thus epilepsy, a long-lasting neurological disorder.

1.2.1.1 Types

Epilepsy can refer to a broad variety of symptoms; thus, it can be variously. It could be classified according to the causes of the seizure; whether it's genetic or a result of trauma, stroke, brain tumor or infection, or of an unknown cause. Standardized classification of seizure was introduced in 1981 by International League Against Epilepsy, ILAE which rather than the anatomical position of the seizures, the classification depended on EEG signal recordings and clinical symptoms. [8]

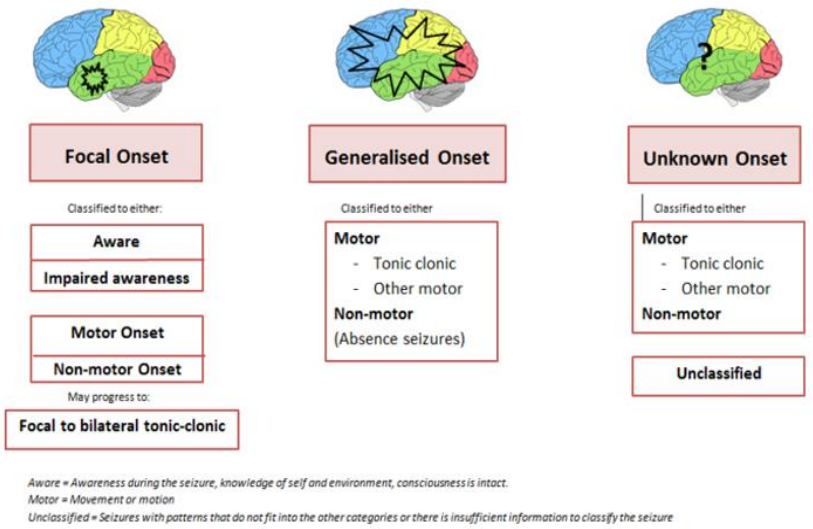


Figure 1-1 Seizure Classification Scheme - ILAE 2017 [5]

1.2.1.1.1 Focal seizures

Focal, or partial, seizures occur in only one hemisphere as opposed to the generalized that could occur in any of the hemispheres. It's shown that about 60% of epileptic patients suffer from focal seizures. [8]

Focal seizures are then subdivided depending on what the patient experiences.

- **Complex Seizure**

According to ILAE 2011 classification, if a focal seizure causes a change in consciousness, it's referred to as a "complex seizure".

- **Simple Seizure**

As opposed to a "complex seizure" it's a focal seizure which is accompanied by motor and autonomic symptoms without losing consciousness. [8]

1.2.1.1.2 Generalized seizures

On the other hand, generalized seizures affect both of the two hemispheres, and are subdivided into six classes in regards to the duration and effect on motor functions; absence, myoclonic, tonic, clonic, and atonic seizures.

It should be noted that there exists a third type of seizures which is neither generalized nor focal; unknown seizures.

1.2.1.2 Traditional Treatment Methods

Medication, surgery, and neuromodulation are available as a form of treatment but none of them is a 100% successful method of treatment. Anti-epileptic drugs, AED, is successful in about 66% of the cases. [3] However, with side effects such as depression and rash. AED aims mainly to prevent the excessive inflow of positive ions by targeting specific ions channels and thus prevent an imbalance of excitation over cell activity. A patient who is not responded to AED would be suffering from refractory epilepsy and thus would typically be subjected to presurgical evaluation.

Surgery would be possible in the case that the epileptogenic zone is localized. Surgery is only a 60% success rate for an epileptogenic zone at the temporal lobe, and a mere 35% success rate in case it was at the extratemporal lobe.[4] In case that surgery was not a viable option, neuromodulation could be the last resort. Neuromodulation methods such as Deep Brain Stimulation, DBS, or Vagus Nerve Stimulation, VNS, could relatively be successful options.

1.2.2 EEG measurements

Electroencephalography, EEG, is a monitoring method to record electrical brain activity. The human brain consists of hundreds of billions of neurons. The cerebellum alone is estimated to contain 55-70 billion neurons. [9] Each of these neurons are connected with several thousand other neurons by synapses; a structure that allows a neuron to pass (fires) electrical signal to another neuron. The action of "firing" an electrical signal

known as *Action Potential*. The EEG device aims to sense groups of action potentials at specific locations in the brain.

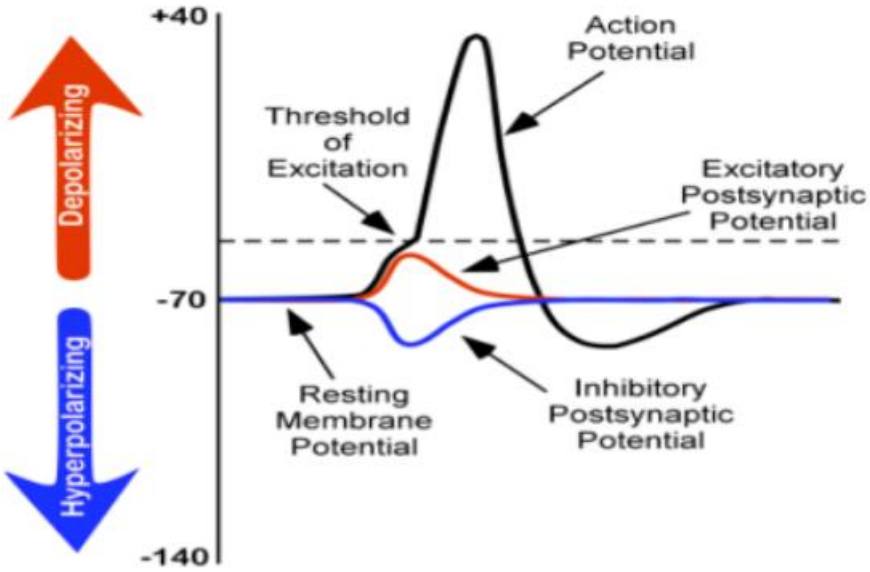


Figure 1-2 Action Potential Curve

1.2.2.1 EEG Types

Caused by multiple action potentials, the extracellular field potentials in the brain naturally fluctuates and measuring these fluctuations is key to observe patterns in the brain. There are several types of EEG signals.

1.2.2.1.1 EEG

Electroencephalogram, EEG, enables us to measure the aforementioned fluctuating potentials by simply placing electrodes on the subject's skull. The fluctuations, observed from the EEG side measured as post-synaptic potentials, are the summation of activities of a large group of neurons. These large group of neurons needs to be positioned in a certain way -parallel to each other- to be measurable by the electrode sensors. The measured signals are usually amplified 10000 times and converted from analog to digital typically with a sampling rate of 1000 Hz. [9]

1.2.2.1.2 iEEG

Another type of EEG measurement is intracranial EEG, iEEG, where electrodes are implanted on the cortex in surgery as opposed to being placed on its surface. A major drawback is a fact that localizing the electrodes with high precision is very critical. However, iEEG recordings have lower spatial and temporal resolutions, lower than one millimeter and one millisecond, thus more accurate measurements as it can record for a smaller scale of neurons. [11]

1.2.2.2 Electrodes Placement

EEG sensing could be satisfied by sensing the cerebral cortex only since it satisfies that a large group of neurons are parallel; pyramidal neurons. However, EEG recordings are improved by increasing the number of electrodes at different parts of the skull and thus increasing signal sensitivity. The number of electrodes could reach up to 27 to 32 electrodes for clinical use and 256 electrodes for research purposes. These are known in the literature as high-density EEG.

Electrodes positioning is almost standardized to allow for a fair comparison between studies. a typical standard used in scalp EEG measurements is the 10/20 system. In such a system, the electrodes are placed 10% then 20% away from the consecutive areas of the brain as shown in Figure 11. Generally, electrodes are labeled according to their positions on the scalp. A, F, C, O, P and T respectively stand for auricular, frontal, central, occipital, parietal and temporal. [10]

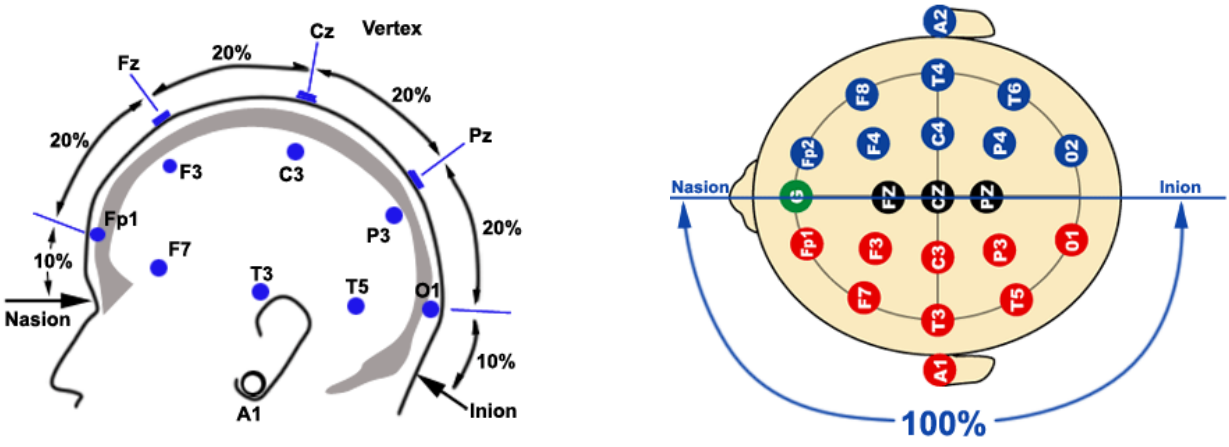


Figure 1-3 10/20 Electrode Placement System [12]

1.2.2.3 Frequency

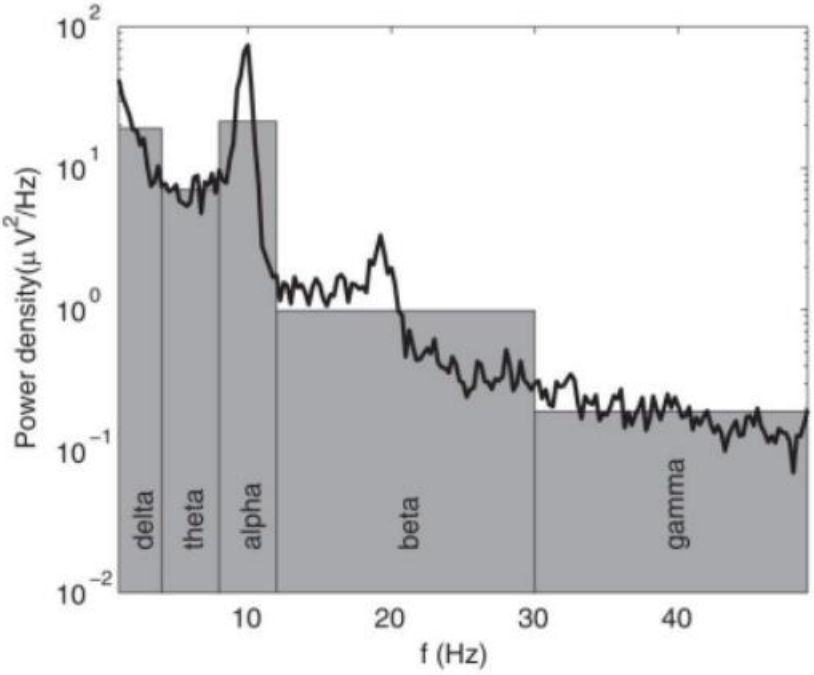


Figure 1-4 EEG frequency spectrum bands [13]

EEG signal frequency domain is divided into multiple frequency bands as shown in Figure 1-4. Measuring EEG at each of these bands would communicate the different type of information as illustrated in Figure 1-4. However, we can even extract the different type of information from EEG recording; transient activity. For instance, spikes in the data would be associated with a seizure.

EEG signal at a specific time could be either rhythmic, arrhythmic, or dysrhythmic signal.

- **Rhythmic signals** are signals of constant frequency.
- **Arrhythmic signals** are signals of variable random frequency.
- **Dysrhythmic signals** are signals with a relatively rare pattern of frequencies.

Table 1-1 EEG Frequency Bands

Frequency Band	Frequency Range (Hz)	Associated with
Delta	<4	• The third stage of the Non-Rapid Eye Movement, NREM, sleep. [14]
Theta	4-7	• The transition from sleep to waking up when measured from the cortical area. [13] • Passively awake subject when measured from the cortical area.[14]
Alpha	8-12	• Relaxation with closed eyes when measured from the occipital lobe. [15] • Brain activities such as memorization or calculation, while it is suppressed with visual activities. [15]
Beta	12-30	• Active thinking and concentration. • Need for muscles tonus, while it is reduced with contracting muscles. [15]
Gamma	30-100	• Communication of external stimulus with brain parts such as the hippocampus and parts of the cerebral cortex. [15] • High alertness and awareness of surroundings. [15]

1.2.2.4 Artifacts

As shown in the previous section, EEG signal features, it is clear that a lot of information could be extracted and the research interprets such signals with different biological behaviors. However, EEG recording is prone to noise. Noise caused by the patient known as “biological artifacts” and others by the environment called “environmental artifacts”.

1.2.2.4.1 Biological Artifacts

Biological Artifacts could be caused by cardiac and muscle movements. One method to reduce such artifacts is to instruct the patients to avoid moving their facial muscle since other muscular activities could be filtered out since they cause higher frequencies. Another method would be the use of Independent Component Analysis, ICA, that could filter out muscular activities, like blinking, from the measurements by applying the superposition principle.

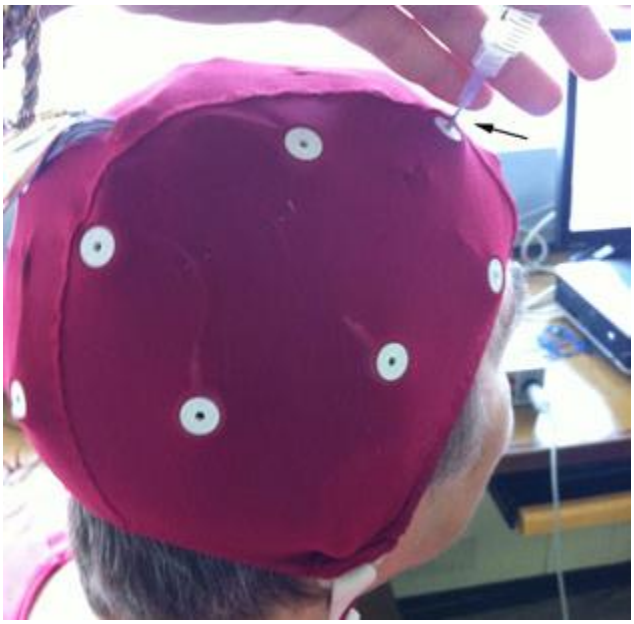


Figure 1-5 Electrode Gel injected in a cap into which EEG electrodes are embedded [12]

1.2.2.4.2 Environmental Artifacts

Environmental Artifacts is associated with the recording devices used; the electrodes and the connection with the scalp. Appropriate shielding of the electronics as well as using conducting gem to improve electrodes conduction could reduce such artifacts.

1.2.3 EEG periods

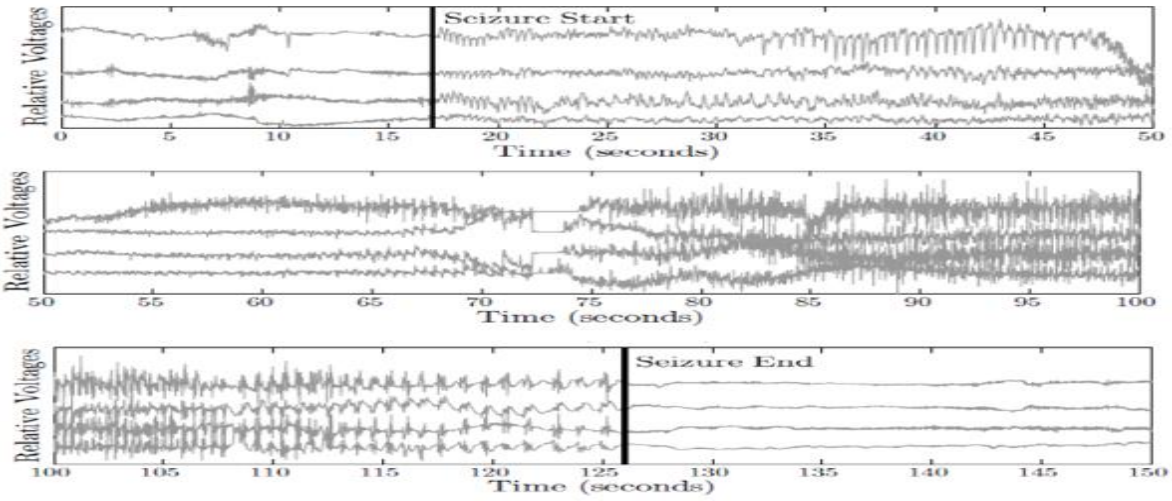


Figure 1-6 Pre-ictal, Seizure, and postictal periods respectively

1.2.3.1 Ictal Period

It ranges from 1 to 3+ minutes at which the seizure occurs.

1.2.3.2 Pre-ictal Period

It ranges from 30 to 60 minutes directly preceding the ictal period.

1.2.3.3 Postictal Period

It ranges from 30 to 60 minutes directly following the ictal period.

1.2.3.4 Inter-ictal Period

It's the normal period between postictal and pre-ictal periods.

2

SEIZURE PREDICTION

2.1 Seizure Detection versus Prediction

The seizure detection is considered a classification problem between two classes, where the first class is the ictal period when a true seizure happens and the second class is the non-ictal period including postictal, pre-ictal and inter-ictal periods. On the other hand, the seizure prediction is also a classification problem between two classes, but the first class is the pre-ictal period and the second class is the non pre-ictal period including inter-ictal, ictal and post-ictal periods as shown in Figure 2.1.

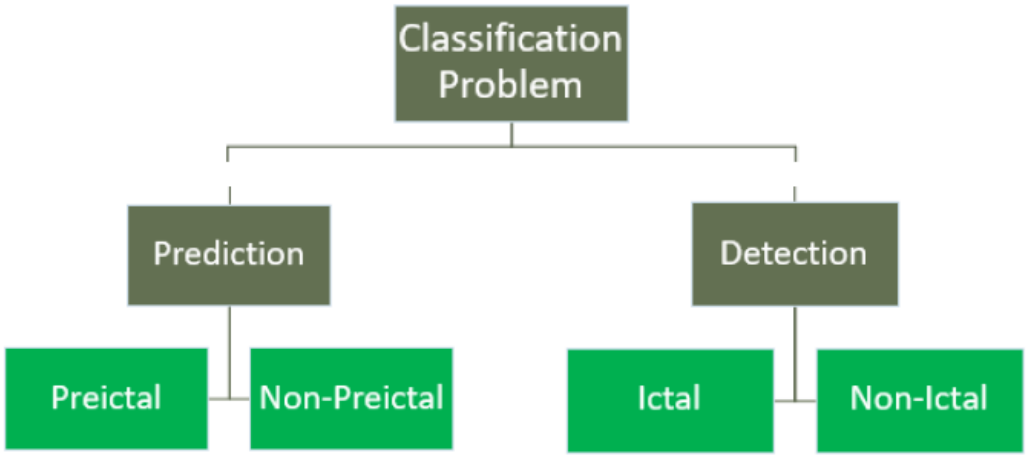


Figure 2-1 Seizure detection Vs. Seizure prediction

Table 2-1 Seizure Detection and Prediction

	Seizure Detection	**Seizure Prediction**
Classification between	Ictal and non-ictal Periods	Pre-ictal and non pre-ictal periods
Implementation Complexity	Relatively low. As shown in Figure 2-1, the ictal period has higher amplitudes and peaks making the classification easier.	Relatively high. Since pre-ictal and inter-ictal periods are not easily distinguishable.
Error-prone	Low	High

2.1.1 Why Prediction?

As shown in the table above, Table 2.1, seizure prediction is way more complex than detection having high error prone. Yet, we decided to focus on prediction in our project as we believe that prediction of seizures will cause a leap in the field. Most of the researches, papers and projects are concerned with detection and good results were achieved so it would be pointless doing the old work over again. Not only was the novelty of our project our main concern, but we also know for sure that prediction of seizures will protect many innocent souls from going through this harsh, painful and unforgettable experience as they will actually be able to stop it before it even begins.

2.2 Existing Solutions for treating seizures

2.2.1 Deep Brain Stimulation (DBS)

It's a technique where some electrodes are implanted in the patient's brain that can sense any abnormality in brain 's activity, accordingly a generator implanted in the chest will generate electrical pulses to suppress any detected seizures, this technique in treating epilepsy not considered an optimum solution since it depends on a surgery, a high cost battery and wires that can get defected under certain conditions.

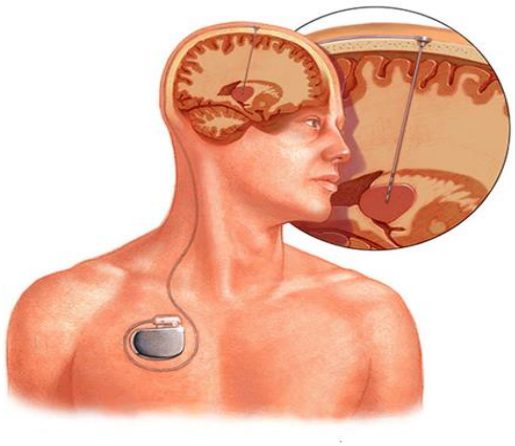


Figure 2-2 Deep brain stimulation's technique.[36]

2.2.2 Vagus Nerve Stimulation

A device like the pacemaker was developed to be inserted in the patient's chest under skin. This device is designed to deliver electrical impulses through the Vagus nerve running through your neck; accordingly, it prevents brain over excitation, thus, reducing the likelihood of seizures. This method of treatment is not fully developed yet as little information is known about how the vagal nerve can modulate mood and have control over seizures.

2.2.3 WINAM

A wearable headset to monitor the EEG signals through the scalp and gives alarm once seizure detected, it then accesses a secured cloud where the patient's doctor can login to it and prepare a diagnosed report regarding the patient's case.[33]

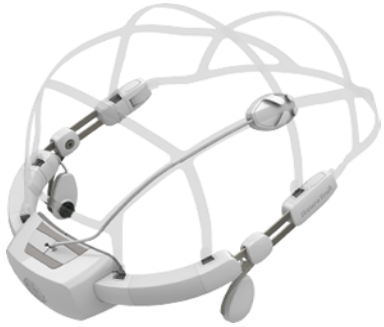


Figure 2-3 WINAM device.[38]

2.3 Project's Flow

2.3.1 Dataset Used

Data and algorithms are the most important basis of any machine learning project. Data must be analyzed and well organized to feed the AI algorithm for much better results. A simple algorithm with good data will outperform a complex one; on the other hand, a complex and powerful algorithm with bad data will cost time and consume power. So, our main project's aim and requirement was to collect data wisely.

For the data gathering, we read many papers published in well-known sites concerning epilepsy, seizure prediction and detection, then we filtered out a lot to only settle on the most trusted and reliable sources.

First, exploiting the fact that rats' brains are the closest to humans', we trained our model with some rats' data that we extracted from the Faculty of Science-Cairo University's labs to visualize the results and check for any needed modifications. Then, after understanding the bigger picture, it was about time we moved forward to data extracted from real patients suffering from epilepsy. Because everyone's health is our main concern, we tried to get the best data to fit our model with, as it is what basically our whole project depends on. There were many online datasets, some of which were trained on in trusted papers such as Freiburg's dataset, Bonn's dataset, MIT's dataset, etc. There were others too that we did not really pay attention to because they were recorded in unknown universities and were not really cited in any papers. And here came our first challenge, which was deciding which dataset we should settle on. Freiburg's dataset was eliminated because we found out it costs 3000 Euros and we were not capable of spending such a huge amount. Bonn's dataset was available for free so we tried testing it out, but eventually we had to omit it as well because it was not cited in many papers that we can fully consider trusted. In addition, the results were somehow misleading and a lot more different than what we expected and previously studied concerning epilepsy and the different seizure periods. Finally, we were left with MIT's dataset, that to our luck, our supervisor was capable of providing us with. MIT is without a doubt one of the greatest universities, so the fear of the source being untrusted was not an issue anymore. In addition, MIT data provided us with 23 different patients with more than one channel collecting data (EEG signals) from the patients' brains.

Although the data was ready for seizure detection, we had to do some analysis and modifications such as labeling the pre-seizure parts of the data in order to highlight it to the model for further training since our main project scope is seizure prediction.

Table 2-2 Dataset Prioritization

Dataset	Decision	Reason
ONE Lab's Rats'	Discarded	The files were a short interval for seizure prediction purposes.
Freiburg's	Discarded	Dataset costs 3000 Euros.
Bonn's	Discarded	The data was not reliable when tested. Very few papers used it in their bibliography.
CHB-MIT Scalp's	Chosen	It was free, documented, and cited in respectable journals. The data files averaged one hour of uninterrupted recording.

2.3.1.1 CHB-MIT Scalp EEG Database

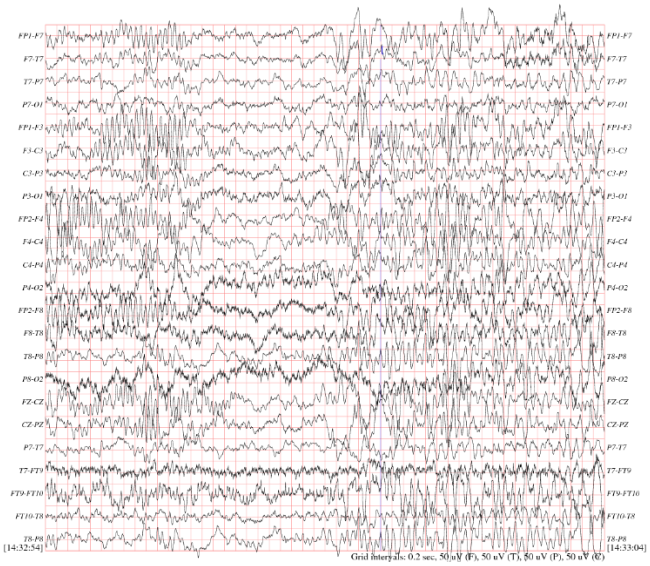


Figure 2-4 Captured EEG Signals [18]

Collected at the Children's Hospital Boston, the open source CHB-MIT Scalp EEG Database offered by PhysioNet was part of a Ph.D. by Ali Shoeb. [16] The database consists of EEG recordings from 22 pediatric subjects with intractable, hard-to-control seizures. Patients are divided into 5 males, aged between 3 and 22 years, and 17 females, aged between 1.5 and 19 years. The subjects were monitored for up to several days following withdrawal of anti-seizure medication in order to characterize their seizures and assess their candidacy for surgical intervention.

The database had between 9 and 42 continuous EDF files per subject. EDF files are considered a simple and flexible format for exchange and storage of multichannel biological and physical signals. In most cases, the EDF files contain 1 hour of digitized EEG signals retrieved from 23 channels since the 10/20 electrode placement system was the one that was used. All the EEG signals were sampled at 256 Hz with 16-bit resolution.

