University of Science and Technology at Zewail City
Department of Nanotechnology Engineering

Biomedical Brain Signal Processing for Epileptic Seizure Detection/Prediction

by

Ahmed Abubakr Ahmed

Ahmed Ibrahim Mohamed

Mostafa Amer Mahmoud

Yehya Ateya Alhazek

Supervised by: Dr. Hassan Mostafa Hassan

Senior Design Project Thesis submitted to fulfill the Bachelor's Degree

June 2018

Abstract

Epilepsy is one of the most common neurological disorders, affecting millions of people worldwide, characterized by an abnormality in the brain activity which leads to recurrent seizures. Owing to the unpredictable nature of epileptic seizure, it represents a major worry and a handicap to epileptic patients causing serious injuries such as fractures and vehicle accidents. The ability to detect, predict and prevent the occurrence of epileptic seizures is very important to prevent such injuries and let epileptic patients lead a normal life.

In this project we developed a hardware chip to predict/detect the seizure onset using a machine learning technique named "Support Vector Machine" (SVM). A training algorithm was developed namely the "Sequential Minimal Optimization" (SMO) which was used in the model development.

A number of discriminant time domain and frequency domain features were extracted namely the power spectral density (PSD), Hjorth mobility, Hjorth complexity, skewness and kurtosis, coastline and average energy features. The raw EEG signal database used were collected at the Children's Hospital Boston [8], in addition, we are planning to extract our own database from rats and use it in the future.

Selection of features was done by inspection to get the most discriminant features from the obtained high dimensional feature space, these features were used to train the SVM using the SMO algorithm on MATLAB program. Then, the developed model is tested for its ability to detect the seizure accurately.

After that, the high-level language (MATLAB) model was transferred to a hardware description language model using HDL programs for hardware implementation. This phase was very important through which an FPGA prototype and ASIC tape-out ready for fabrication was produced as final outcome. Low power and low complexity were some of the most important metrics to take into consideration while designing our prototype.

Acknowledgements

First of all, we would like to acknowledge the continuous help and support provided by our great mentor and supervisor Dr. Hassan Mostafa. His steady and continuous guiding throughout the whole project, in addition to his comments and discussions were of invaluable help.

We would like to express our gratitude for our colleague Karim El-kholy at the Department of Nanotechnology Engineering, Zewail University for his help, support and for the valuable explanations and discussions we had with him.

We would like also to thank the National Telecommunications Regulatory Authority (NTRA) and the Information Technology Industry Development Agency (ITIDA) for the funds received to build our prototype.

Last but not least, we would like to thank our beloved families and friends for their continuous support and sincere love that gave us the power to do our best.

Contents

List of Figures

List of Tables

Abbreviations

ADC	**A**nalog to **D**igital **C**onverter
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
CORDIC	**C**oordinate **R**otation **D**igital **C**omputer
DAC	**D**igital to **A**nalog **C**onverter
DBS	**D**eep **B**rain **S**timulation
EDF	**E**uropean **D**ata **F**ormat
EEG	**E**lectroencephalography
FDR	**F**alse **D**etection **R**ate
FFT	**F**ast **F**ourier **T**ransform
FIR	**F**inite **I**mpulse **R**esponse
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
GDR	**G**ood **D**etection **R**ate
HC	**H**jorth **C**omplexity
HDL	**H**ardware **D**escription **L**anguage
HLL	**H**igh **L**evel **L**anguage
HM	**H**jorth **M**obility
ITIDA	**I**nformation **T**echnology **I**ndustry **D**evelopment **A**gency
KKT	**K**arush-**K**uhn-**T**ucker
MAC	**M**ultiplier **A**ccumulator
ML	**M**achine **L**earning
NTRA	**N**ational **T**elecommunications **R**egulatory **A**uthority
PSD	**P**ower **S**pectral **D**ensity
QP	**Q**uadratic **P**rogramming
RBF	**R**adial **B**asis **F**unction