Beginning of ictal period is market in the data. However, pre-ictal periods are not marked. We took a 30-minute interval preceding the marked beginning of the ictal period

as the pre-ictal period. There are some missing recorded hours in some patients but this didn't cause problems because these non-recorded hours were in the non-seizure period and there were no upcoming seizures but in patients 18, 19, 21 & 22 these missing hours would cause problems due to the fact that they are just prior to a seizure (in the pre-ictal period). It should be noted that we chose to work with the EEG data of several -and not all- patients only due to the limitations of the recorded data; patients 7, 8, 11, 19, 20 & 23.

2.3.2 Features Extraction

Another requirement was for us to choose a set of features that when extracted from the dataset would be iconic in our prediction problem. Unfortunately, one would not be able to clarify the difference between the pre-seizure period and the normal period with the naked eye, they are almost identical. The only observable difference is the one between the actual seizure period and other periods, including the pre-seizure and the normal which would have been useful in the detection problem but not our case. Thus, the EEG signal was not the easiest to work with neither was choosing the right set of features.

After many trials with many different combinations, we settled on 2 different sets of features, which when each used alone, yielded very promising results. So, this had arisen two models that our software team members developed and tested. First model required less computation time and power, and the second model consumed higher power but gave better results. So the next challenge was for us to decide whether we should go with the higher results or the lower power consumption. Obviously, it is a tradeoff, but in our case, since the code is to be transferred to an implantable chip inside the brain, thus lower power consumption had a higher priority and was our main concern despite the fact that the second model is power hungry.

2.3.3 Machine Learning algorithms for training and classification.

This phase is considered the software phase in our project's flow at which the EEG data signals are measured from the patient's brain for producing a suitable training data, this training data is produced under certain features and machine learning algorithms for classifying these data into one of the 2 classes using a specified kernel, so that it can be used as a reference for the testing data generated later for detecting/predicting epileptic seizures.

2.3.4 Low-Level VHDL/Verilog

This phase is considered very important for transferring the software simulations into hardware descriptive language so that it can process with Real-time measured EEG data signals, this EEG signals are considered the testing data that shall be processes under the same features of the training phase for comparing and classifying the results to the rightful class so that it can detect/predict a seizure while calculating performance results; sensitivity, specificity and accuracy.

2.3.5 FPGA for hardware testing.

The FPGA phase is considered the hardware testing phase, at which the FPGA chip is implanted inside the patient's brain for processing the EEG signals that are measured in a synthesizable way.

3 MACHINE LEARNING

When it comes to learning, analyzing and classifying without the intervention of humans, there is no better solution than machine learning. Machine learning is a method of data analysis and statistical observation that is made by computer. Machine learning is an Artificial intelligence (AI) application that is used to perform a specific task without being explicitly programmed. It takes some input data with labeling information and tries to configure some sort of patterns that can be used to identify objects or classify each input entry to its specific class automatically.

Since our project flow is dealing with hundreds EEG signals that are not easy to observe and our main goal is to predict epileptic seizure (differentiate between pre-ictal and non-pre-ictal), we decided to use Machine learning technique fed with EEG signals and their labels as input data to get the required results. There are many major types and variety of machine learning techniques from which we had to choose the most suitable one.

3.1 Machine learning Types

3.1.1 Supervised Learning

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into regression and classification problems.

- **Classification:** A classification problem is when the output variable is a category, such as "red" or "blue" or "disease" and "no disease".
- **Regression:** A regression problem is when the output variable is a real value, such as "dollars" or "weight".

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.so that The majority of practical machine learning uses supervised learning.

3.1.2 Unsupervised Learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal of unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there are no correct answers and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

- **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

3.1.3 Semi-Supervised Leaning

Semi-Supervised Learning is where you have a large amount of input data (X) and only some of the data is labeled (Y) are called semi-supervised learning problems. These problems sit in between both supervised and unsupervised learning.

A good example is a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.

Many real-world machine learning problems fall into this area. This is because it can be expensive or time-consuming to label data as it may require access to domain experts. Whereas unlabeled data is cheap and easy to collect and store.

You can use supervised learning techniques to make best guess predictions for the unlabeled data, feed that data back into the supervised learning algorithm as training data and use the model to make predictions on new unseen data.

3.1.4 Type Choice

The data used in our project is labeled ‘1’ for pre-ictal and ‘0’ in non-pre-ictal stages. After viewing the different types of machine learning, Supervised Learning (especially Classification) is what we need to accomplish our goal and classify our data to eliminate this disorder for good.

3.2 Machine Learning Techniques

Classification Machine learning problem is mainly based on a proper separator between these multi-classed data. There are many techniques used for classification machine learning problem to help configure two or more domains for these classes. The techniques proposed are support vector machine SVM), Artificial Neural Network (ANN) and gradient Ascent (GA)

3.2.1 Support vector machine (SVM)

Support Vector Machine” (SVM) is a supervised machine learning algorithm that is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features extracted) with the value of each feature being the value of a particular coordinate. [17] Then, we perform classification by finding the optimal hyperplane that differentiate the two classes very

well with the help of kernel function (that will be discussed in the following sections).[17]

The figure below shows how SVM generates separator hyperplanes for the given classes:

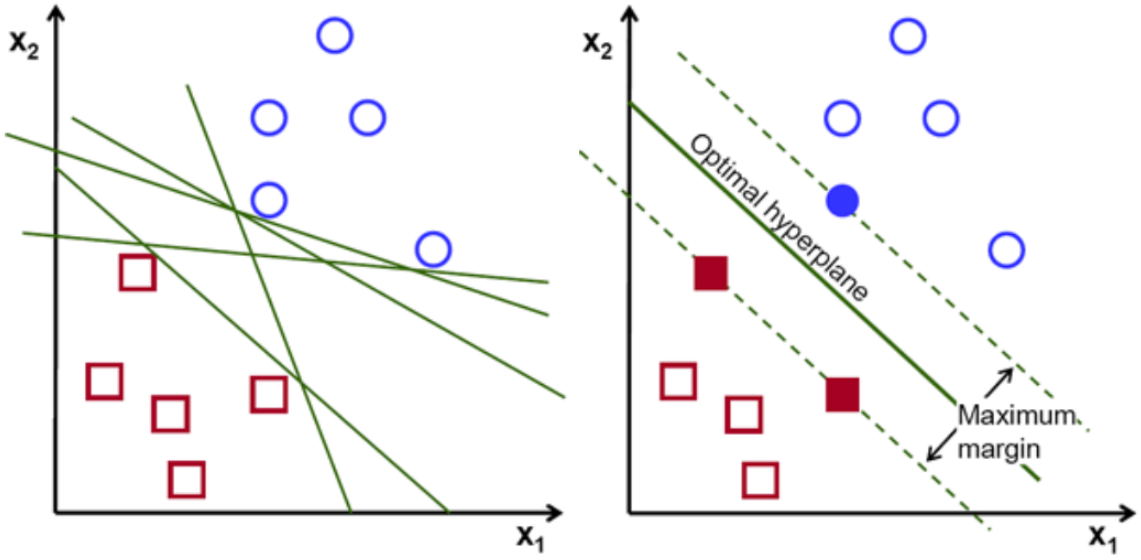


Figure 3-1 SVM's hyperplane generation.[18]

Support Vectors are simply the coordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes.

3.2.2 Stochastic Gradient Descent

Stochastic gradient descent is a simple and very efficient approach to fit linear models. It is particularly useful when the number of samples is very large. It supports different loss functions and penalties for classification.[19]

The procedure starts off with initial values for the coefficient or coefficients for the function. These could be 0.0 or a small random value, and then the cost of the coefficients is evaluated by plugging them into the function and calculating the cost.

The derivative of the cost is calculated. The derivative is a concept from calculus and refers to the slope of the function at a given point. We need to know the slope so that we

know the direction (sign) to move the coefficient values in order to get a lower cost on the next iteration.

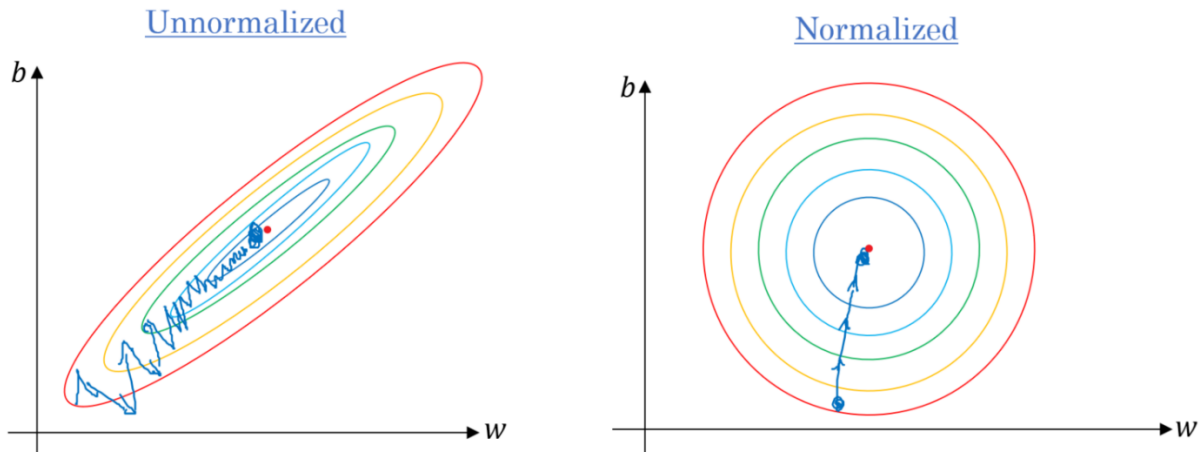


Figure 3-2 Normalized vs Unnormalized Gradient Descent

Now that we know from the derivative which direction is downhill, we can now update the coefficient values. A learning rate parameter (alpha) must be specified that controls how much the coefficients can change on each update. This process is repeated until the cost of the coefficients (cost) is 0.0 or close enough to zero to be good enough.

3.2.3 Artificial Neural Network

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the nervous system inside the human's brain. The key element of this paradigm is the structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in harmony to solve specific problems. ANNs, like people, learn by example. An ANN can be configured to solve the classification problem as we face in our project, through a learning process. [21]

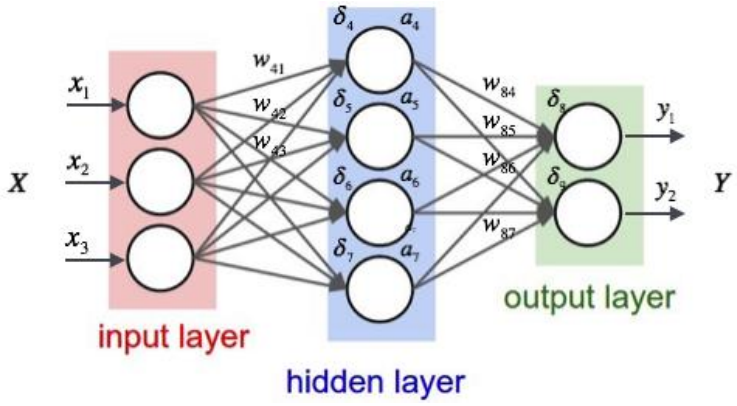


Figure 3-3 Artificial Neural Network.[21]

This learning process is iterative in which inputs' weights are adjusted alongside bias points to provide us with the correct class.

3.2.4 Technique Choice

There are some factors on which we focused before choosing the suitable technique, some of these factors are:

- Accuracy and efficiency of results
- Power and Time consumption
- Ease of hardware implementation
- Minimum Hardware area and Maximum throughput Requirement

Neural Network is one of the most powerful tools for classification problem but it violates one of our focal factors as its Hardware dependent. it requires processors with parallel processing power, and this requires a large area with a lot of difficulty in designing and connecting modules together in hardware design.

As to Stochastic Gradient Descent, it violates not just one factor but many. First, If the learning rate for gradient descent is too fast, you are going to skip the true local minimum which means accuracy will drop greatly and affects the whole system, and If it is too slow, the gradient descent may never converge or it will converge after very relative long

time and this will, unfortunately, consume a lot of time and power. It is also very sensitive to feature scaling which will provide us with inaccurate results.

But in the case of SVM, its models have generalization in practice, the risk of over-fitting is less in SVM. Which means the accurate result will be provided. SVM is always compared with ANN. When compared to ANN models, SVMs give better results. It also needs low power and time consumption compared to SGD, and of course, less hardware complexity compared to both SGD and ANN. One major disadvantage of SVM that Choosing a "good" kernel function is not easy, as there are many types of kernel functions used depending on features extracted and type of SVM classifier used. We managed not only overcome this disadvantage but also get the benefit of it by using two types, one with specific extracted feature and the other with the rest of features and then we compared to power and time consumption to select the best combination. (This will be explained in details in the following chapters).

3.3 Support Vector Machine (SVM)

Support vector machine is one of the greatest machine learning techniques widely used in binary classification problems like ours. It is trained with a series of data already classified into two categories to build a model that will map any input data to its corresponding class. SVM can provide a unique solution and is a strongly regularized which means that it will appropriately fit the model to the data so overfitting problem will not occur.

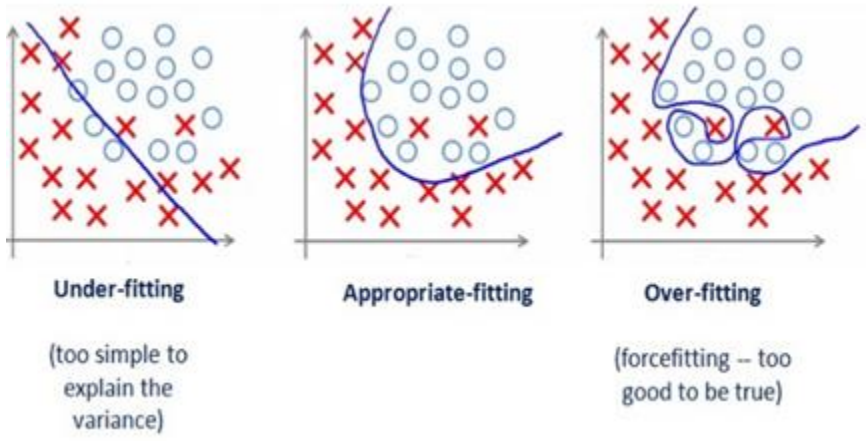


Figure 3-4 Underfitting, and Overfitting binary classification

A support vector machine is a classification method. It works on the principle of fitting a boundary to a region of points Once a boundary is fitted on the training data, for any new test points that need to be classified, we must simply check whether they lie inside the boundary or not. The advantage of SVM is that once a boundary is established, most of the training data is redundant. All it needs is a core set of points which can help identify and set the boundary. These data points are called support vectors because they "support" the boundary. And they are called vectors because they set data that contain values describing the extracted features.

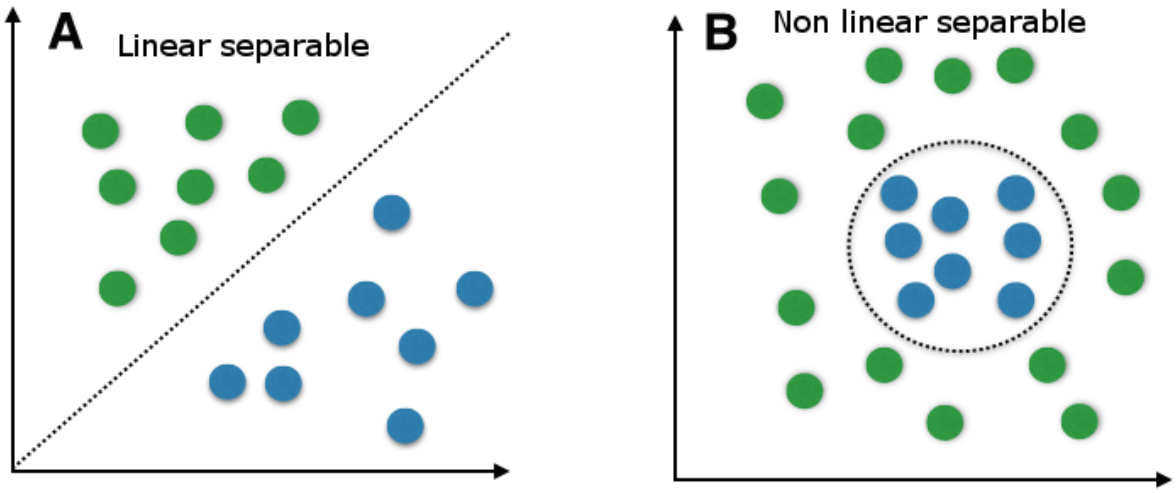


Figure 3-5 Example of linear and nonlinear separation boundaries

This boundary is traditionally called a hyperplane (the dotted line in the figure). If we are dealing with only two features extracted then this boundary can be a straight line or a curve (as shown above). In three dimensions it can be a plane or a complex surface. Higher dimensions are obviously impossible to visualize, so we don't call this separator a line but a hyperplane which is a generic name for the boundary in more than 3 dimensions.

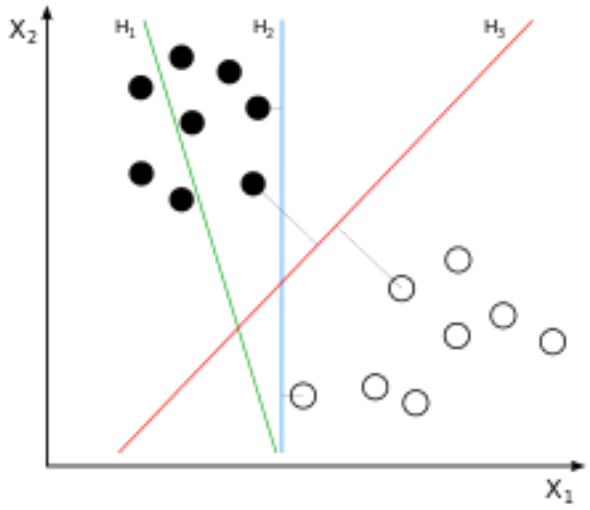


Figure 3-6 Possible separating hyperplanes

As seen in the above image, a number of such hyperplanes can be found. This problem doesn't have a unique solution but an infinite number of hyperplanes can separate between these two classes. The goal is to find the best line that separates the data with maximum distance away from the nearest point of both classes.

3.4 How does it work? (SVM Methodology)

Our classification problem is a binary one, in which there are only two classes the positive one ('1') representing pre-ictal stages and the negative ('0') representing non-pre-ictal. If these data entries are linearly separable and there is an optimal hyperplane that clearly separates without having any point of one class inside the boundary of the other then SVM version is called hard margin. In case of error points, we need to use a more general version of SVM that is with soft margin. First, we will start the flow with a hard margin and then try to extend it to overcome any error in data by using soft margin features.

3.4.1 Hard margin

The hard margin version of SVM separate the feature extracted data linearly (as shown in the figure below) by creating two parallel hyperplanes that separate the two classes (pre-ictal and non-pre-ictal) so that it maximizes the distance between the optimal hyperplane (in red color) and the nearest data point of both classes which is called the margin.[22]

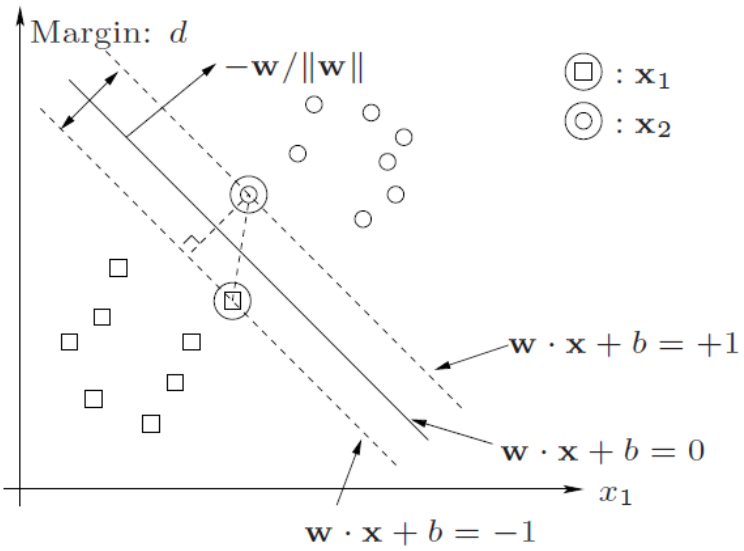


Figure 3-7 Hard margin hyperplane

These hyperplanes can be described by the equations:

$$\vec{w} \cdot \vec{x} + b = 1 \tag{3.4.1}$$

$$\vec{w} \cdot \vec{x} + b = -1 \tag{3.4.2}$$

Anything on or above the boundary of equation (3.4.1) is considered positive sample and satisfy the following equation:

$$\vec{w} \cdot \vec{x} + b \geq 1 \tag{3.4.3}$$

And anything on or below the boundary of equation (3.4.2) is considered a negative sample and satisfy the following equation.

$$\vec{w} \cdot \vec{x} + b \leq -1 \tag{3.4.4}$$

Then we can write a general equation for the output y:

$$y_i * (\vec{w} \cdot \vec{x_i} + b) \geq 1 \tag{3.4.5}$$

If we multiply the output of input point whether it's located in a positive or negative region, the result will always be more than or equal to 1 because both will have the same sign.

Our main goal is to maximize the margin, so now we will calculate the distance and then pass it to Lagrange function for optimization.

Let x_o be a point in the hyperplane $\vec{w} \cdot \vec{x} + b = -1$ to measure the distance between hyperplanes $\vec{w} \cdot \vec{x} + b = -1$ and $\vec{w} \cdot \vec{x} + b = 1$, we only need to compute the perpendicular distance from x_o to plane $\vec{w} \cdot \vec{x} + b = 1$, denoted as r.

Note that $\frac{w}{\|w\|}$ is a unit normal vector of the hyperplane $\vec{w} \cdot \vec{x} + b = 1$. Then we have:

$$w\left(x_o + r\frac{w}{\|w\|}\right) + b = 1 \tag{3.4.6}$$

Since $x_o + r\frac{w}{\|w\|}$ should be a point in hyperplane $\vec{w} \cdot \vec{x} - b = 1$ according to our definition of r. and by Expanding this equation, we have:

$$wx_o + r\frac{\|w\|^2}{\|w\|} + b = 1 \tag{3.4.7}$$

$$wx_o + r\|w\| + b = 1 \tag{3.4.8}$$

$$wx_o + b = 1 - r\|w\| \tag{3.4.9}$$

$$-1 = 1 - r\|w\| \tag{3.4.10}$$

$$r = \frac{2}{\|w\|} \tag{3.4.11}$$

Then our optimization problem is to maximize $\frac{2}{\|w\|}$ or equivalently minimize $\frac{1}{2}\|w\|$

Subject to (3.4.3), and as stated before this can be reduced to Lagrange function defined as follows:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N}\alpha_i(y_i * (\overrightarrow{x_i}.\vec{w}+b) - 1) \quad (3.4.12)$$

By setting $\frac{\partial L}{\partial b} = 0$ and $\frac{\partial L}{\partial w} = 0$ subject to the constraint $\alpha_i \geq 0$ results in

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \text{ and } w = \sum_{i=1}^{N}\alpha_i y_i x_i = 0 \quad (3.4.13)$$

Substituting these results back into the Lagrange function:

$$L(w,b,\alpha) = \frac{1}{2}(w.w) - \sum_{i=1}^{N}\alpha_i y_i(\overrightarrow{x_i}.\vec{w}) - \sum_{i=1}^{N}\alpha_i y_i b + \sum_{i=1}^{N}\alpha_i \quad (3.4.14)$$

$$= \frac{1}{2}\left(\sum_{i=1}^{N}\alpha_i y_i \overrightarrow{x_i}.\sum_{j=1}^{N}\alpha_j y_j \overrightarrow{x_j}\right) - \sum_{i=1}^{N}\alpha_i y_i \overrightarrow{x_i}.\sum_{j=1}^{N}\alpha_j y_j \overrightarrow{x_j} + \sum_{i=1}^{N}\alpha_i$$

$$= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\overrightarrow{x_i}.\overrightarrow{x_j}\right)$$

The result in the following Wolfe dual formulation:

$$\max_{\alpha}\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\overrightarrow{x_i}.\overrightarrow{x_j}\right) \quad (3.4.15)$$

Subject to

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \ and\ \alpha_i \geq 0,, i = 1,\dots N \quad (3.4.16)$$

We managed to minimize our optimization problem from minimizing the margin to minimizing the Lagrange coefficient α , and once α is optimized calculated then we will substitute to get w and b as $w = \sum_{i=1}^{N}\alpha_i y_i x_i = 0$ and b can be computed using KKT condition where:

$$\alpha_i(y_i * (\overrightarrow{x_i}.\vec{w}+b) - 1) = 0 \quad (3.4.17)$$

Then

$$b = 1 - y_i(\vec{x_i} . \vec{w}) \quad (3.4.18)$$

3.4.2 Soft Margin

The figure below is actual footage of two features extracted from EEG data that is used in our project. The figure clearly shows that data points are not linearly separable but there are error points that encouraged us to extend the hard margin version to a soft one.

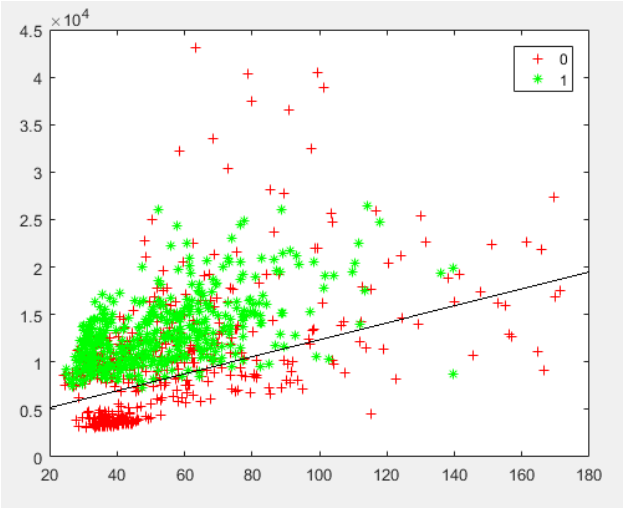


Figure 3-8 Two actual features of EEG data used

In both the soft margin and hard margin case, we are maximizing the margin between support vectors. In the soft margin case, we let our model give some relaxation to a few points, if we consider these points our margin might reduce significantly and our decision boundary will be poorer, so instead of considering them as support vectors we consider them as error points and give certain penalty for them which is proportional to the amount by which each data point is violating the hard constraint.