ROC **R**eceiver **O**perator **C**haracteristic

SLT **S**tatistical **L**earning **T**heory

SMO **S**equential **M**inimal **O**ptimization

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

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

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

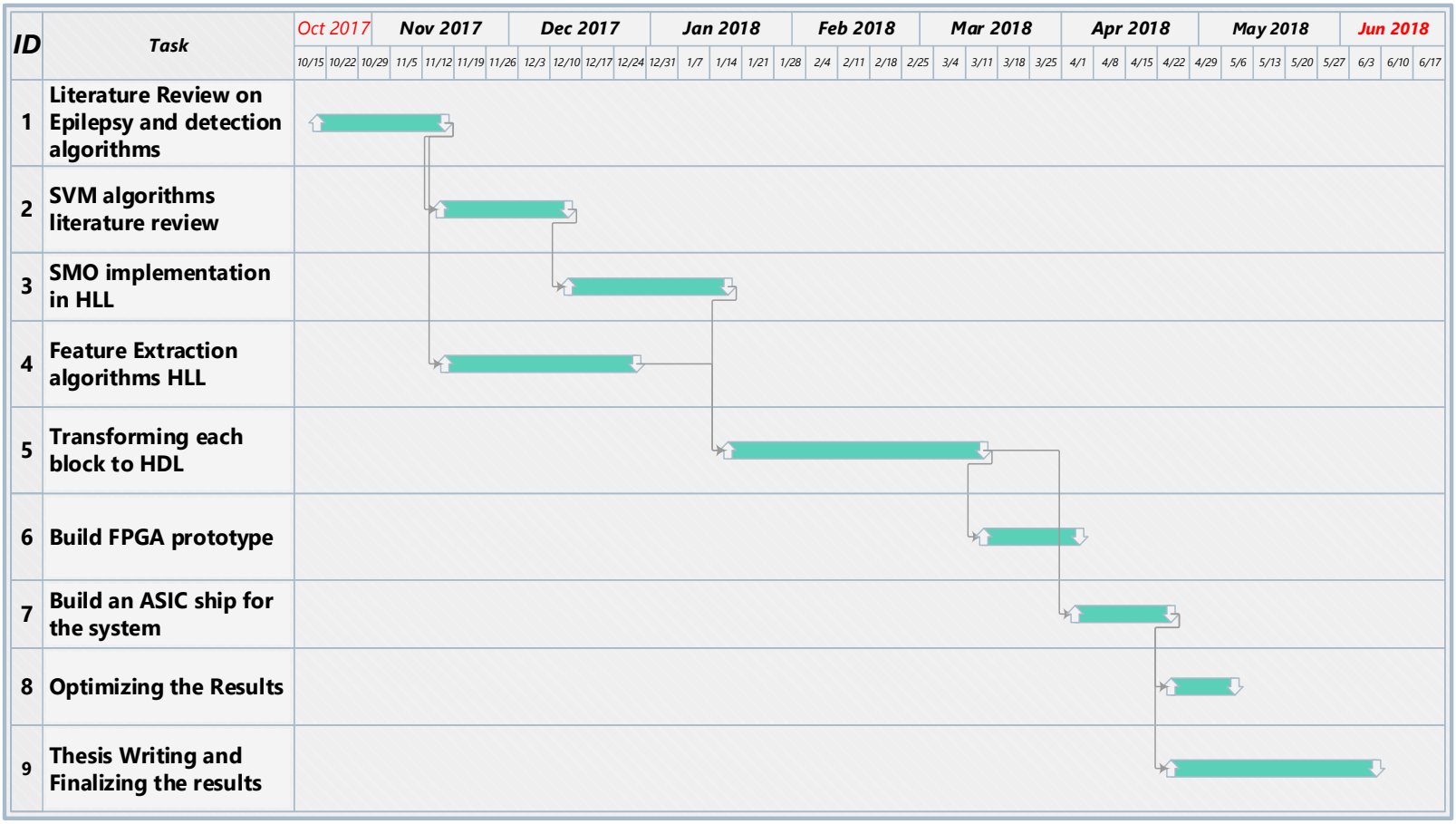
WHO **W**orld **H**ealth **O**rganization

List of Symbols

$\vec{w}$	Margin
x_i	Input vector
B	bias
x_+	Input producing positive category
x_-	Input producing negative category
y_i	Category or label
α_i	Lagrange multiplier if the ith input
ξ_i	Slack variable
C	A parameter capturing the trade-off between having a wide margin with a small number of margin failures or cost function in Artificial neural network
r_i	Zeta requirement handler
$K(x_i, x_j)$	Kernel function
$\Phi(\mathrm{x_i})$	The value of x_i after the transformation into the higher dimension space
α_2^{new}	New value of the second Lagrange multiplier
$\alpha_2^{new,clipped}$	New clipped value of the second Lagrange multiplier
L	Lower bound on alpha
H	Upper bound on alpha
Ψ_L	SMO minimization function on L
Ψ_H	SMO minimization function on H
$a_j(t)$	Artificial neural network activation value
θ_j	Artificial neural network threshold
$x_j(t)$	Artificial neural network input
w_{ij}	Artificial neural network connection weight
$p_j(t)$	Artificial neural network propagation function
$O_i(t)$	Artificial neural network output of neuron *i*
$g_i(x)$	Composition of other function like the propagation