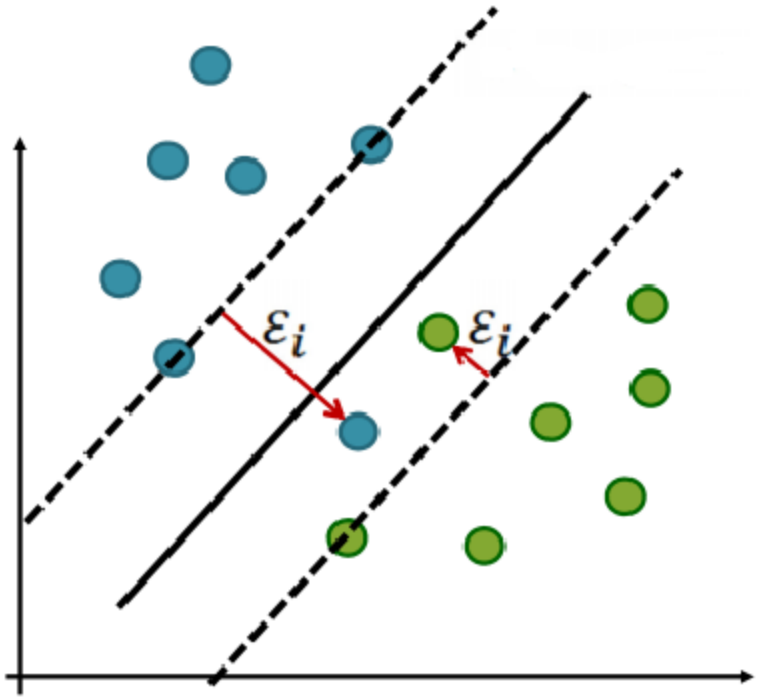


Figure 3-9 Decision boundary

In the hard margin, the minimization problem in equation (3.4.12) is discussed without any error points; in case they exist we need to use a more general version of SVM with slightly different minimization problem constraint, as in soft margin the data cannot be fully separated by a linear hyperplane. A set of slack variables $\xi_i \dots \xi_n$ is introduced with $\xi_i \geq 0$ such that the inequality constraints in SVM become:

$$y_i * (\vec{w} \cdot \vec{x_i} + b) \geq 1 - \xi_i \qquad (3.4.19)$$

Now, ξ helps us in allowing some slackness in constraint. This approach gives a linear penalty to mistakes in classification.

The minimization problem becomes

$$min\frac{1}{2}\|w\|^2 - C\sum_{i=1}^{N}\xi_i \quad (3.4.20)$$

Subject to equation (3.4.19), Where C is a user-defined penalty parameter to penalize any violation of the safety margin for all training data.

The new Lagrange function will be:

$$L(w,b,\alpha,\xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i * (\overrightarrow{x_i}.\vec{w} + b) - 1 + \xi_i) - \sum_{i=1}^{N}\beta_i\xi_i \quad (3.4.21)$$

Where $\alpha_i \geq 0\ and\ \beta_i \geq 0$ are Lagrange multipliers to ensure that $\xi_i \geq 0$ and

$y_i * (\vec{w} \cdot \overrightarrow{x_i} + b) \geq 1 - \xi_i$

By differentiating $L(w,b,\alpha,\xi)$ w.r.t w, b, and ξ_i , we obtain theses equations:

$$\frac{dL}{dw} = w - \sum_{i=1}^{N}\alpha_i y_i x_i = 0 \quad (3.4.22)$$

$$\frac{dL}{db} = \sum_{i=1}^{N}\alpha_i y_i = 0 \quad (3.4.23)$$

$$\frac{dL}{d\xi} = C - \alpha_i - \beta_i = 0 \quad (3.4.24)$$

By substituting in Lagrange function, we will get the same Wolfe dual equation

$$\min_{\alpha}\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\overrightarrow{x_i}.\overrightarrow{x_j}\right) \quad (3.4.25)$$

Subject to

$$0 \leq \alpha_i \leq C,\ i=1\ldots N,\ \sum_{i=1}^{N}\alpha_i y_i x_i = 0$$

To get more insight about C, and ξ the following figure explain the three types of support vectors

1. <u>On the margin</u>

$C > \alpha_i > 0, \xi_i = 0$

$y_i * (\vec{w} \cdot \vec{x_i} + b) = 1$

For point '1'

$\alpha_1 = 2.85 , \xi_1 = 0$

2. <u>Inside the margin</u>

$\alpha_i = C, 0 < \xi_i < 2$

$y_i * (\vec{w} \cdot \vec{x_i} + b) \leq 1$

For point '10'

$\alpha_{10} = 10 , \xi_{10} = 0.667$

3. <u>Outside the margin</u>

$\alpha_i = C, \xi_i \geq 2$

$y_i * (\vec{w} \cdot \vec{x_i} + b) \leq 1$

For point '20'

$\alpha_{20} = 10, \xi_{20} = 2.667$

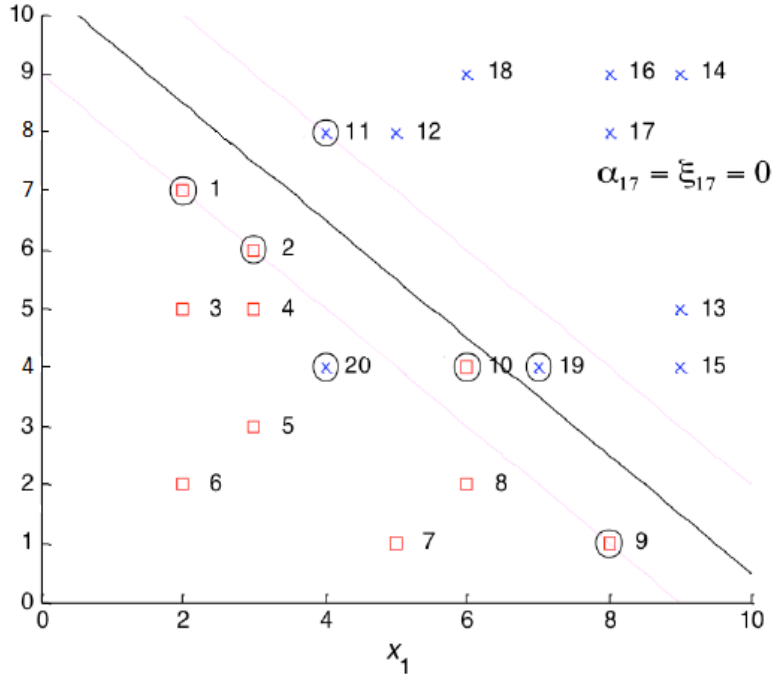


Figure 3-10 Decision margin estimation

3.4.3 Kernel

There are two main problems when visualizing multi-class data.

1. The points are overlapping on each other making it impossible for linear hyperplane to separate between the two classes (shown on the left figure below).

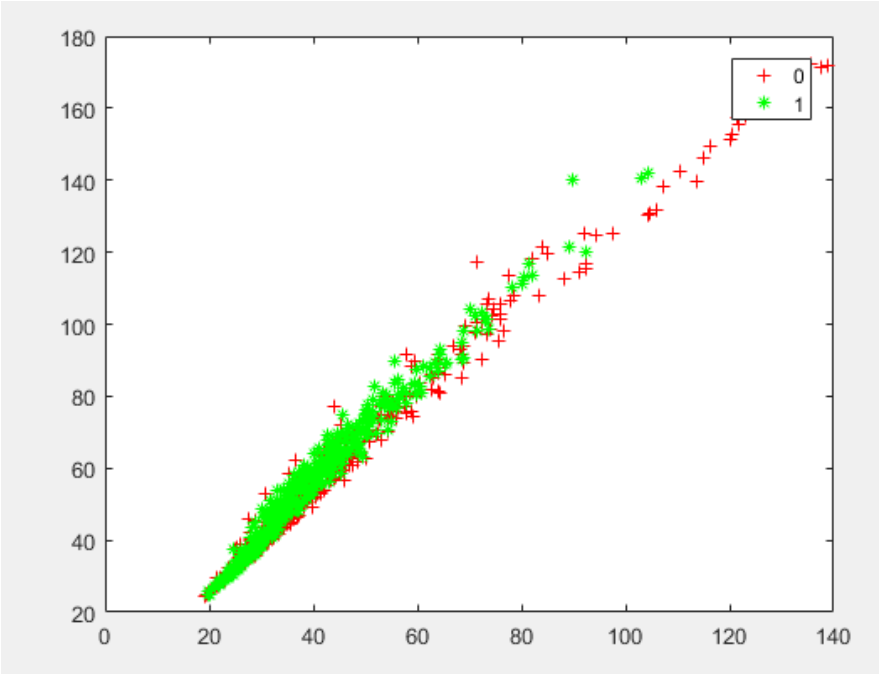


Figure 3-11 Overlapping example of binary classes

2. Points of the first class are surrounded by the data points of the other class, and in this case, a nonlinear hyperplane is needed to fully separate between the two classes (shown on the figure below).

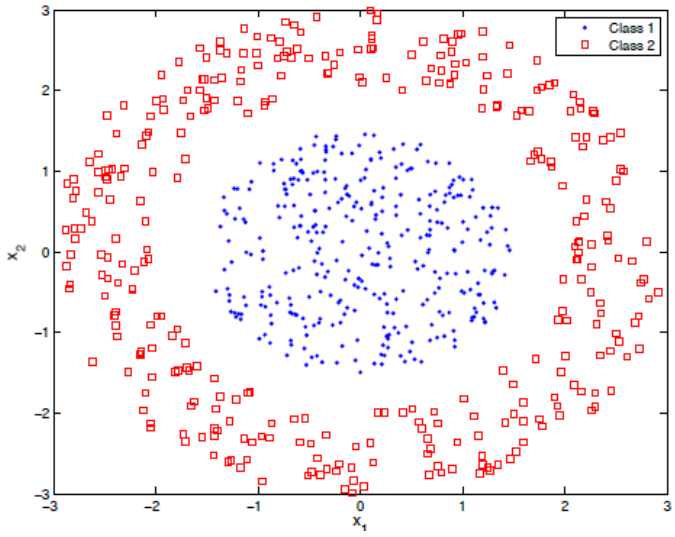


Figure 3-12 Binary classes that need nonlinear separation hyperplane

If we have a 1-D problem, and support vectors are categorized to two classes then we will get two decision boundaries and we can solve this problem by converting the dimension to a higher one just by adding a nonlinear function.

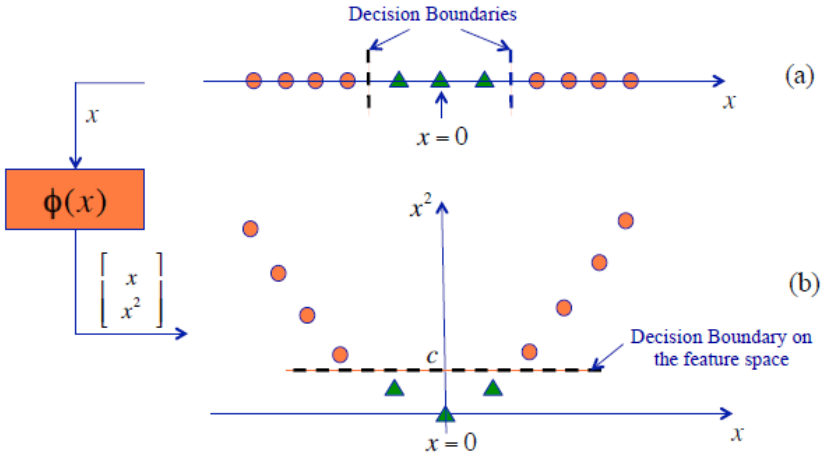


Figure 3-13 Adding a higher dimension to allow for the decision boundary

If our visualization is in the n-Dimensional graph (i.e. n =2) and the points need a nonlinear hyperplane to separate these two classes, then also by adding another nonlinear function will increase these the dimension and enable linear SVM to separate classes as shown in the figure.

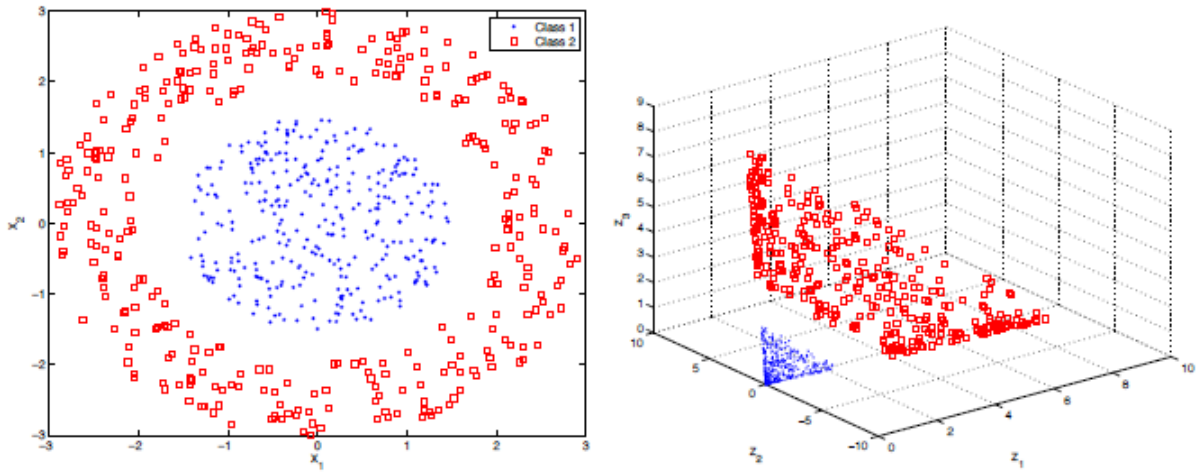


Figure 3-14 Making the data linearly separable by adding a higher dimension

The linear SVM has the form

$$f(x_t) = \sum_{i\epsilon S}^{N} \alpha_i y_i \varphi(x_i)^T \varphi(x_t) + b \qquad (3.4.26)$$

Where S is the set of support vector indices and $\varphi(x_i)$is the value of x_i after transformation into the higher dimension space.

The figures above are just for illustration, as the dimension of φ(x) is very high and may tend to infinite with some features in our case, meaning that this function may not be implementable. Fortunately, the dot product of $\varphi(x_i).\varphi(x_t)$ can be replaced by a kernel function.

3.4.3.1 Types of Kernels

- **Linear Kernel:**

$$k(x_i, x_j) = x_i . x_j$$

- **Polynomial Kernel:**

$$k(x_i, x_j) = (x_i . x_j + y)^m$$

- **Laplacian Kernel:**

$$k(x_i, x_j) = e^{-\psi \|x_i - x_j\|}$$

- **Radial Basis Kernel (RBF) (Gaussian):**

$$k(x_i, x_j) = e^{\frac{\|x_i . - \|^2}{2\sigma}}$$

3.4.4 Sequential Minimal Optimization (SMO)

When using any of the kernel tricks the minimization problem becomes as the following:

$$\min_\alpha \sum_{i=1}^{N} \alpha_i - 0.5 \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \mathrm{k}(x_i, x_j) \alpha_i \alpha_j \quad (3.4.27)$$

However, since the equation in (3.4.27) which is required for finding α_{min} is a Quadratic Programming, QP, problem which:

1. Requires a lot of processing time. For QP problems to be solvable, identifying the upper and lower constraints is crucial, but with a large number, variables result in a more time-consuming calculation. This doesn't allow for real-time optimization.
2. Necessitates large memory, and thus more power consumption. The quadratic form requires a matrix of length equals to the square of the number of the training examples.

Sequential Minimal Optimization, SMO, algorithm could be used to break this Quadratic Programming, QP, problem to easier and less complex QP problems. It was proposed in 1998 as an algorithm specifically for training SVM. [23]

SMO solves for the smallest QP problem that involves only two Lagrange multipliers under linear equality constraint. SVM is updated at each step that SMO solves the small QP problem. Solving this small QP problem doesn't need any complex numerical computation

3.4.4.1 Conditions on Alpha

We try to get minimum alpha that satisfies equation (3.4.27). However, the alpha chosen must obey Karush-Kuhn-Tucker (KKT) conditions, so that the solution is optimal and positive definite. [24]

$$\begin{aligned} \alpha_i = 0 &\leftrightarrow y_i u_i \geq 1 \\ 0 < \alpha_i < C &\leftrightarrow y_i u_i = 1 \\ \alpha_i = C &\leftrightarrow y_i u_i \leq 1 \end{aligned} \qquad (3.4.28)$$

3.4.4.2 Calculating the two Lagrange Multipliers

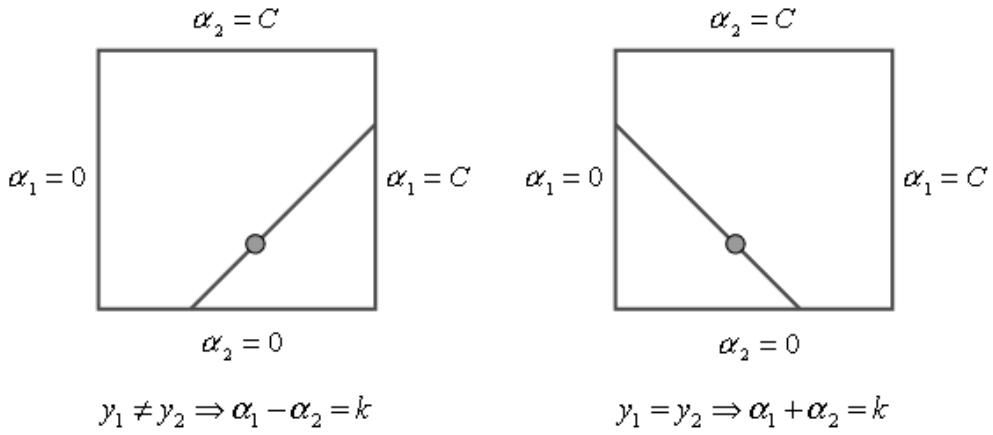


Figure 3-15 Two Lagrange multipliers obeying linear inequality constraints. [23]

Since the multipliers for the simple QP are only two, the constraints can be viewed in a 2D plane as shown in Figure 3.15. The constraint C shown in KKT equations makes the plane appear as a box. The linear constraint of the multipliers appears as a diagonal line is as follows.

$$\begin{aligned} &when\ y_1 \neq y_2,\ L = \max(0,\ \alpha_2 - \alpha_1),\ H = \min(C, C + \alpha_2 - \alpha_1) \\ &when\ y_1 = y_2,\ L = \max(0,\ \alpha_2 + \alpha_1 - C),\ H = \min(C, \alpha_2 + \alpha_1) \end{aligned} \qquad (3.4.29)$$

The second derivative of the objective function along the diagonal is

$$\eta = \mathrm{K}(x_1, x_1) + \mathrm{K}(x_2, x_2) - 2\mathrm{K}(x_1, x_2) \quad (3.4.30)$$

In case the kernel function obeys Mercer's conditions, η will be positive definite. We can calculate minimum alpha along with the diagonal linear equality constraint as follows

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta} \quad (3.4.31)$$

The constrained minimum after that will be calculated by clipping the unconstraint minimum to the ends of the line segment as follows:

$$\alpha_2^{new,clipped} = \begin{cases} H, \ if \ \alpha_2^{new} \geq H \\ \alpha_2^{new,}, \ if \ L \leq \alpha_2^{new} \leq H \\ L, if \ \alpha_2^{new} \leq L \end{cases} \quad (3.4.32)$$

Now the new α_1^{new} is calculated from $\alpha_2^{new,clipped}$.

$$\alpha_1^{new} = \alpha_1 + \ y_1 \, y_2 \, (\alpha_2 - \alpha_2^{new,clipped}) \quad (3.4.33)$$

η will be negative, in case the kernel function doesn't obey Mercer's conditions, [24] Moreover, η may be equal to zero in case the same vector is the same for more than one training sample. For whatever the value of η, the SMO algorithm computed the objective function where the weighted vector, $W = \sum_{j=1}^{N} y_i \alpha_i x_i$, where the threshold is $b = w.x_k - y_k$ for some $\alpha_k > 0$.

3.4.5 SMO vs Gilbert's Algorithm

Much like the SMO, Gilbert's Algorithm aims to solve the SVM QP problem by decomposition of the large QP problem to multiple simpler ones. But Gilbert algorithm is not directly suitable for the soft margin. However, the major drawbacks of Gilbert algorithm are that in many cases it becomes very slow, and it's a power-hungry algorithm.

4

SOFTWARE PHASE

The software phase, in which we implemented our code using Matlab, was one of the major phases in our project. This phase required lots of testing and trials to end up with the optimum and perfect results. That's why we used the software tool Matlab to write and validate our code. Also, to determine the flow. Which means that, we had to yield out a fast, accurate and synthesizable code.

Our software phase passed through the main steps of any software cycle; Initiation, Design, Coding, Testing & Maintenance.

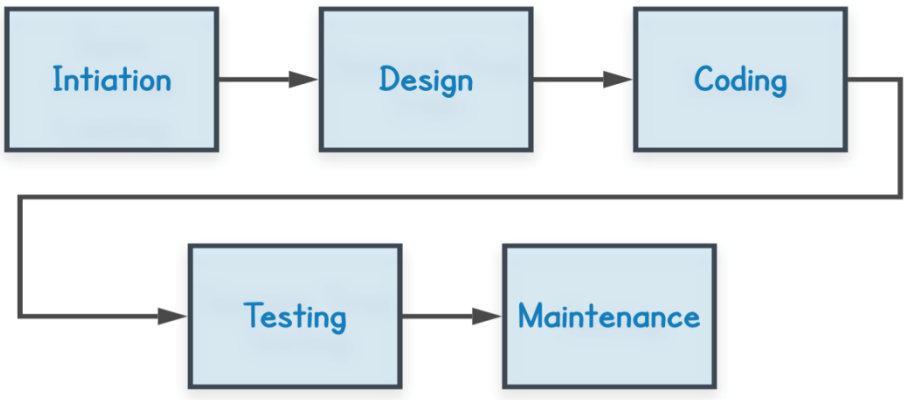


Figure 4-1 The Software Cycle

4.1 Matlab

The reason why we chose Matlab over other alternatives is that it has vectorized operations, optimized graphical output and its basic data element is the matrix. Which of course made dealing with the data saved in arrays form much easier.

4.1.1 Matlab Work Flow

The Matlab code stage is a very essential as it helps us to monitor and see actual results without hardware implementation which makes our work precise, accurate and smart. Our Matlab code blocks flow are as follows:

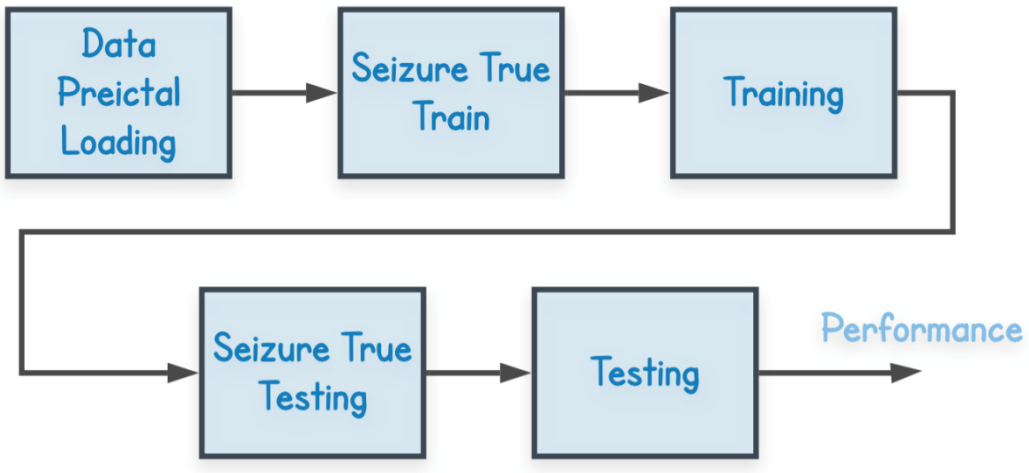


Figure 4-2 The Matlab Flow

As shown in the previous figure, our software flow consists mainly of these 5 blocks. Data Preictal Loading is our first block to start with. This block is mainly a function that is used to determine the start of the pre-ictal period, which is the period prior to the seizure having different features and properties than the normal one. In our algorithm, it is necessary to get the start of the pre-ictal period in order to be able to predict the upcoming seizures and to capture it's characteristics. After that, Seizure True Train block duty starts. This block is about an array that describes the input training data, which is a recording of a patient's brain signal along a period of time. Accordingly, the aim of this array to point out numerically when a seizure comes. That is exactly why this block can be said to be our reference block. Training of any system is always said to be the most

critical part of any project. If it is done carefully and accurately, miraculous outcome could be yielded. Else, a massive amount of chaos will be witnessed. Hence, we had to do many try and error tests in order to know how to perfectly train our system on a wide variety of datasets. For each patient there are many brain signal recordings which are also known as EEG signals. Among those EEG signals recordings, we use only one of them for training and the rest are used for testing. In this block, training is done using our chosen features, which will be discussed later in this chapter. As a consequence, a feature captured data is yielded and then used by the SVM to get our trained data. The Seizure True Test block's function is too close to the Seizure True Train block. The only difference is that it is concerned with the testing data unlike the Seizure True Train which is concerned with the training data. Finally, in our last block Testing, testing is done plenty of times on different recordings for the same patient in order to ensure that our system is adaptive and the features used obtained accurate system with high specificity, accuracy and sensitivity.

4.1.2 Matlab Performance

Our system performance is measured through our main 3 aspects **sensitivity**, **specificity** and **accuracy**.

I. <u>Sensitivity:</u>

Sensitivity is the ability of a test to correctly identify those with the disease which is also known as true positive rate.

$$Sensitivity = \frac{TP}{TP + FN}\ x\ 100$$

II. <u>Specificity:</u>

Specificity is the ability of the test to correctly identify those without the disease which is also known as true negative rate.

$$Specificity = \frac{TN}{TN + FP}\ x\ 100$$

III. Accuracy:

Accuracy refers to how close a sample statistic is to a population parameter.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \; x\,100$$

Where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

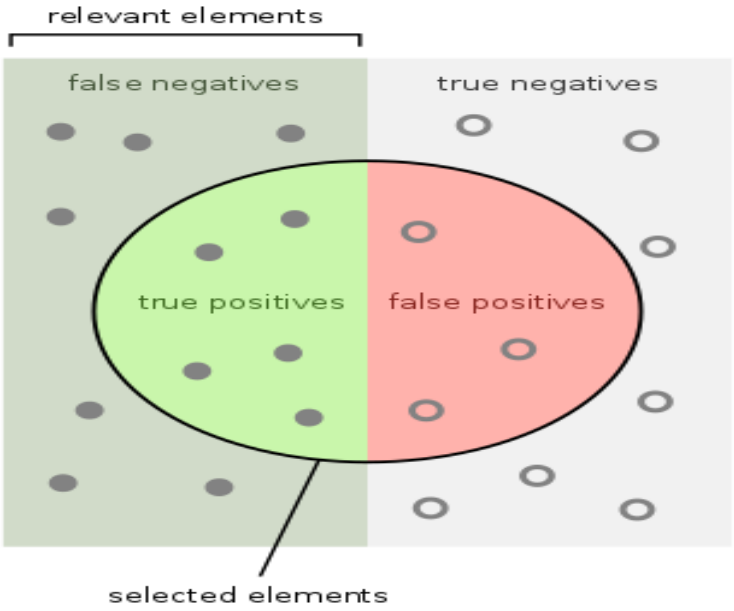


Figure 4-3 TP, TN, FP & FN

4.1.3 The Predictive Output

Since our main aim is to predict upcoming seizures, our output should be in a predictive form as well. Which means that depending on our main 3 aspects the outcome would be stating a percentage of the probability that a seizure will be coming.

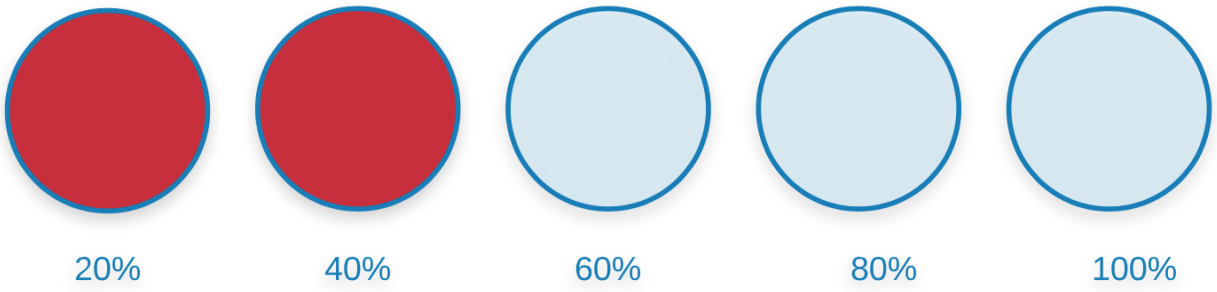


Figure 4-4 Predictive Output

The above figure is an example in which there is a probability of 40% that a seizure is coming.

4.1.4 The Features

Choosing the features was never an easy task as we had to do combinations of 13 feature with 2 kernels; Linear and RBF to get the best results. After a great deal of researching, observing and testing we found out that some features have a special effect on EEG signals. Those features are Coastline, Energy, Hjorth Parameters, Relative Spectral Power, Standard Deviation, Skewness, Kurtosis, Spectral Centroid, Variation Coefficient, Spectral Skewness, Hurst Exponent, Absolute Mean and Root Mean Square. For the sake of making sense of the results a brief description of those special features was needed.

4.1.4.1 Coastline

The Coastline is a fluctuation index that calculates the difference between the absolute two successive EEG signals.

4.1.4.2 Energy

The Energy of a signal is defined as a sum of square of magnitude.

4.1.4.3 Hjorth Parameters

In 1970, the Hjorth Parameters, which are indicators of statistical properties used in signal processing in the time domain, were introduced by Bo Hjorth. The parameters are complexity, activity and mobility. Moreover, they are commonly used in the analysis of EEG signals for feature extraction.

4.1.4.4 Relative Spectral Power

The Relative Spectral Power can refer to the concentration of any radiometric quantity as a function of the signal's wavelength.

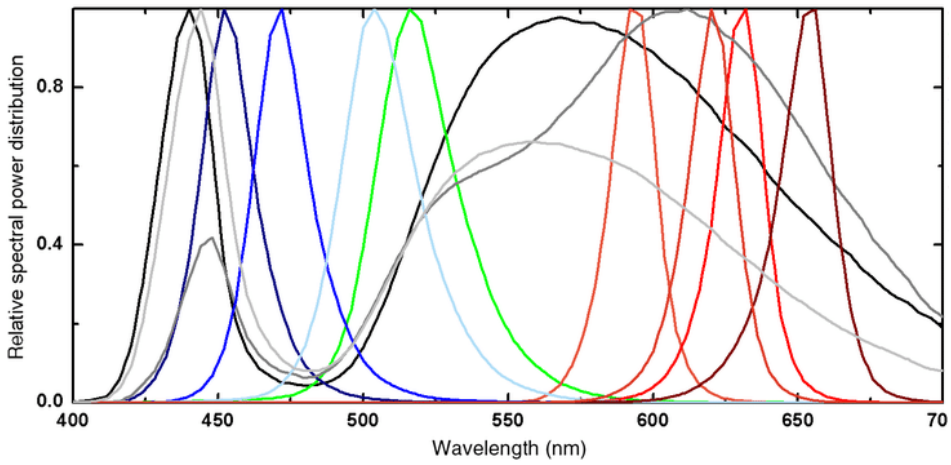


Figure 4-5 Relative Spectral Power Plot

4.1.4.5 Standard Deviation:

The Standard deviation is a statistical tool that measures how much the values of a group differ from the mean value for the group.

4.1.4.6 Skewness

Skewness is a measure of the asymmetry of the probability distribution of a real valued random variable about its mean.

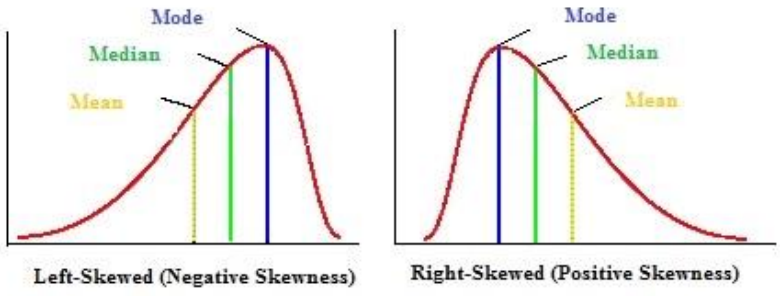


Figure 4-6 Skewness Plots

4.1.4.7 Kurtosis

Kurtosis is a statistical tool that measures the sharpness of the peak of a frequency-distribution curve.

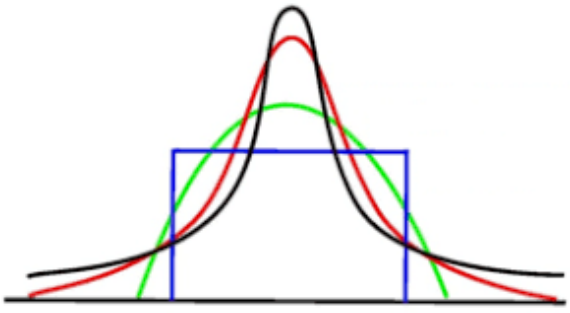


Figure 4-7 Kurtosis Plot

4.1.4.8 Spectral Centroid

The Spectral Centroid is a measure use in signal processing to characterize the spectrum.

4.1.4.9 Variation Coefficient

The Variation Coefficient is a measure of dispersion of frequency or probability distribution.

4.1.4.10 Spectral Skewness:

The Spectral Skewness measures the symmetry of the spectrum around it's arithmetic mean.

4.1.4.11 Hurst Exponent

The Hurst Exponent is a nonlinear feature that is used as a measure of long term memory of time series.

4.1.4.12 Absolute Mean

The Absolute mean is used to measure the central value of a set of positive values.

4.1.4.13 Root Mean Square

The Root Mean Square is defined as the square root of the sum of the mean of the values squared.

4.1.4.14 Fractal Dimension

The Fractal Dimension is a nonlinear feature. It is characterized in an index for describing the fractal sets and patterns by evaluating their intricacy as a proportion of the adjustment in detail to the change in scale. [35]

4.1.5 The Choice of the Features

When we tested all the combinations of those features mentioned above with both kernels, some results were very disappointing that we had to exclude. Regarding the fractal dimension feature, besides that the performance results from this feature and its combinations were not very promising, it also takes plenty of time to simulate which means that this feature is power hungry so we had to exclude it. Yet, some of the combinations showed very promising results. The following table shows average results of the best combinations with which kernel using the patients 7, 8, 11, 19, 20 & 23.

Table 4-1 Features Combinations Results

	Features Combinations	Sensitivity	Accuracy	Specificity	Kernel
1	Kurtosis, Skewness, CL & AM	77.92%	66.70%	63.97%	Linear
2	SD, CL, RMS & AM	76.40%	57.77%	50.85%	RBF
3	SD, CL, RMS & AM	79.15%	66%	62.50%	Linear
4	Hurst Exponent	68.20%	61.20%	61.81%	RBF
5	Energy & CL	73.25%	57.07%	51.24%	RBF
6	Hurst Exponent	59.90%	57.77%	54.21%	Linear
7	SD, Hurst Exponent & RMS	72.71%	63.12%	59.50%	Linear
8	RMS, Hurst Exponent & AM	65.64%	61.41%	59.30%	RBF
9	SD & Hurst Exponent	65.84%	60.76%	59.09%	RBF
10	RMS	77.26%	42.53%	37.46%	Linear
11	Hurst Exponent & CL	71.09%	59.85%	54.80%	RBF
12	RSP, Spectral Centroid & VC	86.13%	55.68%	47.12%	RBF
13	Fractal Dimension	64.08%	60.82%	61.24%	RBF
14	Fractal Dimension & Hurst Exponent	70.55%	56.89%	59.44%	RBF
15	Fractal Dimension & CL	68.38%	54.45%	56.07%	RBF

As shown above, two of the combinations showed auspicious results. That's why, we decided that we will use these two combinations in our project. Which means that our project is having two parallel tracks, one for each combination.

In order to be able to deal with our chosen features, we had to know more details about their nature and how they vary with different datasets. What makes them special and how they operate became our main scope and point of interest. Correspondingly, a more detailed description of each and every feature of our chosen ones was needed.

4.1.5.1 Absolute Mean

The Absolute Mean is a statistical tool to measure the central value of an absolute value set of numbers. What makes this feature special is that it gives us a clue about the variability in a dataset so that we would be capable of differentiating between EEG signal in different states.

$$\text{Absolute Mean} = \left(\sum_{i=0}^{n} |Xi| \right) /n$$

4.1.5.2 Root Mean Square

The Root Mean Square measures the square root of the arithmetic mean of the squares of a set of values. This Feature is a particular case of the generalized mean with an exponent of 2. That's why it is also known as the quadratic mean. RMS can also be defined for a continuously varying function in terms of an integral of the squares of the instantaneous values during a cycle. Not only this but also in estimation theory the RMS is used via an estimator to conduct the imperfection of the fit of the estimator to the data. Accordingly, this statistical feature is used in electrical engineering, error calculation and speed calculation. What makes this feature special is that it gives us a sense for the typical size of the numbers.

$$x_{\text{rms}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}.$$

4.1.5.3 Standard Deviation

The standard deviation in statistics is a measure that is utilized to evaluate the amount of variation or scattering of a set of data values from their mean. If the value of the standard deviation is relatively considered small this is an indication that the data points tend to be near the mean value of the set, while a high standard deviation value indicates that the data points are spread out over a more extensive scope of values. It is considered the square root of its variance and has a useful feature that it is expressed in the same units as the data. Referring to our scope if the standard deviation value is small this may refer to a normal state while, if the standard deviation is large, so does the EEG fluctuations which may indicate to the high rhythmic activity during a seizure or a pre-seizure. The standard deviation is calculated according to the following equation:

$$\text{Standard Deviation} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(Xi - \bar{X})^2}$$

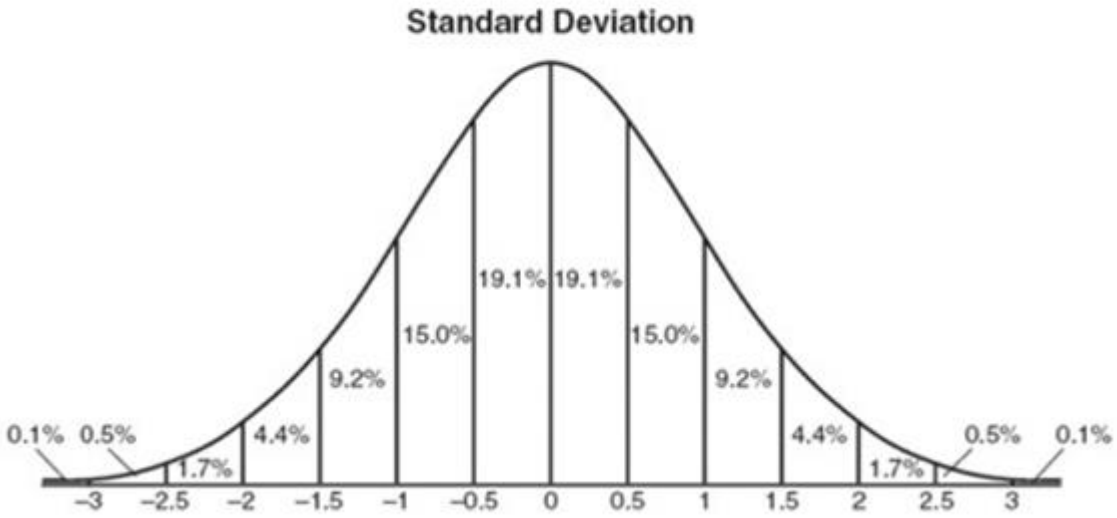


Figure 4-8 Standard Deviation Plot

4.1.5.4 Coastline

Despite the fact that the Coastline is viewed as one of the simplest features however, it proved efficiency in working on our scope. It calculates the distance between each two successive data points on the EEG signal in an absolute manner then windows those differences into the required window size according to the following equation:

$$\mathrm{CL(k)} = \sum_{i=1}^{N} \mathrm{abs(x[i + (k - 1)N] - x[}i - 1 + (k - 1)N\mathrm{])}$$

4.1.5.5 Hurst Exponent

The Hurst Exponent is used as a measure of long-term memory of time series.

It relates to the autocorrelations of the time series, and the rate at which these autocorrelations decrease as the trail or the lag between pairs of values increases.

The Hurst exponent is referred to as the "index of dependence" or "index of long-range dependence". It quantifies the relative tendency of a time series either to regress strongly to the mean or cluster to a long-term equilibrium. A value H in the range 0.5 – 1 indicates a time series with long-term positive autocorrelation, meaning both that a high value in the series will most likely be trailed by another high value and that the values quite a while into the future will likewise tend to be high. A value in the range 0 – 0.5 indicates a time series with long-term switching between high and low values in adjacent pairs, meaning that a single high value will probably be followed by a low value and that the value after that will tend to be high, with this tendency to switch between high and low values lasting a long time into the future. A value of H=0.5 can indicate a completely uncorrelated series, but in fact it is the value applicable to the series for which the autocorrelations at small time lags can be positive or negative but where the absolute values of the autocorrelations decay exponentially quickly to zero. [1]

Rescaled range (R/S) analysis:

To estimate the Hurst exponent, one must regress the rescaled range on a time span n of observations. To do this, a time series of full-length *N* is divided into a number of shorter

time series of length *n* = *N*, *N*/2, *N*/4, ... and the rescaled range is calculated for each of the smaller time series.[1]

For each chunk of observations of length n, $X = X_1, X_2, .. X_n$, compute:

- The mean of the time series,

$$m = \frac{1}{n}\sum_{i=1}^{n} X_i$$

- A mean centered series by subtracting the mean from the series,

$$Y_t = X_t - m \qquad for\ t = 1, 2, 3, .. n$$

- The cumulative deviation of the series from the mean by summing up the mean centered values,

$$Z_t = \sum_{i=1}^{t} Y_i \qquad for\ t = 1, 2, 3, .. n$$

- The Range (R), which is the difference between the maximum value of the cumulative deviation and the minimum value of the cumulative deviation,

$$R(n) = \max(Z_{1,} Z_2, .., Z_n) - \min(Z_{1,} Z_2, .., Z_n)$$

- The standard deviation (S) of the mean centered values,

$$S(n) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (X_i - m)^2}$$

- The rescaled range by dividing the Range (R) by the standard deviation (S).

- Finally, average the rescaled range over all the chunks.

The Hurst Exponent can be estimated by plotting $\ln(\frac{R(n)}{S(n)})$ as a function of ln(n) and fitting it into a straight line; the slope of the line gives H.[1]

For example, if we have the following data:

Chunk #	**1**
Data	0.04
	0.02
	0.05
	0.08
	0.02
	-0.17
	0.05
	0.00

We first divide the data into 3 different chunks as follows:

1. Division 1 – one chunk of 8 observations
2. Division 2 – two chunks of 4 observations each
3. Division 3 – four chunks of 2 observations each

Chunk #	1
Data	0.04
	0.02
	0.05
	0.08
	0.02
	-0.17
	0.05
	0.00

Chunk #	1	2
Data	0.04	0.02
	0.02	-0.17
	0.05	0.05
	0.08	0.00

Chunk #	1	2	3	4
Data	0.04	0.05	0.02	0.05
	0.02	0.08	-0.17	0.00

After dividing the data into chunks, we perform the following calculations on each chunk:

1. First, we calculate the mean of the chunk.

Division #	1	2		3			
Chunk #	1	1	2	1	2	3	4
Mean	0.01	0.05	-0.02	0.03	0.07	-0.08	0.03

2. Then, we create a mean centered series by subtracting the mean from the original data series.

Chunk #	1
Mean Centered Series	0.03
	0.01
	0.04
	0.07
	0.01
	-0.18
	0.04
	-0.01

Chunk #	1	2
Mean Centered Series	-0.01	0.05
	-0.03	-0.15
	0.00	0.07
	0.03	0.03

Chunk #	1	2	3	4
Mean Centered Series	0.01	-0.02	0.10	0.02
	-0.01	0.02	-0.10	-0.02

3. Then, we calculate the cumulative deviation by summing up the mean centered values.

Division #	1	2		3			
Chunk #	1	1	2	1	2	3	4
Cumulative Deviation Y(n)	0.01	-0.01	0	0	0	0	0

4. Then, we calculate the Range (R), which is the difference between the maximum value of the cumulative deviation and the minimum value of the cumulative deviation.

Division #	**1**	**2**		**3**			
Chunk #	**1**	**1**	**2**	**1**	**2**	**3**	**4**
Range (R)	0.18	0.04	0.15	0.01	0.02	0.10	0.02

5. Next, we calculate the standard deviation (S) of the mean centered values.

Division #	**1**	**2**		**3**			
Chunk #	**1**	**1**	**2**	**1**	**2**	**3**	**4**
Standard Deviation (S)	0.0722	0.0218	0.088	0.01	0.02	0.1	0.02

6. Finally, we compute the ratio of the range R to the standard deviation S. This is also known as **the rescaled range.**

Division #	**1**	**2**		**3**			
Chunk #	**1**	**1**	**2**	**1**	**2**	**3**	**4**
Rescaled Range (R/S)	2.49	1.83	1.705	1	1	1	1

Once we have the rescaled range for all the chunks, we compute the mean of each Division and note it along with the number of samples in each chunk of that Division as shown.

Division #	**Number of observations/division**	**Rescaled Range**
1	8	$(\frac{R}{S})_8$
2	4	$\frac{1}{2}((\frac{R}{S}_4)+(\frac{R}{S}_4))$
3	2	$\frac{1}{4}((\frac{R}{S}_2)+(\frac{R}{S}_2)+(\frac{R}{S}_2)+(\frac{R}{S}_2))$

Division #	1	2		3			
Chunk #	1	1	2	1	2	3	4
Average Rescaled Range	2.49	1.7675		1			

Next, we calculate the natural logarithmic values for the size of each region and for each region's rescaled range.

Size	ln(size)	Average Rescaled Range (R/S)	ln(average R/S)
8	2.08	**2.49**	0.91
4	1.39	**1.7675**	0.57
2	0.69	**1**	0

We can finally estimate the Hurst Exponent (H) by plotting ln(average R/S) as a function of ln(size) and fitting it into a straight line to be able to get the slope.

However, this example can not be an accurate indication of the Hurst Exponent since this data set is too small to draw any conclusions. So, to be able to kind of have a reasonable estimation, we need to have at least around 100 data points.[25]

Interpreting the Hurst Exponent:

Using the Hurst Exponent, we can classify the time series into types and gain some insight into their dynamics. Here are some types of time series and the Hurst exponents associated with each of them.

A Brownian time series: "Brownian motion" as we previously studied is the random movement resulting from continuous bombardment of particles. Likewise, is the Brownian time series (also known as a random walk), there is no correlation between the current and future observations; being higher or lower than the current observation is equally likely. Series of this kind are hard to predict. Such type of time series is indicative to a Hurst Exponent of approximately 0.5. [26]

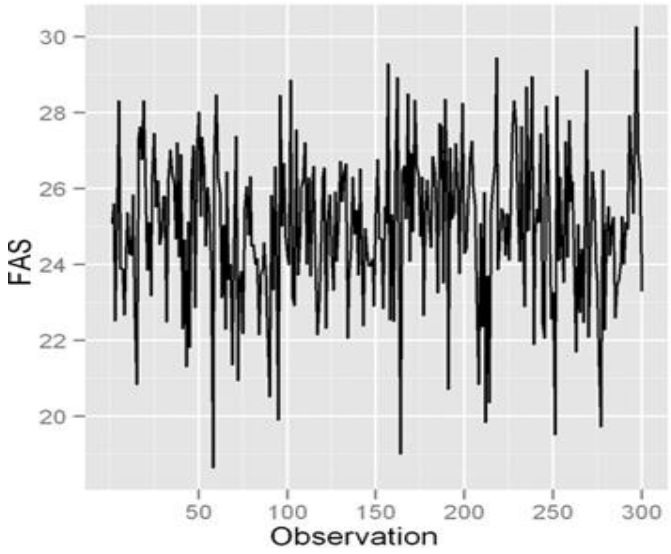


Figure 4-9 Brownian time series with Hurst exponent= 0.53.[26]

An anti-persistent time series: An anti-persistent time series reverses itself more often than a random series would. In an anti-persistent time, series (also known as a mean-reverting series) an increase will most likely be followed by a decrease or vice-versa. This means that future values have a tendency to regress to a long-term mean. Such type of time series is indicative to a Hurst Exponent falling in the range of 0-0.5, and the closer the value is to 0, the stronger is the tendency for the time series to revert to its long-term means value. [26]

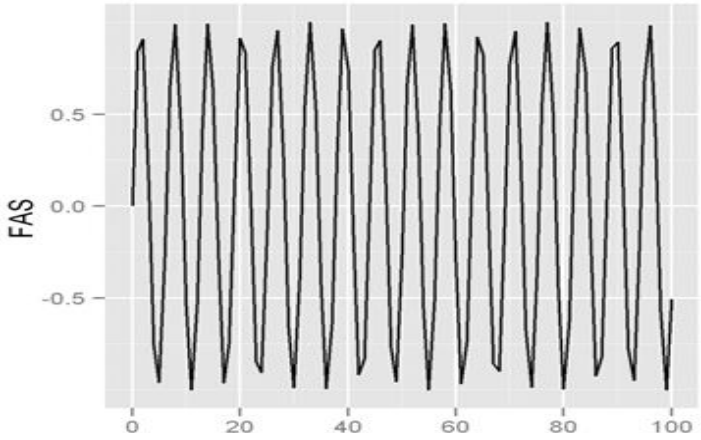


Figure 4-10 Anti-persistent time series with Hurst exponent= 0.043.[26]

A persistent time series: In a persistent time series any change in the value will be influencing all future predictions. So any single value will be closely related to the previous one. Such type of time series is indicative to a Hurst Exponent falling in the range of 0.5-1, and the closer the value is to 1, the stronger is the trend.[26]

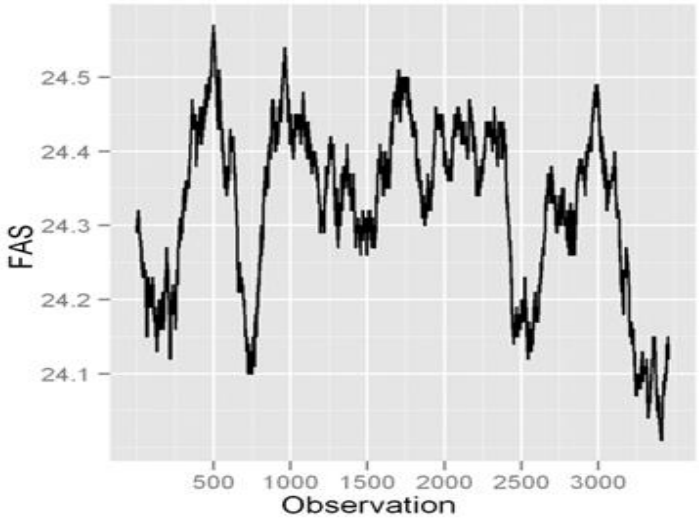


Figure 4-11 Persistent time with Hurst exponent= 0.95.[26]

4.1.6 Segmentation

In Marketing, segmentation means to divide the marketplace into chunks or segments, which are profitable, accessible, actionable, definable and have a growth potential. Which means that, a company would find it impossible to target the whole market due to plenty of factors including time, cost and effort restrictions. That's why it needs to have definable segments. Accordingly, we decided to do the same process on data and call it data segmentation. In other words, data segmentation is the process in which the data is being divided into equal parts which makes dealing with it more effective.

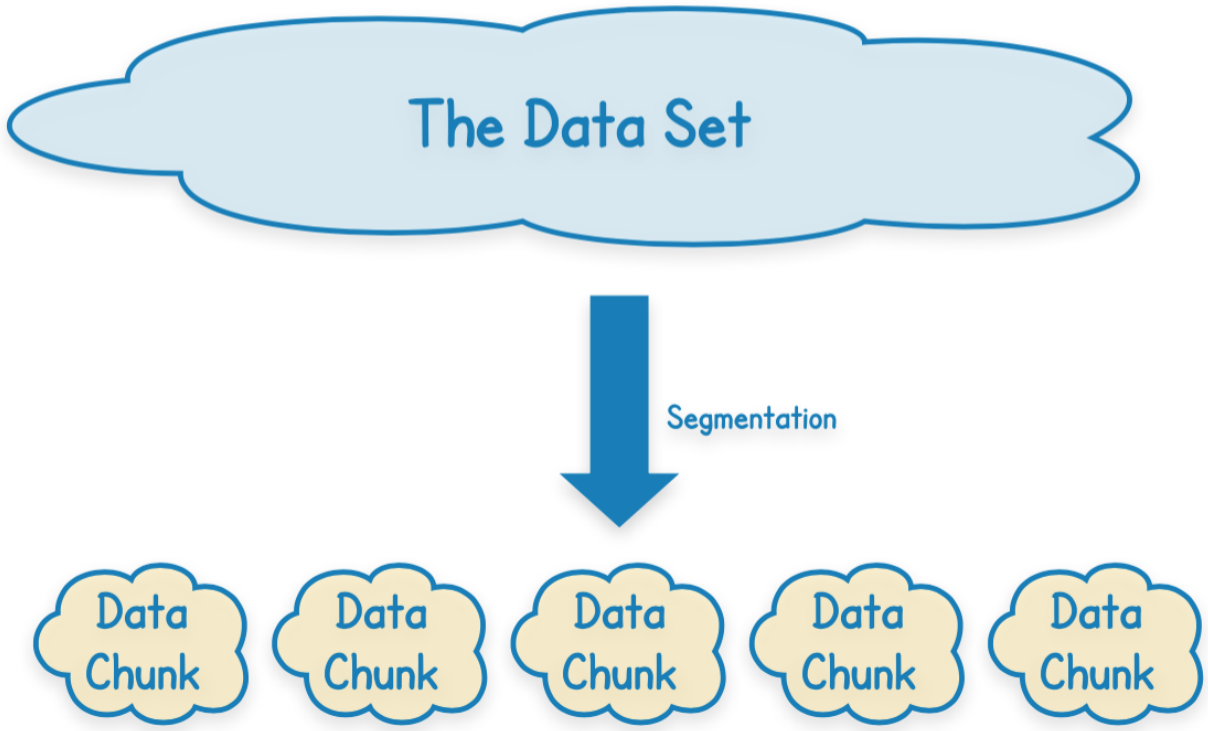


Figure 4-12 Segmentation Process

As shown in the figure above, the segmentation process makes dealing with a small data chunks possible. It allows us to easily conduct an analysis on our data. Which means that the identification of potential opportunities and challenges based within it becomes possible.

We believed that if we applied the data segmentation process on our testing data in the testing block, it will boost the results and the performance of the system. Which mean that, more detailed characteristics and properties of the data will be captured. In other words, if we decided to divide the data into 5 segments, instead of giving the testing block a huge chunk of data the testing block with be given 5 small chunks of data as one after the other and then the decision would be taken on those chunks concatenated. Although this process will increase the complexity of the code but we believe that the results would be worth it.

We found out that number of segments is one of the major factors affecting the results. That's why we had to try a range of number of segments in order to know which of them yields the best results and whether they actually boost our system's performance or not.

4.1.6.1 Segmentation Approaches

We had 2 approaches for the decision making and the classification, either we make our decision based on the segment, or we make it based on the 30-minute period.

1. For the **segment-based decision**, each segment has its own test labels that enter the classifier with the SVM trained data. So, if we have 5 segments for example, we will accordingly have 5 test labels and 5 classifications, and then finally we apply the majority rule on these 5 classifications to decide whether the classification tends towards 0 or 1.

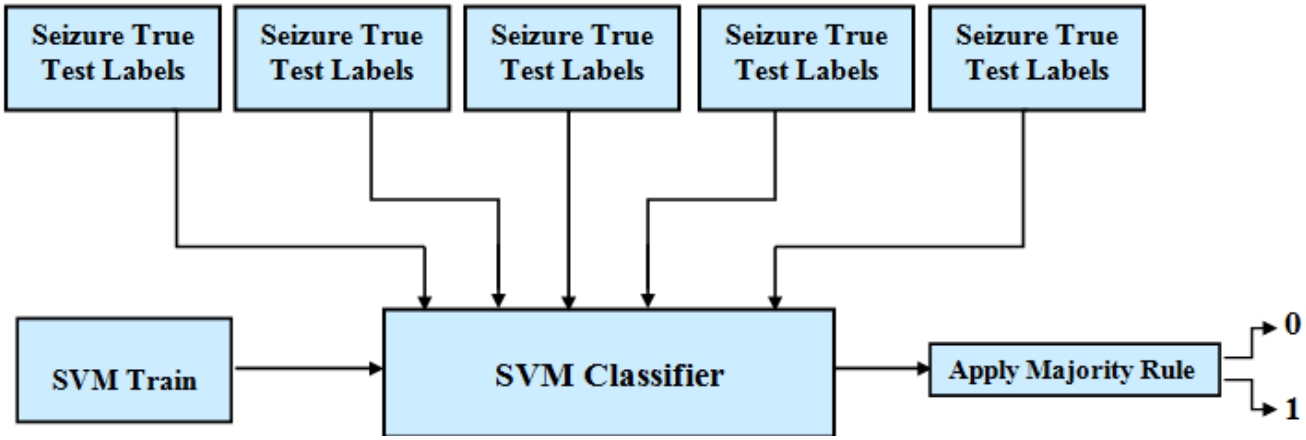


Figure 4-13 Segment Based Decision

2. For the **30 minute-based decision**, we actually apply the majority rule a bit earlier in the process. For example, if we have 5 segments, we analyze the true test labels of each segment to check if it has more ones than zeros and vice versa. Then we keep record of all 5 true test labels, and in the end, if more than half of the total number of segments have more ones than zeros, then it is safe to assume a final true test label vector of all ones. The final true test label vector enters the SVM classifier with the SVM trained data, which will result in a single final classification in the end.

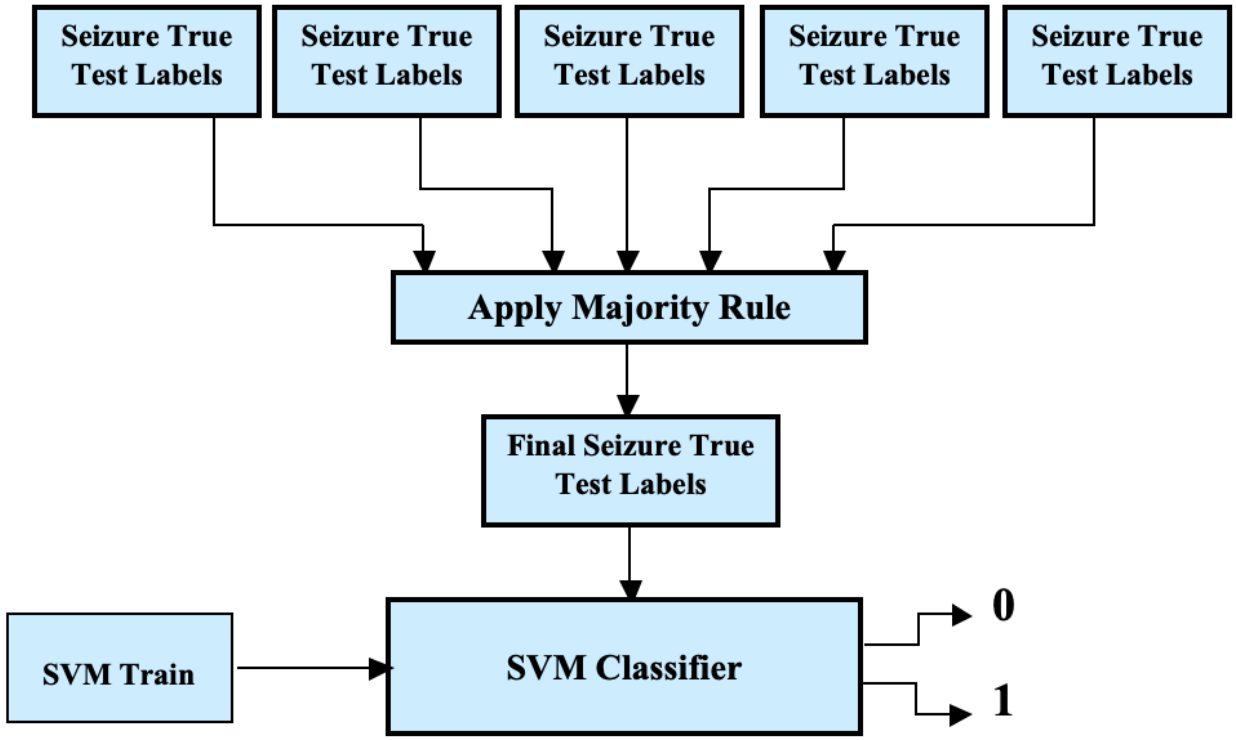


Figure 4-14 30 Minute Based Decision

4.1.6.2 Results

We tried varying the number of segments and testing the effect of each number on our chosen features and performance results. Finally, for each feature model, we plotted the **number of segments** versus our performance metrics: **Sensitivity** and **Specificity**, to study and analyze how varying the number of segments affects each one of them.

Using the segment-based decision:

I. Linear Features + Linear Kernel Model:

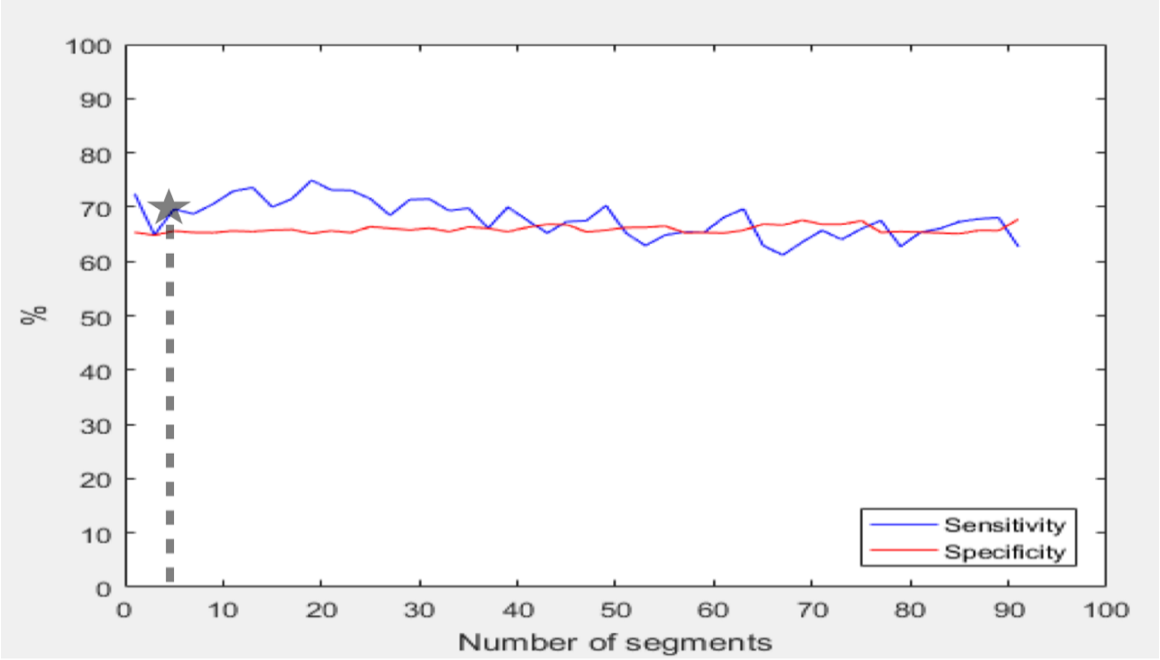


Figure 4-15 Segment Based Decision, Linear Features & Linear Kernel Plot

II. Non-Linear Feature + RBF Kernel Model:

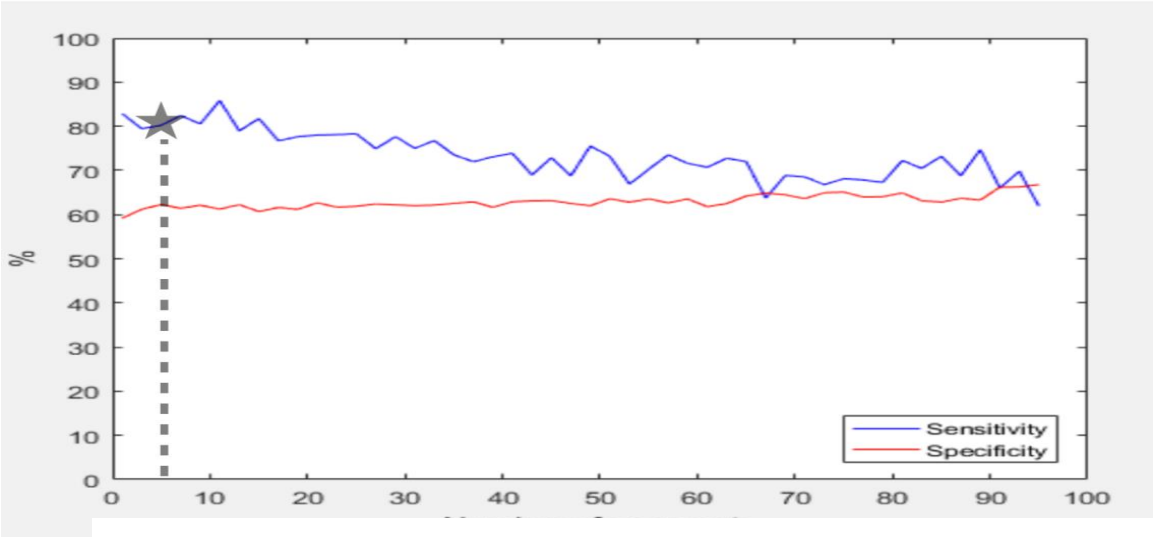


Figure 4-16 Segment Based Decision, Non-Linear Features & RBF Kernel Plot

Using the 30 minute-based decision:

I. Linear Features + Linear Kernel Model:

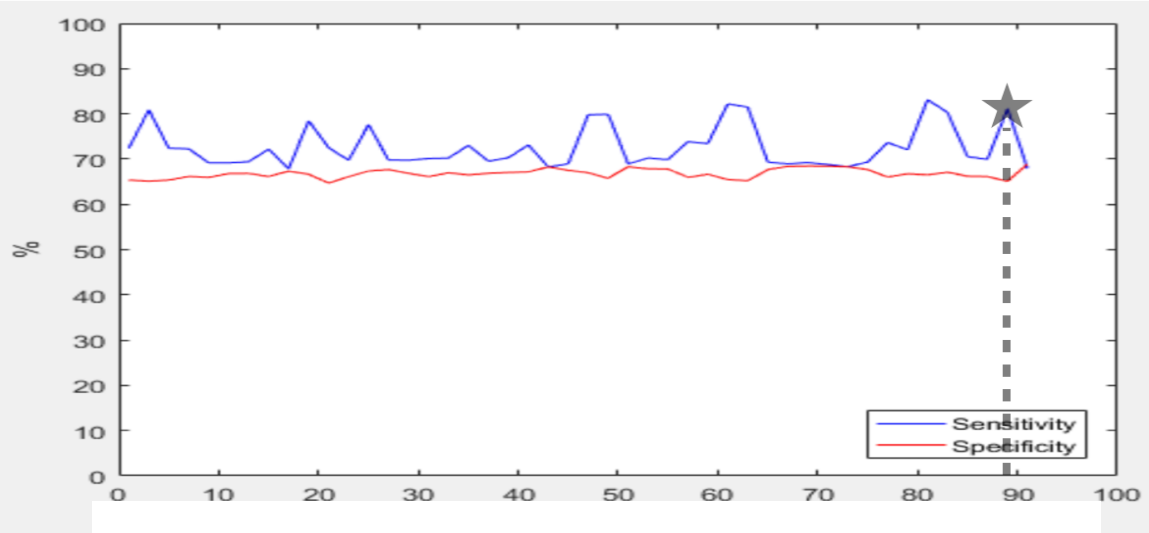


Figure 4-17 30 Minute Based Decision, Linear Features & Linear Kernel Plot

II. Non-Linear Feature + RBF Kernel Model:

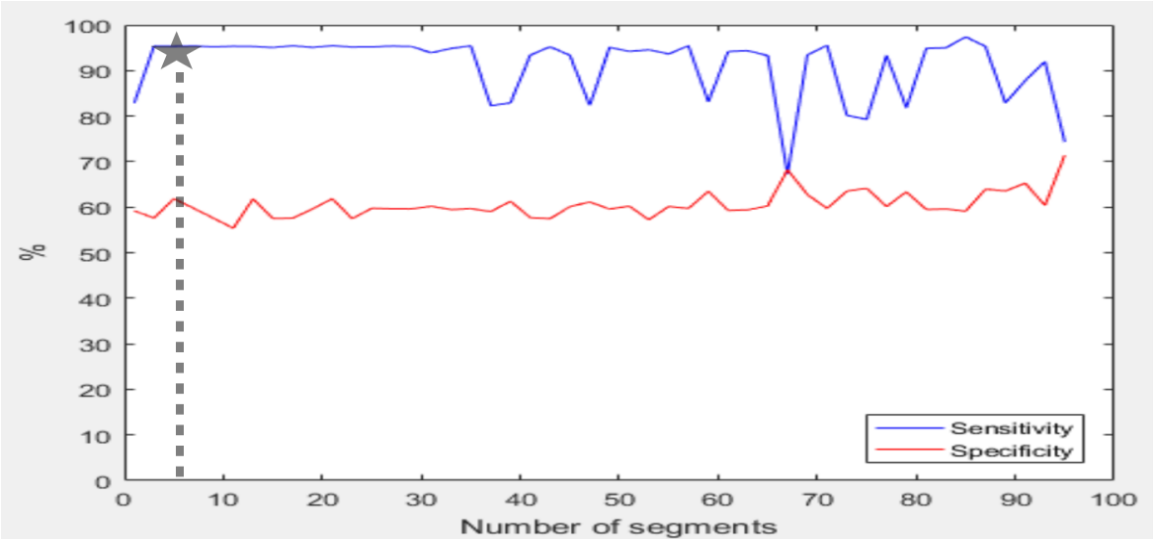


Figure 4-18 30 Minute Based Decision, Non-Linear Features & RBF Kernel Plot

Taking a glimpse at these graphs, we can now easily study the effect of increasing/decreasing number of segments. One might have thought that increasing the number of segments will always yield better results, but since this is machine learning so there's nothing we can be absolutely certain of. So, greater number of segments does not necessarily mean better performance results, and our justification to this, is that our model might have undergone "Overfitting" which is as unwanted as "Underfitting".

We can also conclude from the above graphs that the 30 minute-based decision yields better performance results than the segment-based decision. In addition, we can easily spot which number of segments yields higher Sensitivity and Specificity. As marked on both graphs, around **89 segments** for the Linear Features + Linear Kernel Model and around **5 segments** for the Non-Linear Feature + RBF Kernel Model.

Finally, using these graphs and feature models, we can benefit from multiple aspects that not everyone is aware of. We all know that the patient will need a highly sensitive product that will, without a doubt indicate most of the times that there is an upcoming seizure when there is an actual seizure coming. Thus, we are talking about "high

Sensitivity" here, so one might think that "low Specificity" is not that much of an issue, however, "low Specificity" means that the device's battery will be drained in no time, and no patient would want that. Therefore, we thought about creating an "Adaptive" model by merging the best of both worlds. We can create a model that when fully charged, and at full capacity, can operate with high Sensitivity and low Specificity, then after consuming most of the battery's power; it can then shift to high Specificity and average Sensitivity, putting in mind that the device will supposedly be charged soon.

5

HARDWARE PHASE

The hardware phase is an exceedingly challenging stage in which we had to change our whole mindset and direct our way of thinking from the software perspective to the hardware point of view. Which means that we had to start with building our high-level system architecture relative to the software module which lead us to the first step in this stage which is the RTL. RTL stands for Register Transfer Level which is a design abstraction that models a synchronous digital circuit in terms of the flow of the EEG data between the registers and the logical operations performed on those EEG signals. This implies that your VHDL code describes how data is transformed as it is passed from register to register. That's why, we had to consume a lot of time and effort to get the optimum RTL design. We implemented our modules using both Verilog and VHDL languages using Modelsim program. Not only did we optimize our code to ensure that it satisfies our requirements like low power but also put into consideration the most important factor which is synthesizability, so that our code would actually be compatible with the FPGA. Thereafter, we used Vivado which is s a software suite for synthesis and analysis of HDL designs to examine our system. Thus, this chapter will be discussing the flow of the hardware phase mentioned above.

5.1 High Level Design

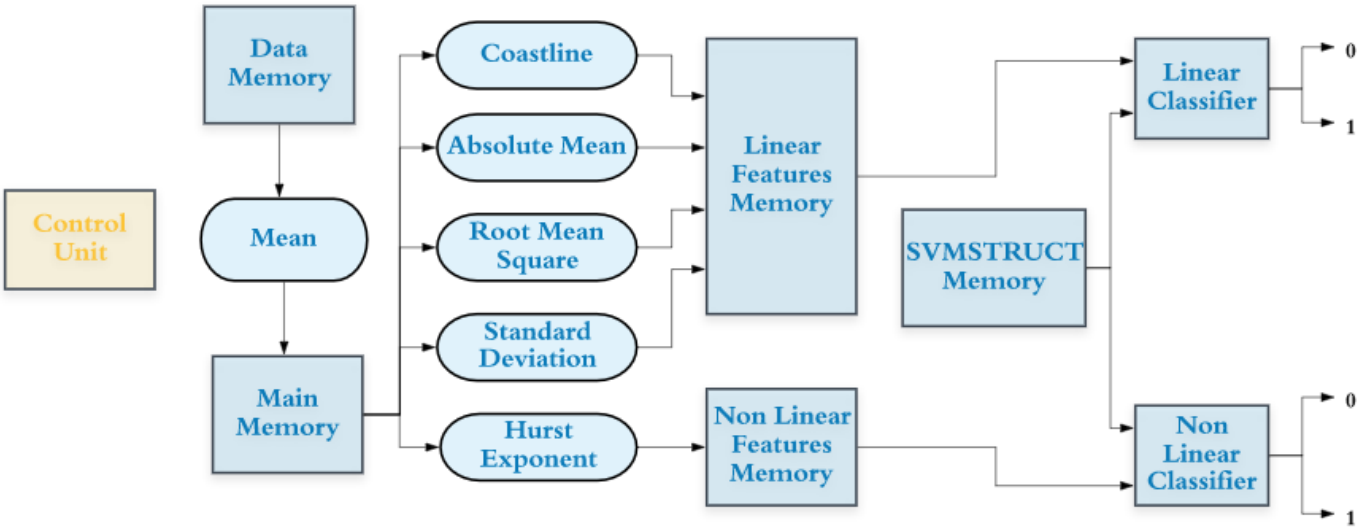


Figure 5-1 High level RTL block diagram

The diagram above shows the whole system flow. First of all, we have the control unit which is the block responsible for supplying all the blocks with the signals needed. Then comes the data memory which is the rom memory having our patients' dataset saved in. This patients' dataset is the data that should be used by the features shown in the oval shapes in order to produce the features' captured data. Unfortunately, the problem rose here as some of those features like the Hurst Exponent and Standard Deviation require the mean of the data along with the data itself. That's why, we needed a work around to solve the issue that showed up. We decided to use another memory which is the Main Memory. Main Memory, is a rom memory having our patients' dataset and its mean conducted via the mean feature block. As a consequence, the Main Memory is now responsible of feeding the features' blocks in order to yield the features' captured data. As mentioned in chapter four, we implemented our system using the two tracks that yielded the best results as clearly shown in the diagram above. Then all of the features' captured data conducted from linear features which are used by the first track are saved in the Linear Features Memory. As for the features' captured data conducted from a nonlinear feature which is track two are saved in the Non Linear Features Memory. Finally, a linear classifier and a nonlinear RBF classifier are used to make a decision for each of the tracks.

5.1.1 Features Blocks

Each and every feature needed to be accurately, precisely and perfectly implemented to map the Matlab code in order to get same performance results. That's why a detailed RTL for each feature is needed.

5.1.1.1 RTL Coastline

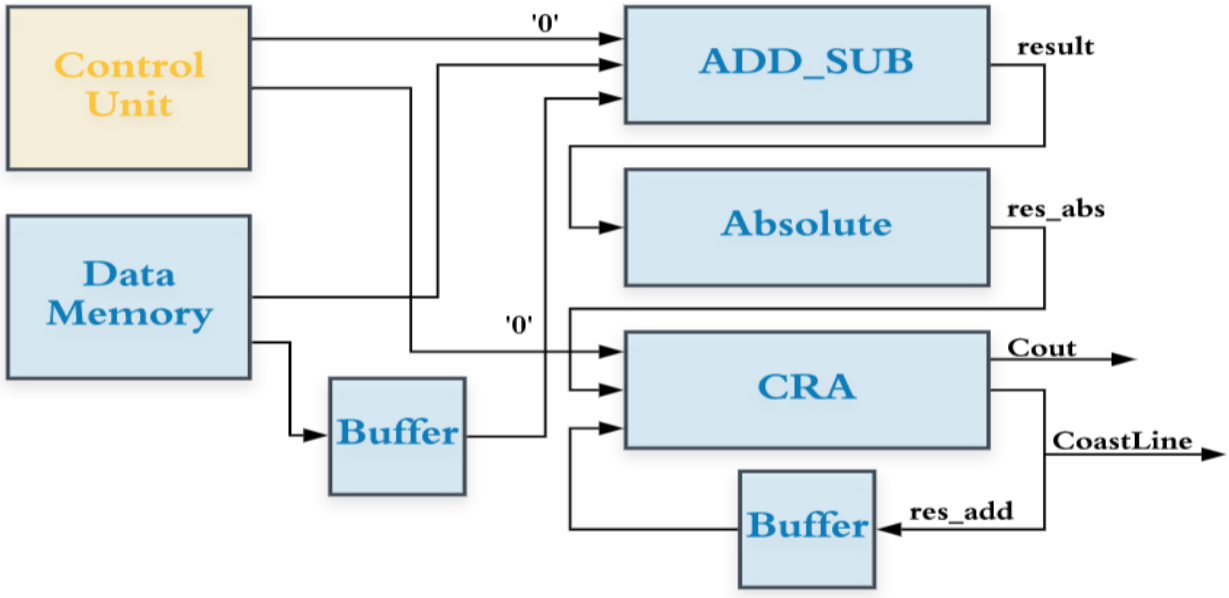


Figure 5-2 RTL coastline block diagram

The coastline feature as we mentioned before is characterized by calculating the summation of the absolute distances between each two successive EEG data points. According to the previous diagram that represents the RTL implemented coastline feature, the control unit here is only responsible for the controller bit of the "Add_Sub" block that is switched to '1' for adding the values and switched to '0' for subtracting the values entering the block, and the carry in bit regarding the "CRA" block. The "Data Memory" keeps supplying the "Add_Sub" block with the data samples, the buffer is used here so that the "Data Memory" can pass each data sample and its delayed version at a time, then the "Add_Sub" block keeps subtracting each two successive data points respectively by setting the controller bit to '0' for subtraction, this subtracted value passes through an absolute block for calculating the absolute distance before entering to the "CRA" block that is responsible for adding these absolute distances that are calculated

from the previous blocks, the buffer is also used in the "CRA" block for storing the final added value for adding on it the next absolute distance until the data samples are done according to the size of the window, at this point the coastline is considered to be calculated as the final output of the "CRA" block.

5.1.1.2 RTL Absolute Mean

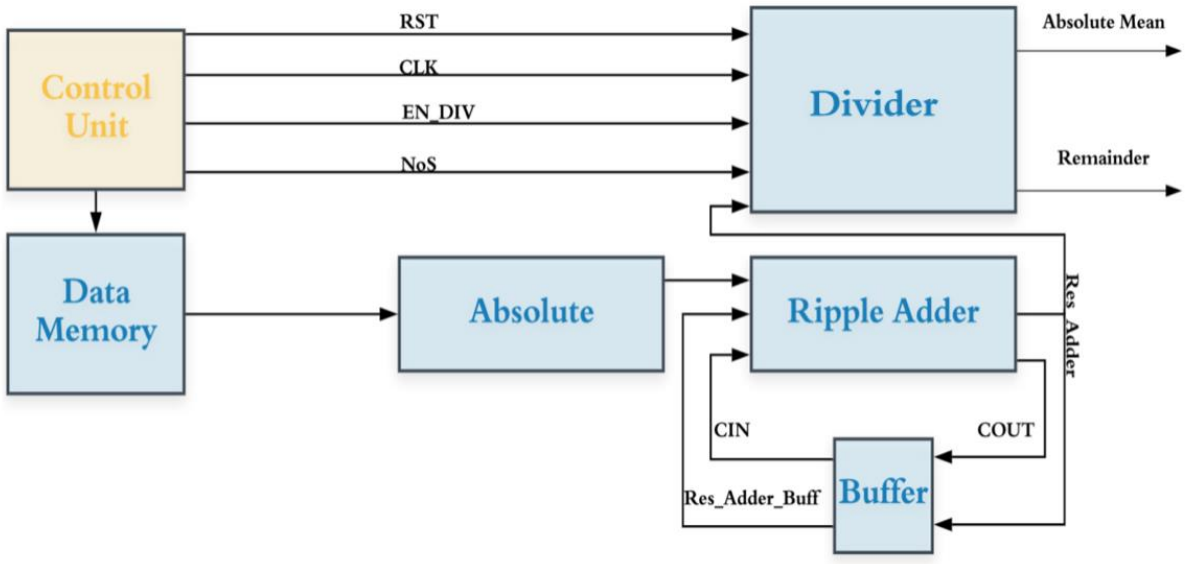


Figure 5-3 RTL absolute mean block diagram

The absolute mean feature flow once of the simplest features. The patients' data is fetched from the data memory with data samples which are equal to the predefined number of samples. Then, the data is used as an input to the absolute block. The absolute block gets the absolute value for all of the data. Which means that the output of this block is positive data. After that, this positive data is given to the ripple adder as input so that it would be added. the control unit responsibility pops up here as the divider block needs some signals to start operating correctly which are RST, CLK, EN_DIV and NoS. RST is the signal that is used to reset the module to its initial state. CLK is the clock that the divider block will be using in order to be able to perform the division process. Then comes the EN_DIV which is on of the most important signals used by the divider as it acts as the main switch of the block. Not only does this signal turn on the divider but also

turns it on in a specific time which is just after the result of the ripple adder is yielded. The NoS signal is the signal that supplies the data memory and the divider with the predefined number of samples that the feature will be operating on. After a number of clock cycles the output of the divider which is the absolute mean is conducted and ready to be served to the next step.

5.1.1.3 RTL Root Mean Square

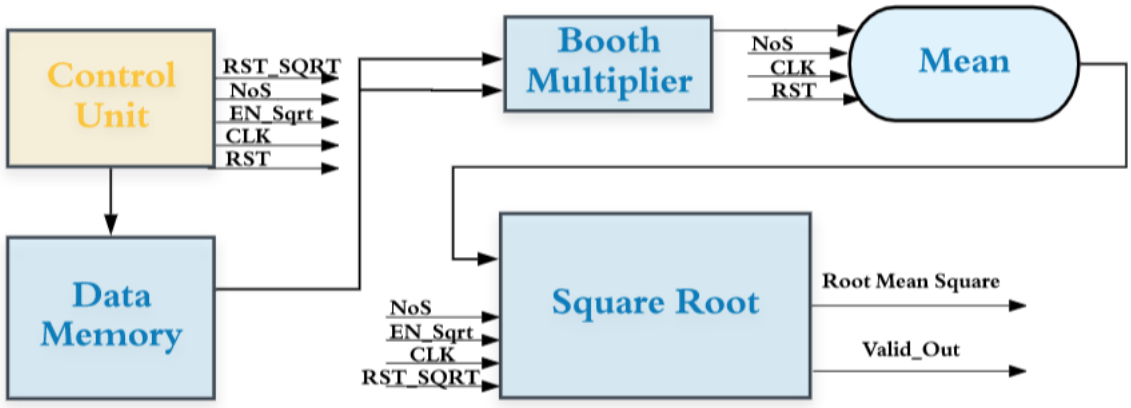


Figure 5-4 RTL root mean square block diagram

The RMS feature is a main block that is composed of 3 other blocks, the Booth Multiplier, Mean and Square Root block.

The Control Unit here is responsible for supplying the RMS feature with the system's clock, reset, number of samples, enable square root and reset square root which both switch to '1' whenever the Mean block finishes and is ready to hand the result to the Square Root block. The Data Memory keeps supplying the Booth Multiplier block with data samples according to the predefined number of samples or window size.

Feature Flow:

1. Data from the Data Memory will enter the Booth Multiplier twice for squaring. The booth multiplier will output the square value instantly (ie: in the same clock cycle).
2. Afterwards, the square value will enter the Mean block.
3. As soon as the Mean block finishes its computations, the Square Root reset (**RST_SQRT**) and enable (**EN_SQRT**) signals will be switched to '1'.
4. The **RST_SQRT** will be switched to '0' the following clock cycle.
5. Finally, the Root Mean Square value will be computed as soon as **Valid_Out** switches to '1'.

Booth Multiplier:

Multiplication involves two basic operations: **generation of the partial products** and their **accumulation**. Therefore, there are two possible ways to speed up the multiplication, either to **reduce the number of partial products** or **accelerate their accumulation**. A smaller number of partial products also reduces the complexity, and as a result, reduces the time needed to accumulate the partial products. So as an alternative to this type of multiplication, we use Booth Multipliers which are based on the Booth algorithm.

The Booth algorithm is implemented in two steps: **Booth Encoding** and **Booth Selection**, and it works on reducing the number of partial products generated by taking into account two bits of the multiplier at a time. This results in a speed advantage over other multiplier architectures such as the serial multiplier. Booth multipliers are known for their fast calculations, the output is calculated in combinational logic (i.e. in the same clock cycle that the inputs change, the output changes accordingly). Since a Booth multiplier reduces the number of partial products, thus, it's extensively used in multiplication with long operands (>16 bits).

The main disadvantage of Booth multipliers is the complexity of the circuit needed in the Booth encoding.

5.1.1.4 RTL Standard Deviation

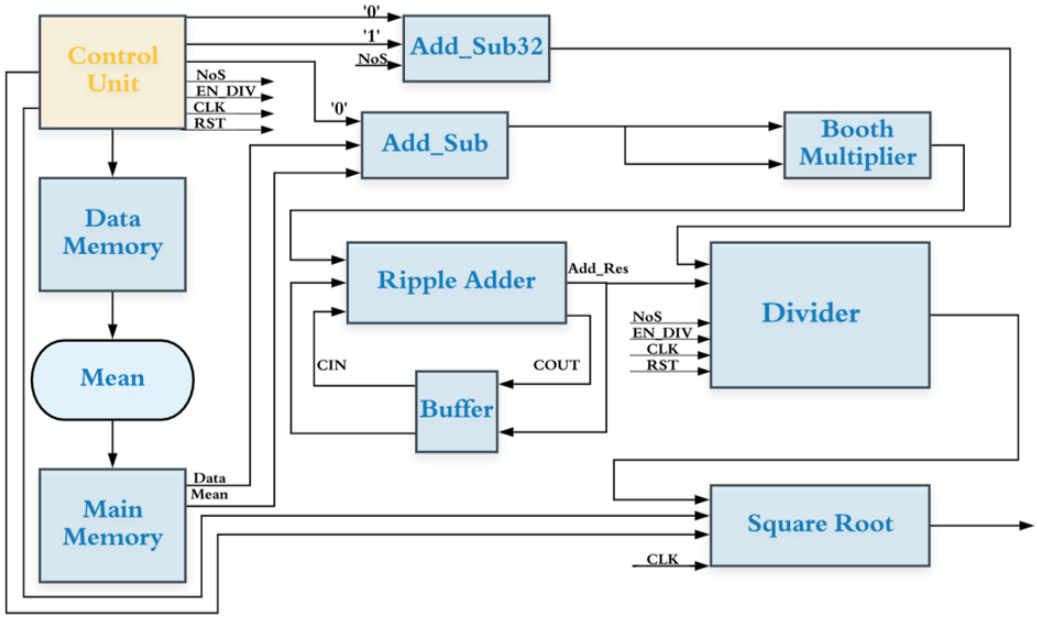


Figure 5-5 RTL standard deviation block diagram

The standard deviation block can be said to be the second most complicated block after the Hurst exponent. As mentioned in 5.1.1. This block needs the mean of the data along with the data itself. That's why the main memory appears in the diagram. The data and the mean is given as an input for the Add_Sub block along with the '0' signal from the control unit which is a selection line for the block that states that Add_Sub is used as a subtractor and not an adder. After that, the output of this block should be squared. Therefore, the output is fed to the booth multiplier as two inputs so that the square of the input would be conducted. The Add_Sub and Booth Multiplier stage is done for a number of data samples and their mean. The number of samples that will be used is determined by the NoS signal. Subsequently, all of the squared values yielded from the booth multiplier would be added using the ripple adder. To conduct the variance, the output value of the ripple adder which is Add_Res should be divided by the number of samples - 1. That's why we needed another Add_Sub unit to subtract the value 1 from the number of samples. Not only does the divider require the nos-1 but also needs a CLK, RST, NoS & EN_DIV as discussed in 5.1.1.2. After a number of a clock cycles the variance appears as the output of the divider block. In order to obtain the standard deviation, the square

root of the variance should be done through the square root block which also requires some signals from the control like the CLK to be able to do the process itself, EN_SQRT to to turn on the block in the desired time and finally finally RST_SQRT which resets the module to its initial state.

5.1.1.5 RTL Hurst Exponent

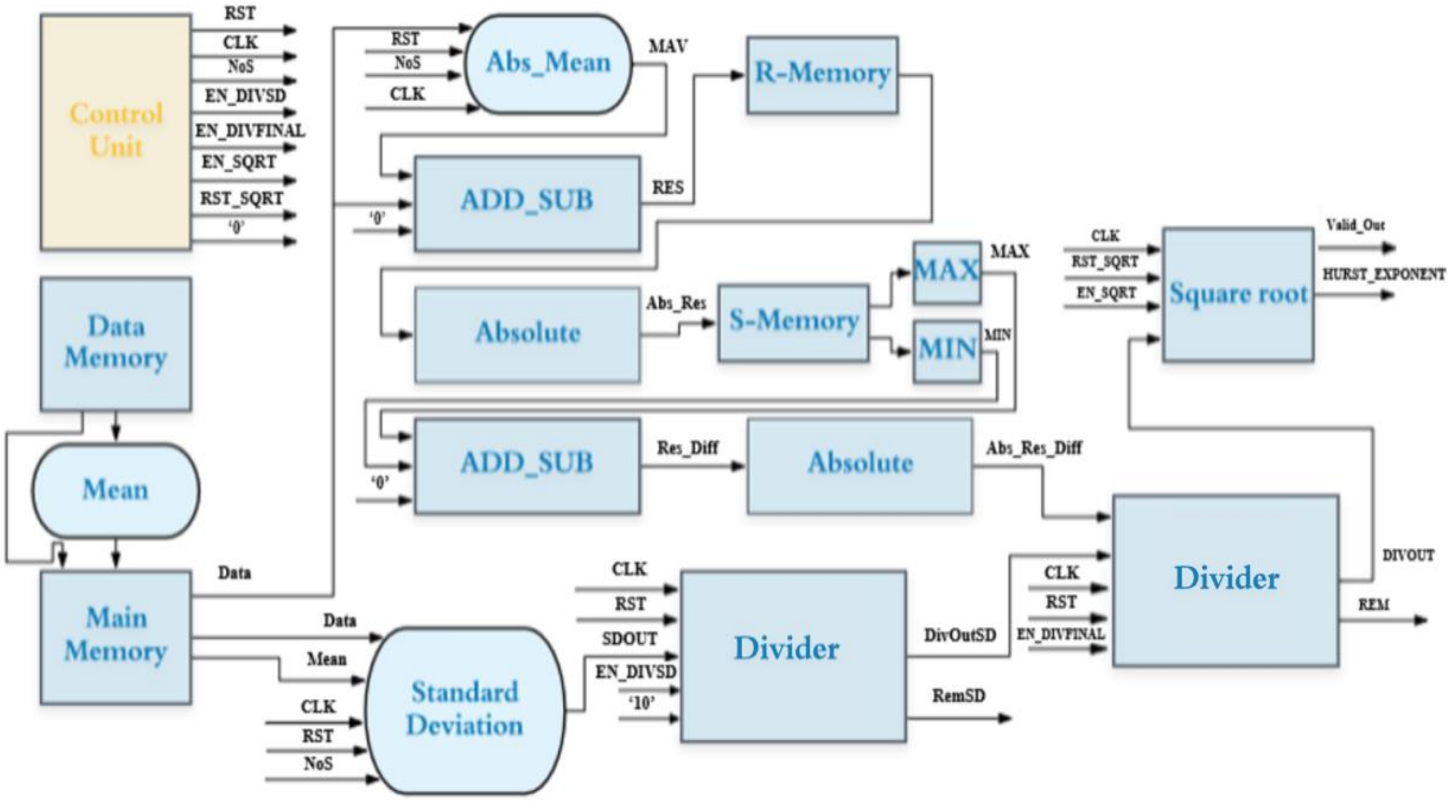


Figure 5-6 RTL Hurst exponent block diagram

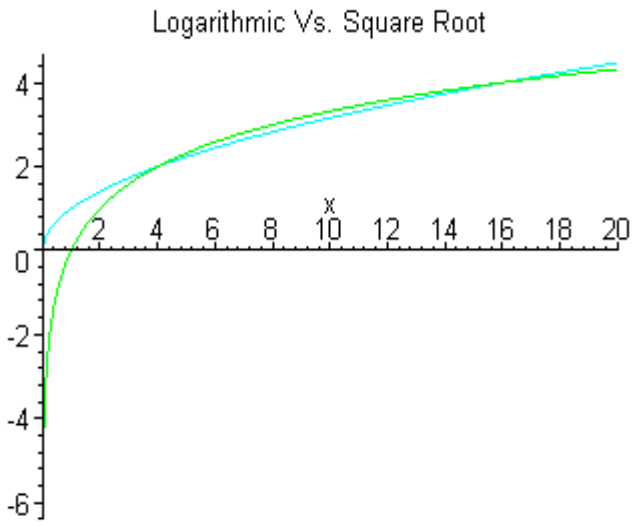


Figure 5-7 Logarithmic VS. Square root relation (Green is log while cyan is square root).[41]

The Hurst exponent is one of the most powerful non-linear features as we explained before in the previous chapter. When referring to its hardware RTL implementation, the flow of the block diagram is as follows; the control unit here is the most important block since it's responsible for all of the enables of the dividers and the square root blocks and it is also providing the controller bit of the "Add_Sub" block to control if it is summing or subtracting, it also supplies the clock and the number of samples, or in other words the window size of the samples. The block diagram's "Main Memory" is providing the EEG data samples and their mean value to the absolute mean and the standard deviation blocks that were explained before in the previous sections. The standard deviation block here calculates the value of the standard deviation of the EEG data samples, this value enters to a divider block waiting for the value to be calculated from the absolute mean block's output, this output passes through an "Add_Sub" block with a controller bit '0' in order to subtract the mean absolute value (MAV) from each EEG data sample before saving these values in the R-memory in a form of an array, the values in the R-memory enter to an absolute block for saving the same values but in an absolute manner in the S-memory, from this memory we calculate the maximum deviation from mean(MAX) and the minimum deviation from mean(MIN), these two values are subtracted from each other according to an "Add_Sub" block with a controller bit '0' and we take the absolute of this subtraction result using an absolute block, this final result is characterized in variable **R**, this final value **R** is then divided by the standard deviation value **S** of the EEG data sample that was already calculated through a divider block, there is a standard deviation "Enable_Divide" bit provided by the controller here and it is automated to be set as '1' when the two values "**R** and **S** " are calculated so that it divides these two values not any other values. At this point, we faced a challenge while implementing this feature; when referring to the previous chapter we figured out that the Hurst exponent is calculated according to this formula **ln(R/S),** until now we calculated the **R/S,** and to finalize the feature we had to apply the logarithmic function **ln** to this formula, but it was complex to implement the logarithmic function on a hardware simulation tool and it was also hard searching for it online. We kept looking for other solutions for implementing this **ln** function until we found a way for doing this

using the look up tables (LUTs); by mapping values according to the expected outcomes of the **ln** function, but it also appeared to be somehow complex in implementing and it could have consumed a lot of power, so we had to change our mindset to find an alternative for this problem till we tried figuring out the relation between the logarithmic function and the square root function that appeared to be very satisfying regarding our case in implementing the Hurst exponent feature. We can figure out from the previous figure the relation between the square root of a value and the logarithmic of the same value is almost the same specially in the higher numbers. In our case the R/S is always less than 1 (<1), so we first had to modify our relation (**R/S)** to be able to use this relation by passing the standard deviation value **S** by a divider with a value of **'10'** so that the value of **R/S** is always higher than 1 for applying the Square-Root block instead of using the logarithmic relation. We can now take the value **R** with the modified standard deviation **S** to Square-Root block which has an enable bit, this bit is switched to '1' when the modified **R/S** is ready to enable the block to start and calculate the square root of the **R/S**, the "Reset SQRT" bit here is responsible for the number of cycles that this Square-Root block will take to get its value according to the number of bits of the EEG data samples, in this case the Hurst exponent is now considered calculated and it also turns up a validation bit as an indication that the value of the Hurst exponent value is calculated.

5.1.1.6 RTL Mean

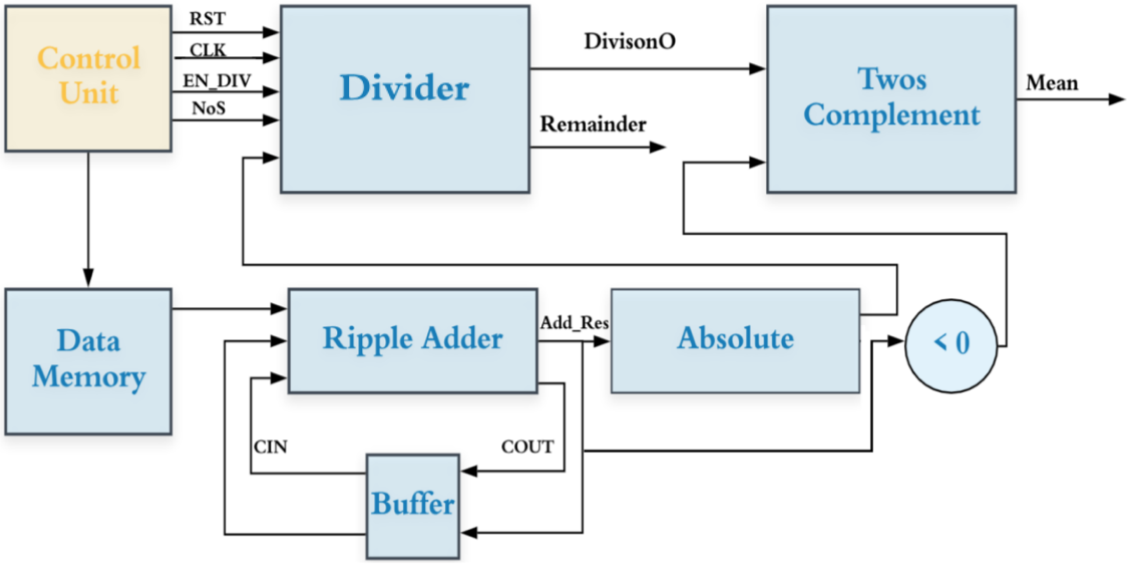


Figure 5-8 RTL mean block diagram

The mean feature is used to conduct the mean of the data samples to be saved along with the data itself in the main memory to be used by the system features. In order to conduct the mean, the data samples are added together through the ripple adder then the result of the addition should be divided by the number of samples. The problem showed up as our divider module takes only positive values and the output of the ripple adder might be negative value as the data samples vary between positive and negative values. To solve this problem, the absolute of the output of the ripple adder is conducted and is then fed as an input to the divider which needs a number of signals from the control unit as discussed in 5.1.1.2. At the same time ripple adder output is checked whether if it is a positive or a negative value. If it is a positive value, then the divider value is the mean and the two's complement block takes a '0' value so that it would act as a buffer. Else, if the ripple adder value is negative then the output of the divider is not the mean and the two's compliment of the divider output must be done. That's why the comparator will be sending '1' signal to the two's compliment in order to operate and yield the mean.

5.1.2 Classifiers Blocks

5.1.2.1 Linear Classifier

The support vector machine (SVM) binary classifier is responsible for classifying the EEG testing data after being feature extracted into one of the following two classes, either the pre-ictal state class represented in bit '1' or a non pre-ictal state class represented in bit '0'. The linear support vector machine classifier we implemented is divided into several modules defined as follows:

1. <u>Rom Module:</u>

We used three Rom modules here for filling the data corresponding to the **SVMTRAIN Struct** in the Matlab (Support vectors, Alphas and the vector of the input features), the sizes of these Roms are determined on the basis of the window size.

2. <u>Multiplier module:</u>

The multiplier module is used for multiplying two corresponding data samples.

3. <u>Adder module:</u>

The adder module is used for adding two corresponding data samples.

4. Inner module:

The inner module uses the previous two modules (Multiplier and Adder modules); for multiplying multiple samples and adding them in a respective manner.

5. Top Classifier module:

The top classifier module is considered the core module in this binary linear classifier, it first passes the support vector samples, testing data sample and the alpha samples to the classifier module, the classifier multiplies this testing data sample with all the support vector samples and adds them respectively using the inner module, the output result of the inner module is then used as an input of the adder module with a bit '1', this result is then passed again to the top classifier module with the "alpha_y" register that is calculated using the alpha samples and the vector of the input features, there is another inner module in the top classifier that is used to determine the number of testing data that we want to classify as an indication for the window size, and finally after applying these modules we can figure out our output classifier bit according to a "valid out" bit that is switched to '1' as an indication that the classifier bit is calculated, the final classifier bit's value can be either '0' or '1' according to the case; if it is a pre-seizure case or not.

5.1.2.2 Non-Linear RBF Classifier

The support vector machine (SVM) binary classifier is responsible for classifying the EEG testing data after being feature extracted into one of the following two classes, either the pre-ictal state class represented in bit '1' or a non pre-ictal state class represented in bit '0'. As previously mentioned in Chapter 3, the linear, polynomial and RBF or Gaussian kernels are simply different in the case of making the hyperplane decision boundary between the classes. The kernel functions are used to map the original dataset (linear/nonlinear) into a higher dimensional space. Usually, the linear and polynomial kernels are less time consuming but they provide less accuracy than the RBF or Gaussian kernels. The RBF support vector machine classifier we implemented is divided into several modules defined as follows:

1. Alpha_Mem Module:

We used this Memory module for filling the data corresponding to the Alphas from the **SVMTRAIN Struct** in the Matlab, the size of this Memory is determined based on the window size.

2. Kernel_Mem Module:

We use this module for filling the data corresponding to the EEG testing data samples in the Matlab but in different concatenated ways.

3. Y_Mem Module:

We used this Memory module for filling the data corresponding to the Seizure True Train labels from the **SVMTRAIN Struct** in the Matlab, the sizes of this Memory is determined based on the window size.

4. Expx Module:

We used this module to model the exponential, to be used in the calculation of the RBF kernel on 2 samples x and x'.

$$K(x, x') = \exp(-\frac{||x - x'||^2}{2\sigma^2})$$

$||x - x'||^2$ *may be recognized as the squared Euclidean distance between the two feature vectors.*

σ *is a free parameter.*

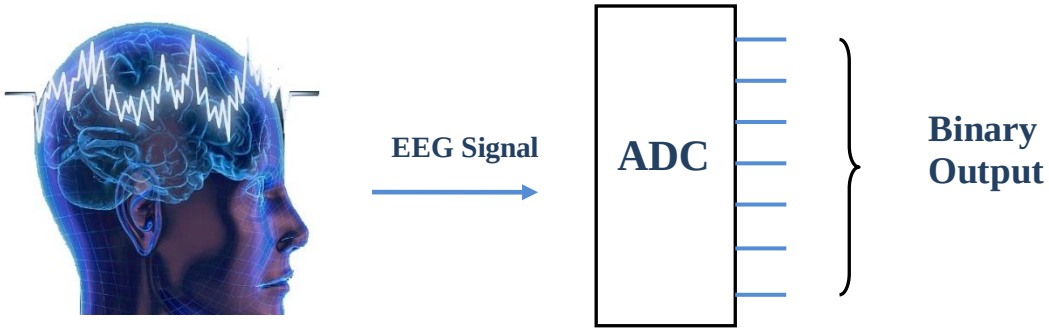
5. Para_Adder_Subtractor Module:

This module can act either as an adder or a subtractor.

6. Classy_Top Module:

The **classy_top** module is considered the core module in this binary RBF classifier, it first passes the testing data samples, the alpha samples, the kernel samples and the bias to the classy module, the **classy** module calls the **expx** module and passes the **expx** result to the **para_adder_subtractor** together with the bias to add them together, and then "done_exp_f" turns '1'. The summation becomes the new bias and the procedure is repeated on all window samples and when finished the "done_f" turns '1' to allow the **classy** module to produce the output.

5.2 Bits Reduction



As we all know, EEG signal values are all floating point values for sure, and as we moved to the RTL phase, we were left with 2 options, either to deal with floating points or convert them to fixed points. But, before moving on to which format is better; let's discuss the desired features in any design.

List of Desired Features:

1. Large dynamic range, which is the largest number that can be represented.
2. Small data path size. Routing is always a major concern in ASICs and FPGAs, so computational types must keep the data path width to a reasonable size at the expense of loss of precision.
3. Numeric precision. The calculations need to remain "essentially correct" after going through many rounding/truncation operations.
4. Small fixed latency.
5. Low complexity.
6. Reasonable implementation cost.

Mentioning this, we need to accurately highlight our system requirements and criteria to ensure certainty on whether dealing with fixed-point or floating-point is more ideally suited for our purpose.

Table 5-1 Floating-Point vs Fixed-Point

	Floating Point	**Fixed Point**
Dynamic Range	Large	Small
Data path size	Large	Small
Numeric Precision	High	Low
Latency	More	Less
Complexity	Very High • Floating-point support in an FPGA often uses more than 100 times as many gates compared to fixed-point support. • Floating-point takes up almost 3X the hardware of fixed-point math.	Low
Implementation Cost	Expensive	Cheap
Power Consumption	Very High	Relatively Low

Using the data in the above table, we can conclude that Floating-Point math is very complex, needs a lot of hardware resources and consumes a lot of power. Despite the fact that it has many desirable features, but our main target is low power consumption and small area so we will be more biased towards the Fixed-Point format.

First of all, converting the floating points to fixed points was easily implemented in MatLab, and then we converted the fixed-point decimal values to binary values in order to export it to HDL.

Moving to the RTL, we figured out about some of vhdl new “sfixed” and “ufixed” packages to deal with the signed and unsigned fixed point numbers but these packages weren’t so popular and we couldn’t find that much of resources to help us work with them. So we took a step backwards to make a workaround in MatLab that we believed would help ease a lot of our upcoming work if it worked as expected.

We multiplied our data by a constant factor, and surprisingly the results were almost still the same, so we decided on how many decimal figures we would like to keep and chose our constant factor accordingly. As a matter of fact, our dynamic range increased as well as the needed number of bits. We needed 32 bits as a consequence which was relatively a

huge number. Needless to say, this choice increased the overall system's power, the depth of the needed memory and the chip's area. Thus, we exploited the fact that the power decreases by almost one quarter as the number of bits decrease by half. We knew for sure that truncating some of the decimal figures would decrease the power bearing in mind that it would also degrade our performance results, so there must have been a tradeoff, but it was worth the shot.

We then tried to reduce the number of bits to 16, meaning that we had to truncate almost all the decimal points for the sake of the system's power. Consequently, we needed to check the traded degraded system's performance. However, luckily, the Sensitivity was almost still the same and the Accuracy had only undergone a negligible decrease. Moreover, the overall power decreased significantly as a result. Therefore, we decided that we were going to work with this arrangement.

5.3 FPGA

5.3.1 What is FPGA?

FPGA stands for Field Programmable Gate Array. It is an integrated circuit which can be "field" programmed to work as per the intended design. As implied by the name itself, the FPGA is field programmable. So, an FPGA working as a microprocessor can be reprogrammed to function as the graphics card in the field, as opposed to in the semiconductor foundries. The designs running on FPGAs are generally created using hardware description languages such as VHDL and Verilog.[28]

FPGA is made up of thousands of Configurable Logic Blocks (CLBs) embedded in an ocean of programmable interconnects. The CLBs are primarily made of Look-Up Tables (LUTs), Multiplexers and Flip-Flops. They can implement complex logic functions. Apart from CLBs, and routing interconnects.[28]

While the FPGA is used in this project but it is important to mention another class of a -somehow- a similar device which is ASIC. ASIC stands for Application Specific Integrated Circuit. As the name implies, ASICs are application specific. They are designed for one sole purpose and they function the same their whole operating life.

5.3.2 FPGA Vs ASIC

FPGA can be reconfigured with a different design. It even has the capability to reconfigure, ASIC is permanent circuitry. Once the application specific circuit is taped out into silicon, it cannot be changed. FPGA is preferred for prototyping and validating a design or concept. Many ASICs are prototyped using FPGAs themselves. FPGA is easier to make sure design is working correctly as intended using FPGA prototyping. It's here the major advantage of FPGA over ASIC can be found.

After the researching phase, the ASIC would come to its role to be better for its lower power, lower unit costs and faster than FPGA with higher performance.

5.3.3 The function of FPGA in the design

In this project, the need for FPGA is very critical to boosting the performance of our design. As the FPGA can perform the SVM training on the data much faster than the CPU. This is due to the dividable nature of the algorithm used. The capability of the FPGA to provide a high level of parallelism is employed to solve the training problem in parallel chunks yielding much faster training. The FPGA gives a great advantage as it supports fast memory access.

In the development phase, assigning the training function to the FPGA off-loads it from the CPU. Leaving the CPU to process performance measurements simultaneously using the updated parameters that the FPGA provides. This could be done by passing the continuously updated training parameters from the FPGA to the CPU, allowing for hardware acceleration.

FPGA is Preferred for prototyping and validating a design or concept. This is employed in the algorithms methods that we have for further future expansion. (That is; to store all the training algorithms on the chip and reconfigure to the currently required as explained previously). So, it's more efficient to consider this vision in the current development phase.

5.3.4 FPGA in use

Figure 5-9 FPGA Kit

The FPGA device used in this project was provided by Altera - an Intel business unit. **Xilinx Zynq-7000 SOC ZC702 Evaluation Kit.**

Zynq-7000 SOC ZC702 Evaluation Board Features:

Table 5-2 Zynq-7000 SOC ZC702 Features.[34]

Logic Cells (K)	85
Block RAM (Mb)	4.9
DSP Slices	220
Maximum I/O Pins	200

The total available lookup tables are 53200 slices, and the total available registers are 106400 slices.

5.3.5 FPGA Design Tools

Hardware description languages (HDLs) such as VHDL and Verilog evolved into the primary languages for designing the algorithms running on the FPGA chip. Once the FPGA design has been created using HDL, the compilation tool takes the text-based logic and, through several complex steps, synthesizes the HDL down into a configuration file or bit stream that contains information on how the components should be wired together. As part of this multistep manual process, then mapping signal names to the pins on the FPGA chip that used.[28]

5.3.6 High-Level Synthesis Design Tools

Vivado programming environment is distinctly suited for FPGA programming because it clearly represents parallelism and data flow, In addition, Vivado can be used to integrate existing VHDL into your Vivado FPGA designs. Because Vivado FPGA is highly integrated with hardware, there is no need to rewrite code in VHDL to meet timing or resource constraints, as may be the case in many HLS code generators. Figure 5.10 shows us the flow of the project which ends by applying synthetization to the HDL code using Vivado. Vivado enables developers to synthesize their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.[28]

Figure 5-10 Project's different phases

5.3.7 Access the FPGA

It is noteworthy to mention that **ONE LAB** considering obtaining the access of the FPGA using its server which facilitates dealing with the FPGA. Figure 5.11 shows the website that helps us to access the FPGA by reserving an empty slot.

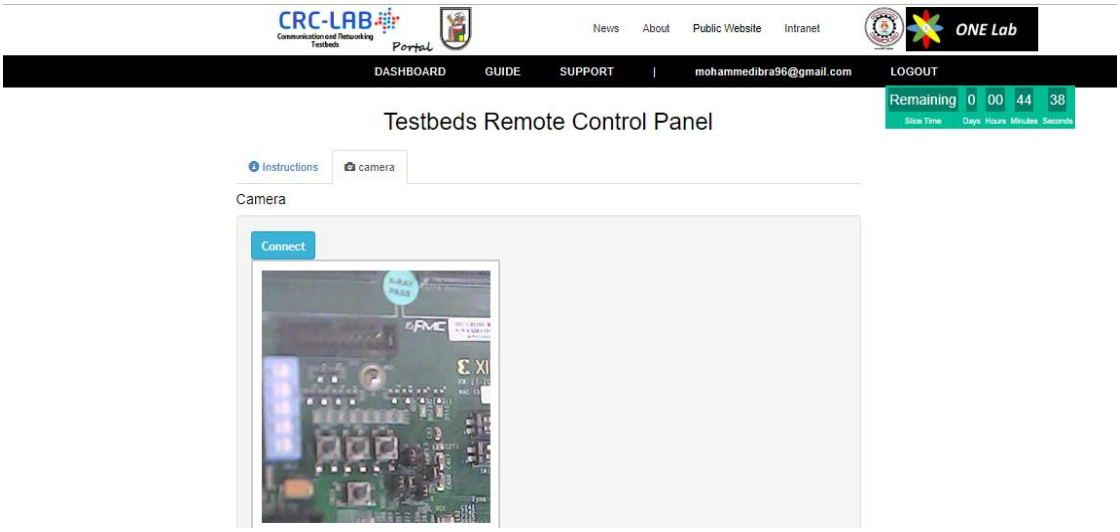


Figure 5-11 Accessing the FPGA through CRC-LAB website

5.3.8 Vivado's Power Analysis

The Vivado power analysis feature performs power estimation through all stages of the flow: after synthesis, after placement, and after routing, as shown in the 5.13. It is most accurate post-route since it can read from the implemented design database the exact logic and routing resources used. Figure 5.12 presents the summary power report and the different views of your design that can be navigated: by clock domain, by type of resource, and by design hierarchy. Vivado Integrated Design Environment (IDE) support adjust environment settings and design activity so it can reduce the design supply and thermal power consumption. It supports cross-probe into the design from the power report, which aids in identifying and evaluating high power consuming hierarchy/resources used in the design.[29]

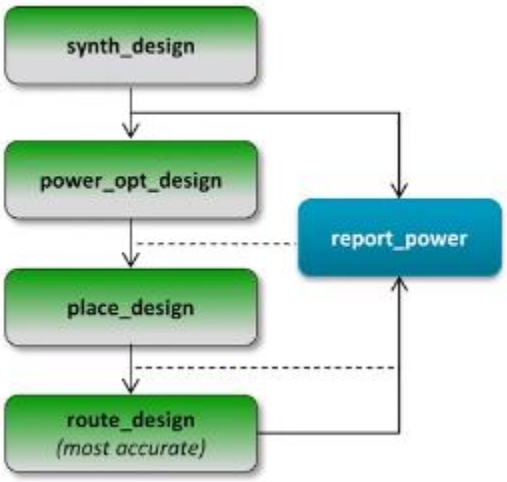


Figure 5-13 Reporting power after every stage

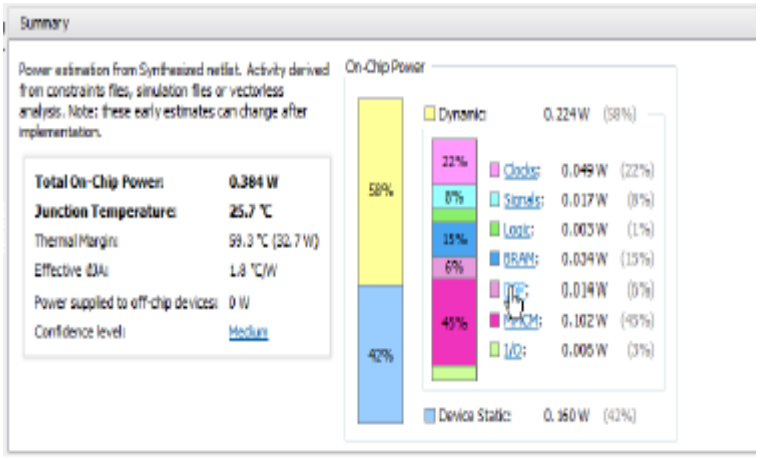


Figure 5-12 Power report summary

5.3.9 Synthesis

Synthesis transforms HDL code into a *netlist* describing the hardware (e.g., the logic gatesand the wires connecting them). The logic synthesizer might perform optimizations to reduce the amount of hardware required. The netlist may be a text file, or it may be drawn as a schematic to help visualize the circuit.

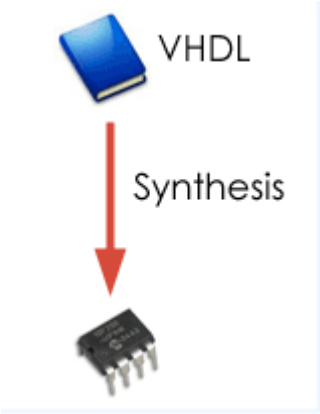


Figure 5-14 VHDL synthesis

5.3.10 Types of Power

5.3.10.1 Device Static Power

The power from transistor leakage on all connected voltage rails and the circuits required for the FPGA to operate normally post configuration. Device static power is a function of process, voltage, and temperature. This represents the steady state, intrinsic leakage in the device.[29]

5.3.10.2 Design Power:

The power of the user design, due to the input data pattern and the design internal activity. This power is instantaneous and varies at each clock cycle. It depends on voltage levels and logic and routing resources used. This also includes static current from I/O terminations, clock managers, and other circuits which need power when used. It does not include power supplied to off-chip devices.[29]

5.3.10.3 Total On-Chip Power:

The power consumed internally within the FPGA, equal to the sum of Device Static Power and Design Power. It is also known as Thermal Power.[29]

5.3.11 Power calculation

There are two modes to calculate the power using Vivado

1) Vector less mode

This mode is a simple mode for power analysis it is considered a tool to calculate the probability of switch rate. It is not accurate compared to the vector based.

2) Vector-based (SAIF/VCD)

This mode is run using a SAIF/VCD file generated from behavioral simulation for Vivado Report Power Analysis that will enable to do more accurate power estimation on our design. We apply this mode to calculate accurate power

5.3.12 Area Estimation

One of the most important constraints required by the FPGA hardware designer is that a design fits inside the specified FPGA. This requires an estimate of the number of CLBs (Configurable Logic Blocks) required by the design. This is because the FPGA has fixed logic as well as fixed routing resources. A simple estimation of the FPGA CLB area from a Register Transfer Level (RTL) description of the hardware based upon the number of lines of VHDL code would be grossly inaccurate because a single line of RTL code can create a wide complex multiplier while another line might require just a register to hold the result. Since, both the Control Logic and the Data path of any design are implemented as lookup tables inside the CLBs, an accurate prediction of the number of CLBs required would need a count of the total number of hardware resources required by the design for the control logic and the data path.in the figure 5.15 shows part of the utilization report.[30]

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	190	0	53200	0.36
LUT as Logic	190	0	53200	0.36
LUT as Memory	0	0	17400	0.00
Slice Registers	387	0	106400	0.36
Register as Flip Flop	387	0	106400	0.36
Register as Latch	0	0	106400	0.00
F7 Muxes	0	0	26600	0.00
F8 Muxes	0	0	13300	0.00

Figure 5-15 Utilization report

5.3.13 Delay Estimation

Most high-level synthesis tools perform rapid design exploration and output a design which meets the user-specified attributes which normally include certain area and performance constraints. To achieve rapid design closure, such tools should not only have

rapid area estimators as outlined in the previous section but also, delay estimators to determine whether the synthesized design would meet the frequency constraints.in the figure 5.16 shows the timing summary which contains the multiple paths delays.

Name	Slack ^1	Levels	High Fanout	From	To	Total Delay	Logic Delay
Path 1	27.637	5	55	dbg_hub/inst/BS.../state_reg[0]/C	dbg_hub/inst/B...temp_reg[0]/D	5.405	1.518
Path 2	27.644	5	55	dbg_hub/inst/BS.../state_reg[0]/C	dbg_hub/inst/B...temp_reg[1]/D	5.402	1.518
Path 3	27.661	5	55	dbg_hub/inst/BS.../state_reg[0]/C	dbg_hub/inst/B..._temp_reg[1]/D	5.306	1.518
Path 4	27.666	5	55	dbg_hub/inst/BS.../state_reg[0]/C	dbg_hub/inst/B..._temp_reg[2]/D	5.303	1.518
Path 5	27.710	5	55	dbg_hub/inst/BS.../state_reg[0]/C	dbg_hub/inst/B..._temp_reg[0]/D	5.259	1.518
Path 6	27.714	5	55	dbg_hub/inst/BS.../state_reg[0]/C	dbg_hub/inst/B... temp_reg[4]/D	5.254	1.518

Figure 5-16 Timing summary containing multiple paths delays

6

RESULTS, CONCLUSION& ACHIEVEMENTS

6.1 Results

The project was mainly divided into two main phases: software simulation and hardware simulation. Each of the two phases yielded results from which we made decisions accordingly.

6.1.1 Software simulation results

Using Matlab simulation tool, we simulated most the possible features along with the linear kernel and a nonlinear kernel (RBF). We used CHB-MIT Scalp's dataset and we chose to work with the EEG data of several -and not all- patients only due to the limitations of the recorded data and their complex feature space on having nearly fully overlapping classes. We worked with patients: 7, 8, 11, 19, 20 & 23. MIT's dataset didn't mark the pre-ictal interval in the patients' files so we simulated different pre-ictal intervals along with linear features: Coastline, Energy, Relative spectral power, Hjorth, Standard deviation, Spectral centroid, Skewness, Kurtosis, Spectral skewness, and Variation Coefficient. As shown in Table 6.1, the best performance was obtained from taking a 30-minute pre-ictal period. Although accuracy and specificity decreased, sensitivity is higher and it is the most crucial for seizure prediction

Table 6-1 Performance results versus different pre-ictal intervals

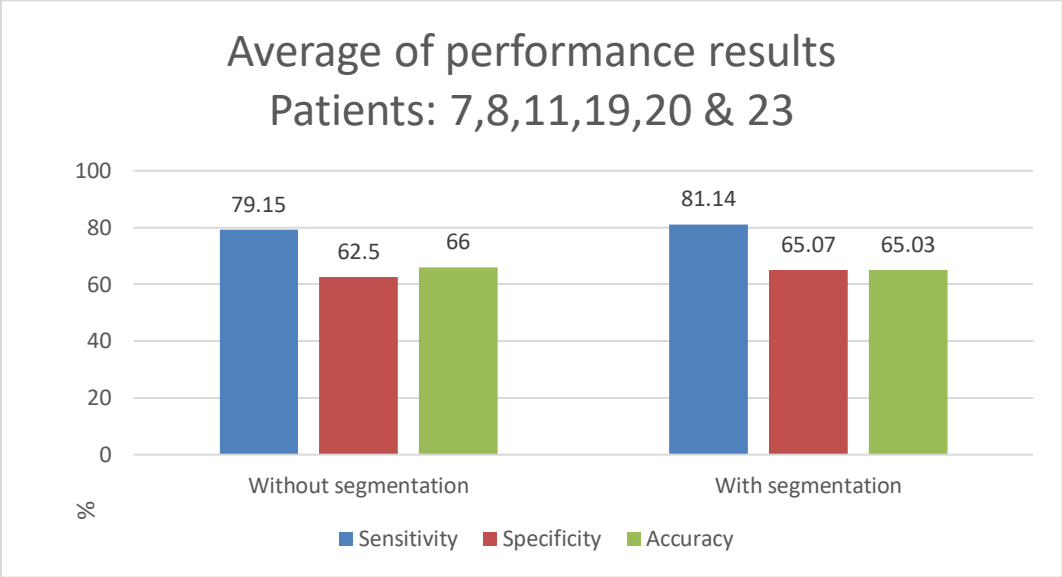
	15-minute pre-ictal	**20-minute pre-ictal**	**30-minute pre-ictal**
Accuracy	74.90%	69.9375%	68.8875%
Specificity	77.375%	71.8825%	73.025%
Sensitivity	41.675%	47.81%	**50.625%**

Since we took the 30-minute interval preceding the marked beginning of the ictal period as the pre-ictal period. We simulated several linear and non-linear features along with RBF and linear kernels. The results obtained of the different combinations are present in table 4.1. Some results yielded very low performance, so they were excluded. From these simulation results, we decided to work on two different combinations only, the highest performance of all the combinations:

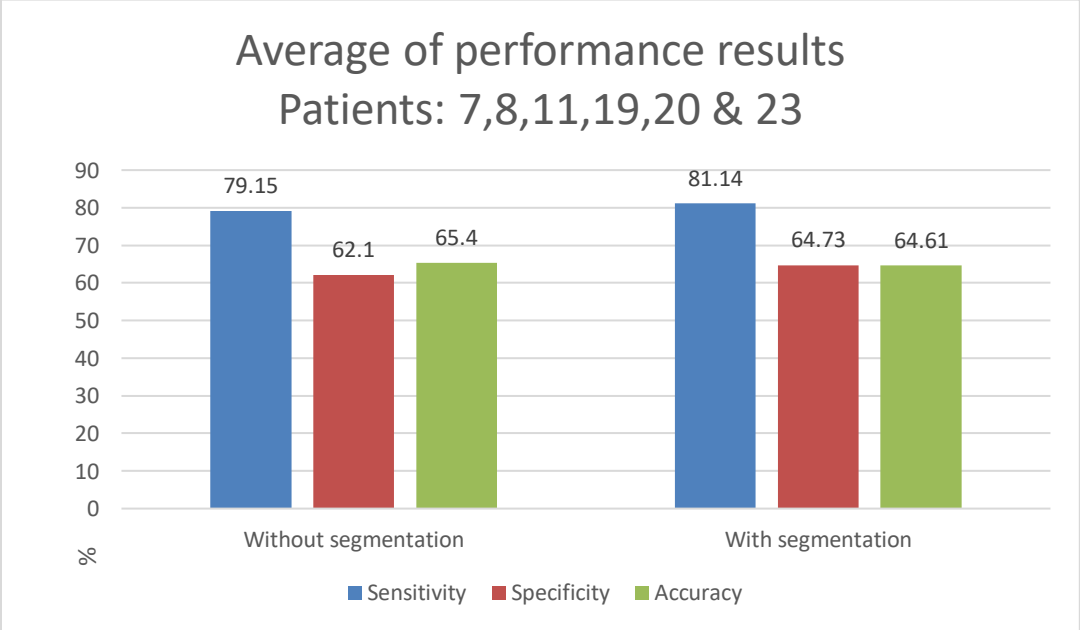
1. Linear feature-kernel combination: Four linear features with a linear kernel: Coastline, Root Mean Square, Absolute Mean, and Standard Deviation with linear kernel.

2. Nonlinear feature-kernel combination: One non-linear feature with the non-linear kernel: Hurst feature with RBF kernel.

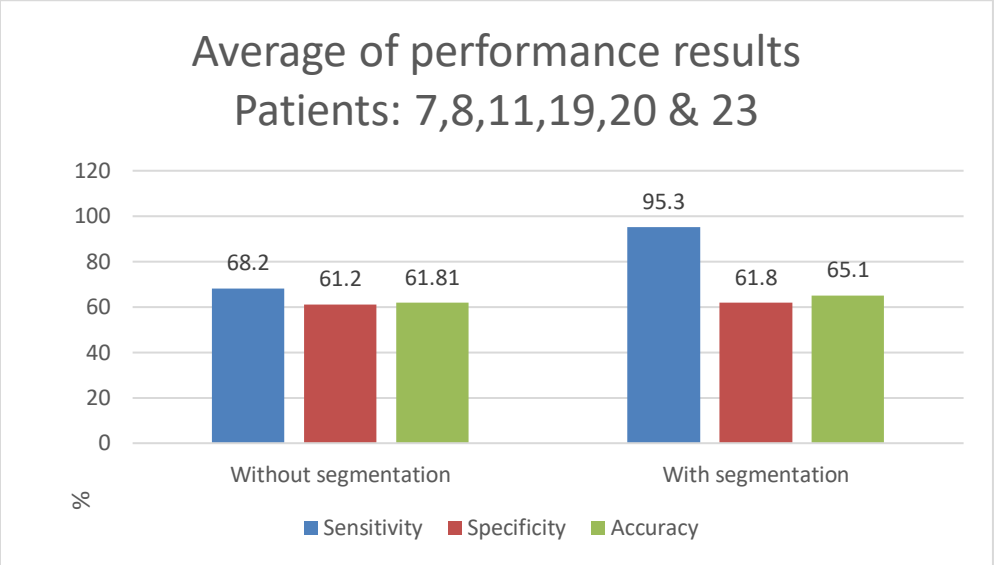
6.1.1.1 Case: Linear kernel without segmentation VS with segmentation



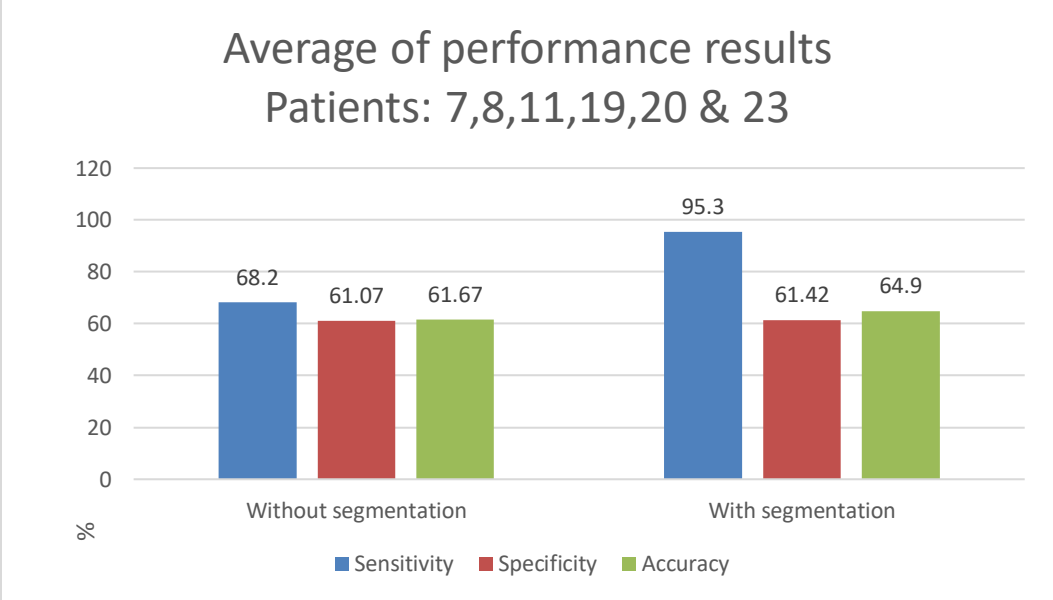
6.1.1.2 Case: Linear kernel without segmentation VS with segmentation (bits reduced to 16-bits)



6.1.1.3 Case: RBF kernel without segmentation VS with segmentation



6.1.1.4 Case: RBF kernel without segmentation Vs. with segmentation (bits reduced to 16-bits)



6.1.1.5 Patient 8's analysis

We can check our performance results in its different cases on **Patient-8** separately, since it's EEG data records are considered very clear and not complex at all having a clear seizure period and a linearly separable feature space –on another context both classes are nearly not overlapping- as show in the following two figures:

Figure 6-1 Patient 8's recorded EEG signal.[42]

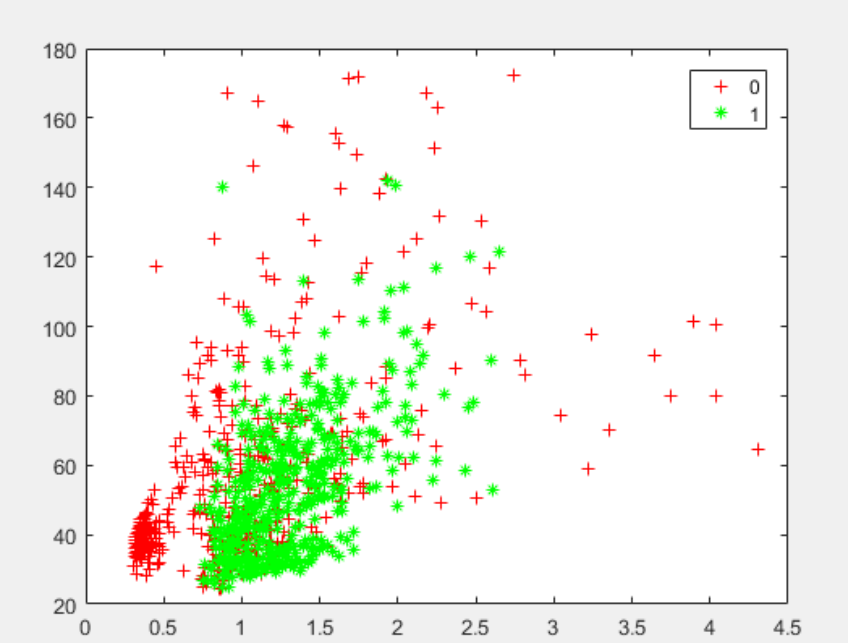
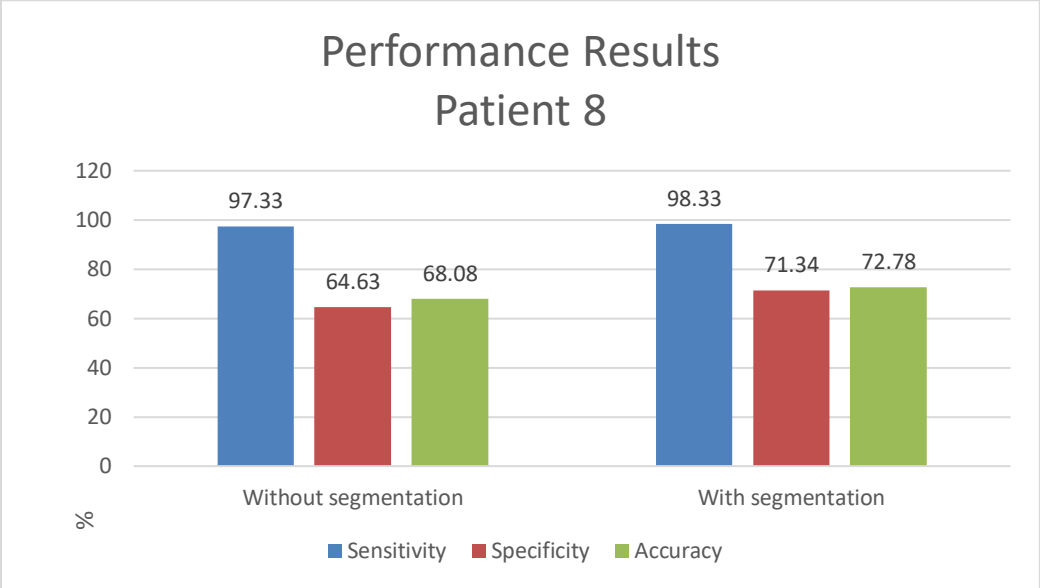
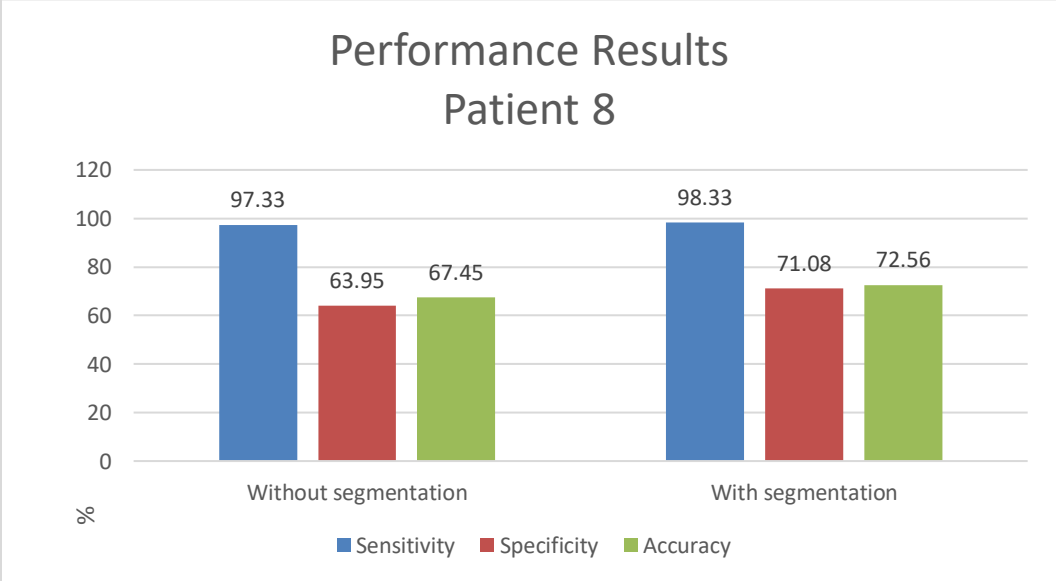


Figure 6-2 Patient 8's feature space

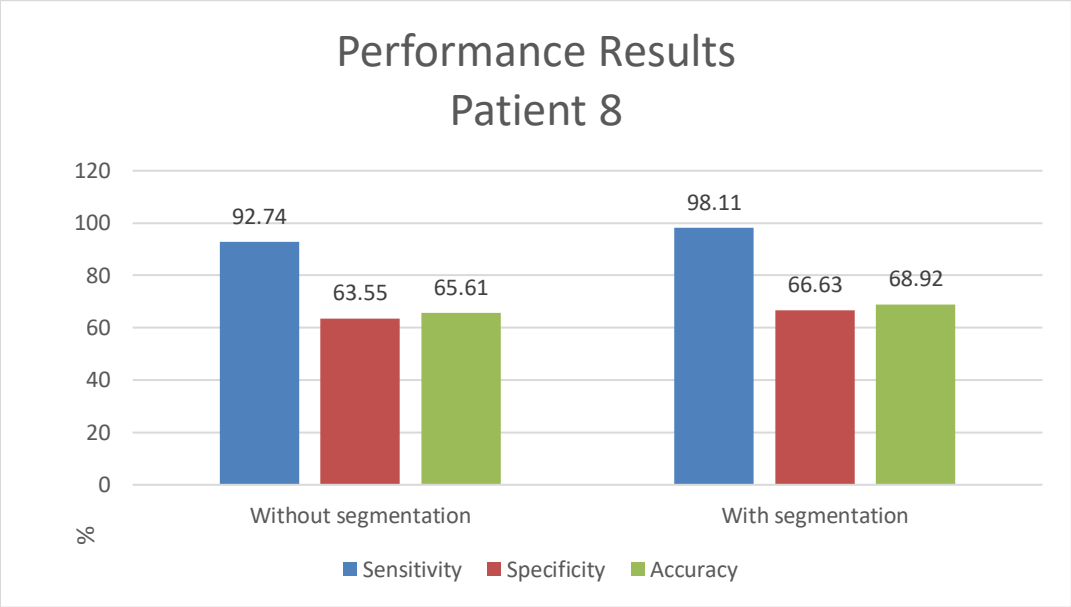
6.1.1.6 Case: Patient 8's linear kernel without segmentation VS with segmentation



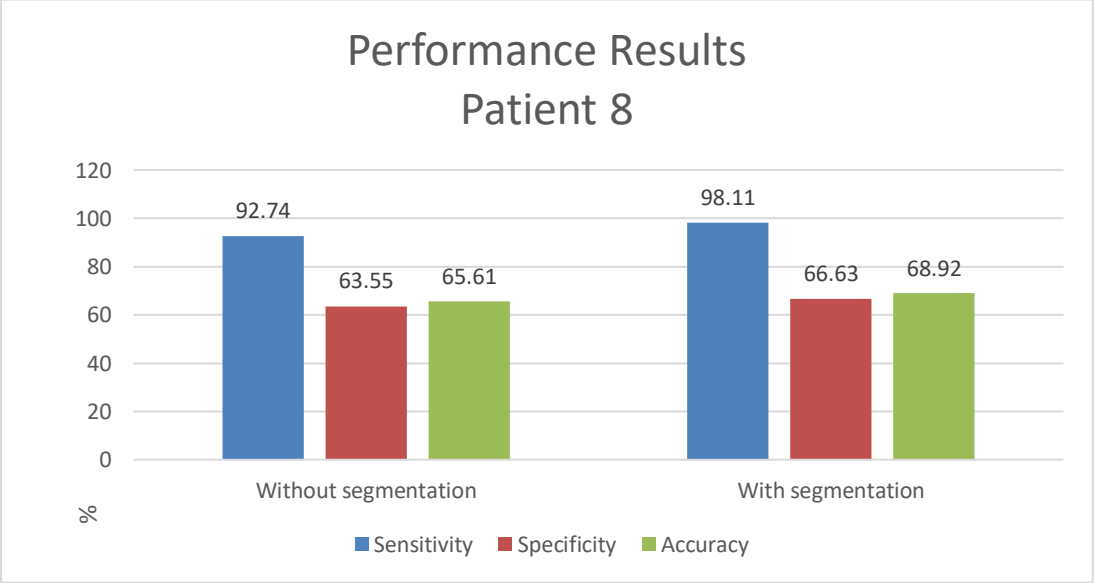
6.1.1.7 Case: Patient 8's linear kernel without segmentation VS with segmentation (bits reduced to 16-bits)



6.1.1.8 Case: Patient 8's RBF kernel without segmentation VS with segmentation



6.1.1.9 Case: Patient 8's RBF kernel without segmentation VS with segmentation (bits reduced to 16-bits)



6.1.1.10 Discussion and Analysis of the software results

We can figure out from the previous figures that segmenting the sampled EEG data and using majority rule gives a higher overall performance as discussed in section 4.1.6. Simulation of different segmentation intervals for each of the two feature-kernel combinations resulted in two different segmentation intervals for each of the combinations. The linear feature-kernel combination would have the highest performance with 89 segments, while the nonlinear combination would have the highest performance when implemented with 5 segments.

We can also conclude as discussed before in section 4.1.6 that reducing the number of bits is considered a tradeoff between the dynamic power consumption and the performance results; as the dynamic power consumptions significantly decreases, while the performance results decreases in a negligible way on the basis of the specificity and the accuracy only, therefore reducing the number of bits' in either the cases of segmenting the testing data or not segmenting it is considered necessary in our case for saving power.

6.2 Hardware simulation results

In this section, we will discuss the results obtained from Vivado on terms of the most important aspects for implementing our algorithm to the FPGA kit; the total power, area, and the total delay. Table number 6.2 shows the results obtained from the linear features; standard deviation, coastline, root mean square and absolute mean with the linear kernel, while table number 6.3 shows the results obtained from the non-linear Hurst-exponent feature with the non-linear (RBF) kernel.

Note that Vivados's minimum measureable value for power is 1mw, while the FPGA-Kit we are using (Zynq-7000 XC7Z020 SoC) have the following specifications:

- Total Available LUTs = 53200 Slice.
- Total Available Registers = 106400 Slice.

Table 6-2 Linear features with linear kernel hardware results

Hardware Performance	Linear Features (Standard Deviation, Coastline, Root Mean Square, Absolute Mean)	Linear Kernel
1-Total Power (Dynamic power)	6mw	< 1mw
2-Area **A-Slice LUTs** **Utilization %** **B-Slice Register** **Utilization%**	 4606 slice 8.66% 2941 slice 2.76%	 567 slice 1.07% 959 slice 0.9%
3-Total Delay	5.762 ns	5.459 ns

Table 6-3 Non-Linear feature with RBF kernel hardware results

Hardware Performance	Hurst Feature	RBF Kernel
1-Total power (Dynamic power)	6mw	3mw
2-Area **A-Slice LUTs** **Utilization %** **B-Slice Register** **Utilization%**	 1745 slice 3.28% 1917 slice 1.8%	 1323 slice 2.49% 1293 slice 1.22%
3-Total Delay	4.62 ns	5.405 ns

6.2.1 Discussion and Analysis of the hardware results

By comparing between the two aforementioned feature-kernel chosen combinations in terms of hardware applicability. A tradeoff between the two possible combinations was clearly presented. The power in the linear features with linear kernel is 27.78% lower than that of the nonlinear feature with the nonlinear kernel (RBF). On the other hand, the area of the nonlinear feature and kernel is smaller than that of the linear features-kernel combination since the linear features-kernel combination has 68.6% more LUTs and 21.49% more registers than that of the nonlinear feature and kernel.

We can conclude that if the battery-life is our focus, implementing the linear track would be better, however if the area of the chip was of higher importance, we would need to implement the nonlinear track algorithm.

6.3 Comparison Between Our Proposed Models and Other Models

Table 6-4 Proposed models vs. other models

Model	Zewail	Zewail	CUFE	Hindawi	Proposed	Proposed
Data set	CHB-MIT	CHB-MIT	CHB-MIT	CHB-MIT	CHB-MIT	CHB-MIT
Detection/Prediction	Detection	Detection	Detection	Prediction	Prediction	Prediction
Kernel	Linear	RBF	Linear	Linear	Linear	RBF
Sensitivity	61.81%	62.14%	88%	92.23%	81.14%	95.3%
Specificity	96.89%	99.93%	87%	-	64.73%	61.42%
Accuracy	96.81 %	99.85 %	89.3%	-	64.61%	64.9%
Power	40 mw	40 mw	156.13mw	-	7 mw	9 mw
Delay	9.29 sec	3.31 s		-	11.221 ns	10.025 ns
Number of registers			1183		3900	3210

Table 6.4 shows a comparison between our proposed models and other models in terms of software performance, hardware aspects and kernel type. Zewail's models and CUFE model are used in seizure detection using support vector machine (SVM) to classify between seizures and non- seizures classes while our proposed models and Hindawi's model are used in seizure prediction which is a binary classification between pre-ictal and non-pre-ictal periods.

In Zewail's models and CUFE model, the Specificity and accuracy are around 30% higher than the proposed models. However, in the proposed models the sensitivity is around 20% to 30% higher than Zewail's models and CUFE model. Concerning the hardware aspects there is huge difference between the proposed models and the other models in terms of the power and the total delay, the proposed models have lower power consumption and lower delay.

The proposed models support the patient to use it in an adaptive way, the implantable chip can be used in the low power mode to extend the battery's life which will get activated when the battery is almost empty. Low power mode reduces the processing by decreasing the number of segments of testing data that will affect the performance in negligible way.

6.4 Conclusion

About 1% of the global population are epileptic patients with an estimate of their third are not respondent to medicine. [2] With the low success rate of traditional treatment methods for epilepsy and high economic demand for a viable treatment option. Data processing for seizure prediction using the seizure prediction algorithms along with minimal hardware design to conserve battery use is presented in this work as a feasible option.

The project was divided into two main phases. The first is software simulation with Matlab tool and the second is hardware simulation with ModelSim and Vivado. Since seizure prediction is a binary classification between pre-ictal and normal periods, machine learning, a method of data analysis and statistical observation that is made by computer, is used extensively for the software phase. Support Vector Machine, SVM, is a supervised machine learning algorithm that is mostly used in classification problems and it was ideal since our problem is mainly a binary classification problem.

Our simulation was mainly dependent on the CHB-MIT Scalp EEG Database offered by PhysioNet was part of a Ph.D. by Ali Shoeb. [16] The dataset was recorded by placing electrodes on the scalp of 23 subjects to predict epileptic seizure. After multiple simulations, we set the pre-ictal period as a 30-minute window before the start of the ictal period and for the sake of having a continuous limitation free dataset, we choose six patients to work with.

Choosing the suitable features was the main objective of the software simulation phase. The decision takes accuracy, specificity, and sensitivity into account. After researching, and multiple Matlab simulations, only two combinations of 13 features and 2 SVM kernels were promising:

- Four linear features (Standard deviation, root mean square, absolute mean, coastline) along with the linear kernel (SMO).
- One nonlinear feature (Hurst) along with the nonlinear kernel (RBF).

Segmenting the EEG data was found to be very effective in improving the overall performance of seizure prediction. After multiple simulations with different segments for the previous two feature-kernel combinations, we chose to segment the pre-ictal period into 5 segments for the nonlinear combination and 89 segments for the linear combination.

The second phase was the hardware simulation phase. The main goal of this phase is to quantify the hardware needs for each of the chosen software feature-kernel combinations. The decision is mainly based on the chip area, power, and delay. We used FPGA, a device that's made up of thousands of Configurable Logic Blocks and embedded in an ocean of programmable interconnects, as our tool for such simulation. The FPGA can perform the SVM training on the data much faster than the CPU due to the parallelism employed by the FPGA as it supports fast memory access. The kit we used for this purpose was *Xilinx Zynq-7000 SOC ZC702 Evaluation Kit*.

The hardware simulation showed a tradeoff between the area and the power. In the case of linear feature-kernel combination, the power is minimized, but the area is much bigger relative to the second option. On the other hand, the nonlinear feature-kernel combination presented a smaller, more compact area but with a 38.46% higher power consumption than the first option.

6.5 Future work

ASIC implementation could be the next option for having a working seizure prediction device. Moreover, on-the-chip training implementation could be considered. We train the model before implanting the chip using Matlab. However, training the model while implanted on the brain could possibly yield higher prediction performance overtime and more doable.

6.6 Achievements

6.6.1 Competitions

6.6.1.1 JAC-ECC 2018

We had an **accepted poster** in Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC 2018) competition as well as attending some useful technical sessions during the conference program in the field of electronics and communications and our poster presentation was able to secure the **3'rd place** in this competition after going through reviewing processing via a team of expertise in the area of the project.

This day was full of experience technically and socially we did great in presenting our work to the audience, we made new friends, we knew more about the Japanese culture and we enjoyed gaining knowledge from professors and expertise.

Figure 6-3 JAC-EEC certificate of achievement

6.6.1.2 CRC FPGA Competition

We participated in an FPGA Cryptography competition where we are given a certain message and a key and we should figure out the encryption algorithm and implement the decryption algorithm on a remotely programmable FPGA.

In this competition, the user does not know the implemented encryption algorithm. We should search and figure out the encryption algorithm by ourselves. We did this by decrypting the message against different online tools, and here's one of them: http://tripledes.online-domain-tools.com/

By applying the given encrypted message and the key on the online tool and trying the given different algorithms (AES, DES, 3DES, BLOWFISH, BLOWFISH-compat, RIJNDAEL_256, RC4 (ARCFOUR), SERPENT and TWOFISH) we found that the decrypted message was "congratulations." while the decryption algorithm is AES (advanced encryption standard).

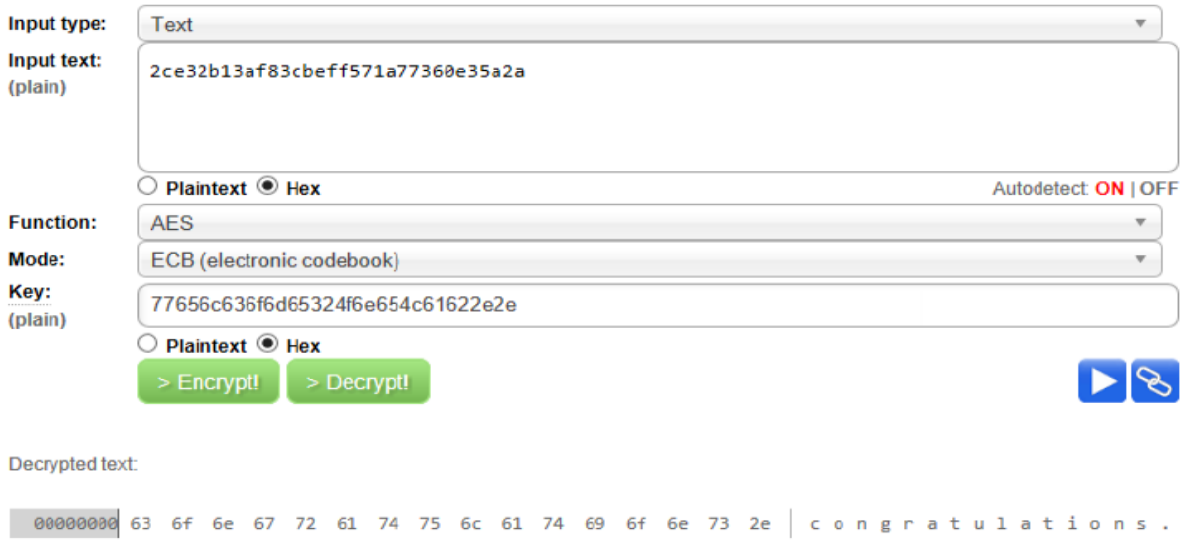


Figure 6-4 Algorithm detection via an online tool

Our implemented HDL code was then simulated and tested on Vivado tool without any errors or warnings and yielded very promising results in terms of power analysis, throughput and utilization.

Finally, we were able to win the **3'rd place** in this competition as well.

6.6.1.3 Dell EMC Competition

For the first phase, we had an accepted Abstract Submission that answered necessary questions regarding our project such as what is the problem exactly that we're tackling? What's our solution? What's novel about our approach? It also included high level description of the main project's flow and architecture. We were then qualified to the final phase after submitting an Interim Design Report that had more in-depth information about our project like detailed project description, project plan, system requirements, system design and implementation including the platforms that we're going to use.

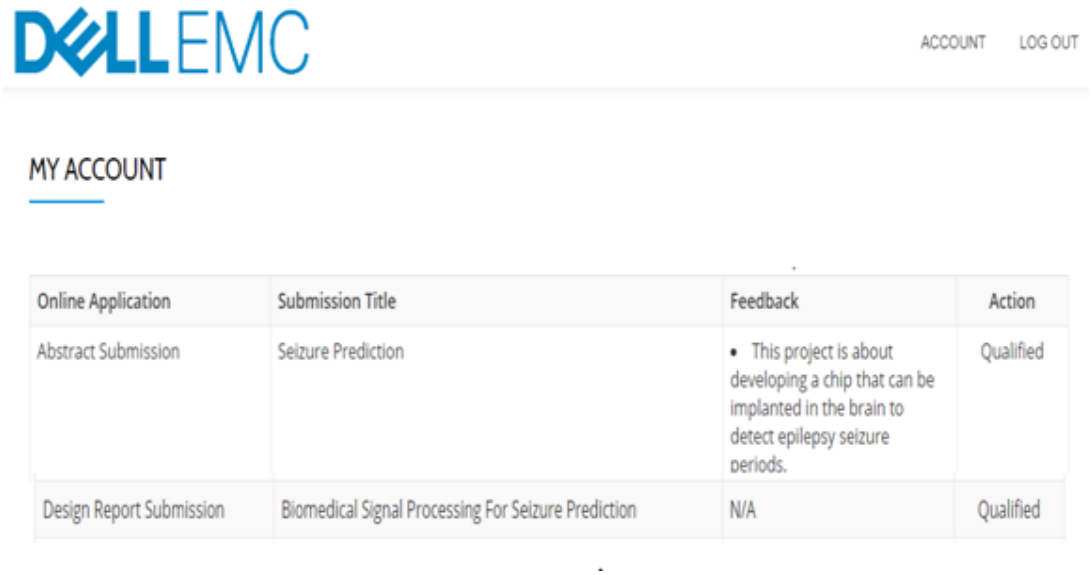


Figure 6-5 DELLEMC Qualification to the final phase

6.6.2 NTRA Funding

We have previously applied for the NTRA Graduation Project Funding by sending our proposal and budget sheet and luckily, we received an acceptance email, thus our graduation project will be funded by NTRA.

Figure 6-6 NTRA's Funding

Bibliography

[1] "Hurst exponent," 16 2 2019. [Online]. Available: https://en.wikipedia.org/wiki/Hurst_exponent?fbclid=IwAR3dfqlSv5NVeWbykLzeD5aCNW0CSr85W1nk4Ub89gwQ4DDSNEhvgMGfrOo.

[2] National Academy of Neuropsychology Foundation, "What is Epilepsy?," [Online]. [Accessed April 2019].

[3] M. Brodie, S. Barry, G. Bamgaous, J. Norrie and P. Kwan, "Patterns of treatment response in newly diagnosed epilepsy," *Neurology*, no. 78, p. 1548, 2012.

[4] P. Boon, T. Vandekerckhove, E. Achten, E. Thiery, L. Goossens, V. K, D. M, G. Van Hoey, B. Vanrumste, B. Legros and e. al., "Epilepsy surgery in Belgium, the experience in gent," *Acta Neurologica Belgica*, no. 99, p. 256, 1999.

[5] W. H. Organization, "Epilepsy in the WHO Eastern Mediterranean region: bridging the gap," 2010. [Online].

[6] "Seizure Types and Classification," Epilepsy Action Australia, 2017. [Online]. Available: https://www.epilepsy.org.au/about-epilepsy/understanding-epilepsy/seizure-types-and-classification/.

[Accessed April 2019].

[7] S. Sanei and J. A. Chambers, EEG Signal Processing, Hoboken: John Wiley & Sons Ltd, 2007.

[8] D. L. Andeles, "Proposal for revised clinical and electroencephalographic classification of epileptic seizures," *Epilepsia,* no. 22, p. 489, 1984.

[9] C. von Bartheld, J. Bahney and S. Herculano-Houzel, "The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting," *The Journal of Comparative Neurology,* vol. 18, no. 524, pp. 3865-3895, 15 December 2016.

[10] V. Brodbeck, L. Spinellie, A. M. Lascano, M. Wissmeier, M.-I. Vargas, S. Vulliemoz, C. Pollo, K. Schaller, C. M. Michel and M. Seeck, "Electroencephalographic source imaging: a prospective study of 152 operated epileptic patients," *Brian,* no. 134, p. 2887, 2011.

[11] J. S. Ebersole and S. V. Pacia, "Intracranial EEG substrates of scalp ictal patterns from temporal lobe foci," *Epilepsia,* vol. 38, pp. 642-654, 1997.

[12] T. M. P. V. Lab, "EEG: introduction," [Online]. Available: https://www.medicine.mcgill.ca/physio/vlab/biomed_signals/eeg_n.ht

m. [Accessed April 2019].

[13] S. J. Van Albada and P. A. Robinson, "Relationships between Electroencephalographic Spectral Peaks Across Frequency Bands," *Frontiers in human neuroscience,* no. 7, p. 56, 2013.

[14] J. L. Cantero, M. Atienza, R. Stickgold, M. J. Kahana, J. R. Madsen and B. Kocsis, "Sleep-dependent θ oscillations in the human hippocampus and neocortex," *The Journal of Neuroscience,* vol. 23, p. 10897, 2003.

[15] P. J. M and S. Palva, "New vistas for α-frequency band oscillations," *Trends in neurosciences,* vol. 30, p. 150, 2007.

[16] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. Ivanov, R. Mark, J. Mietus, G. Moody, C.-K. Peng, H. Stanley, PhysioBank, PhysioToolkit and PhysioNet, "Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215," 13 June 2000. [Online]. Available: http://circ.ahajournals.org/cgi/content/full/101/23/e215. [Accessed 15 April 2019].

[17] "Understanding Support Vector Machine algorithm from examples (along with code)," 13 9 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

[18] "Support Vector Machine vs Logistic Regression," 12 8 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f.

[19] "7 Types of Classification Algorithms," 19 1 2018. [Online]. Available: https://www.analyticsindiamag.com/7-types-classification-algorithms/.

[20] I. Dabbura, "Gradient Descent Algorithm and Its Variants," 21 12 2017. [Online]. Available: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3.

[21] "Artificial Neural Networks Advantages and Disadvantages," 27 1 2018. [Online]. Available: https://www.linkedin.com/pulse/artificial-neural-networks-advantages-disadvantages-maad-m-mijwel/.

[22] "https://www.quora.com/What-are-the-objective-functions-of-hard-margin-and-soft-margin-SVM," 26 11 2006. [Online]. Available: https://www.quora.com/What-are-the-objective-functions-of-hard-margin-and-soft-margin-SVM.

[23] J. Platt, "Sequential minimal optimization: A fast training of support vector machines.," *Technical Report, Microsoft Research,* no. 14, p. 98, 1998.

[24] H. Kuhn and A. W. Tucker, "Nonlinear programming," in *Second Berkeley Symposium*.

[25] "The Hurst Exponent," 29 10 2018. [Online]. Available: https://blog.quantinsti.com/hurst-exponent/?fbclid=IwAR3FhhDVup2AMzPntbqgPfyJV6QLjjC04UvKyxdQ1wFhiam2aT7RzM5oHo8.

[26] S. Mansukhani, "The Hurst Exponent: Predictability of Time Series," 8 2012. [Online]. Available: http://analytics-magazine.org/the-hurst-exponent-predictability-of-time-series/?fbclid=IwAR0PxvQjAaTk7Lyhd4V_tb6zOnNJSKdsBvQQ8ibc3FAkVn9D3duBum5pVlY.

[27] D. G. Ray, "BETWEEN FIXED AND FLOATING POINT," 4 2 2010. [Online]. Available: http://chipdesignmag.com/display.php?articleId=3921&fbclid=IwAR1MK1KBGPwwLk6jc2dqPiV0QC_7bYwooTUCxyNTuGw3qUuaPMr-5bRPQB0.

[28] N. Instruments, "FPGA Fundamentals," 19 5 2019. [Online]. Available: http://www.ni.com/en-lb/innovations/white-papers/08/fpga-fundamentals.html.

[29] xilinx, 10 1 2013. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx201

2_4/ug907-vivado-power-analysis-optimization.pdf.

[30] N. University, "Accurate Area and Delay Estimators for FPGAs," [Online]. Available: https://www.date-conference.com/proceedings-archive/PAPERS/2002/DATE02/PDFFILES/08E_2.PDF.

[31] Forscher; Epileptologie Bonn; AG Lebnertz, "EEG time series download page," Forscher, 15 Mars 2009. [Online]. Available: http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3&changelang=3 .

[32] "EEG Database – Seizure Prediction Project Freiburg.," Freiburg, [Online]. Available: http://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database.

[33] "When do support vector machines trump other classification methods," 28 1 2013. [Online]. Available: http://www.simafore.com/blog/bid/112816/When-do-support-vector-machines-trump-other-classification-methods.

[34] "Xilinx Zynq-7000 SoC ZC702 Evaluation Kit," [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html.

[35] "CRC-LAB," [Online]. Available: http://41.32.238.244:58825/accounts/login/?next=/portal/dashboard/.

[36] "mayoclinic," [Online]. Available: https://www.mayoclinic.org/tests-procedures/deep-brain-stimulation/about/pac-20384562.

[37] "wearable-technologies," july 2016. [Online]. Available: https://www.wearable-technologies.com/2016/07/wearables-rescue-epilepsy/.

[38] "neuropro," [Online]. Available: http://www.neuropro.ch/winam.

[39] B. B. Mandelbrot, The fractal geometry of nature, Macmillan, 2012.

[40] "Mathplanet," [Online]. Available: https://www.mathplanet.com/education/algebra-2/quadratic-functions-and-inequalities/standard-deviation-and-normal-distribution.

[41] norris, "peopleuncw," Matlab, [Online]. Available: http://people.uncw.edu/norris/133_sp04/Onotation/someFunctions1.html.

[42] "Physionet," Physionet, [Online]. Available: https://physionet.org/cgi-bin/atm/ATM.

[43] "3DES – Symmetric Ciphers Online," [Online]. Available: http://tripledes.online-domain-tools.com/.