K	Activation function
P_i	Spectral power of i^{th} sub-band
P_{tot}	Total spectral power
σ	Variance
μ	The sample mean
N	Number of data points
k	Kurtosis
$E(x)$	The expected value of x.
M_k	Statistical moment

Gantt Chart


ID
Task
Oct 2017
Nov 2017
Dec 2017
Jan 2018
Feb 2018
Mar 2018
Apr 2018
May 2018
Jun 2018
10/15 10/22 10/29 11/5 11/12 11/19 11/26 12/3 12/10 12/17 12/24 12/31 1/7 1/14 1/21 1/28 2/4 2/11 2/18 2/25 3/4 3/11 3/18 3/25 4/1 4/8 4/15 4/22 4/29 5/6 5/13 5/20 5/27 6/3 6/10 6/17
1 Literature Review on Epilepsy and detection algorithms
2 SVM algorithms literature review
3 SMO implementation in HLL
4 Feature Extraction algorithms HLL
5 Transforming each block to HDL
6 Build FPGA prototype
7 Build an ASIC ship for the system
8 Optimizing the Results
9 Thesis Writing and Finalizing the results

"Dedicated to the Memory of Engineer Karim Abo El Makarem"

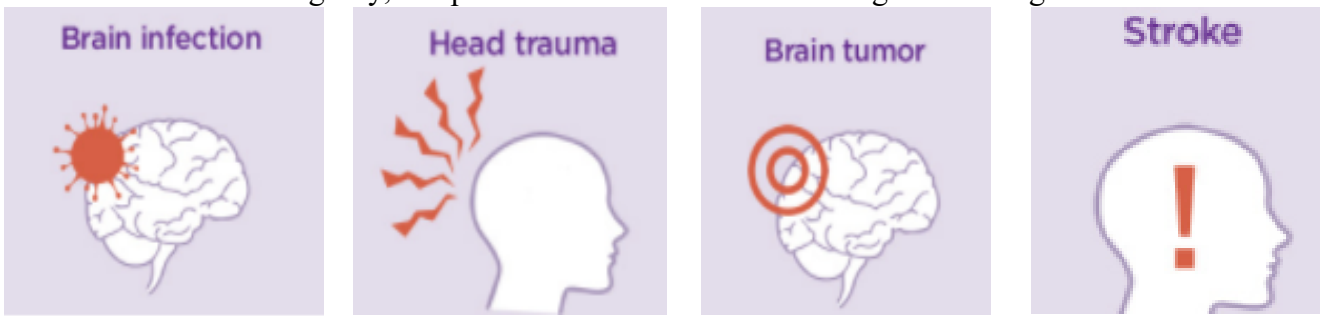
Chapter 1

In this chapter, a complete review about epilepsy, its causes, types, treatments and statistics is presented.

Epilepsy and Its Causes

Epilepsy is a long-lasting neurological disorder which causes the patient to experience frequent spontaneous seizures that take place without any previous warning. A seizure is an electrical surge in the brain caused by structural malfunctions of the brain [9]. People can experience one or more seizures in their lifetime, however, they are not diagnosed with epilepsy. Epilepsy is defined as having two or more unprovoked seizures within a time period of at least 24 hours [10]. Epilepsy disease is more common among younger and older people with a slight more tendency in males than females [11].

There are several reasons why one might experience seizures and, accordingly, get diagnosed with epilepsy. The four most common reasons are summarized in Figure 1. First reason shown in Figure 1(a) is the brain infection, which is mainly caused by viral infection either causing inflammation and swelling of the brain itself (encephalitis disease) or causing acute inflammation of tissue layers enclosing the brain and the spinal cord (meninges disease) [12]. Second reason shown in Figure 1(b) is the head trauma, which is an injury to any of the scalp, the skull or the brain that may cause a concussion in which the brain is shaken from its normal place. This injury may be caused by accidents, falls or physical assault. Third reason shown in Figure 1(c) is the brain tumor, which is a group of abnormal cells that continuously grow inside a restricted area enclosed by the skull, and due to the skull rigidity, the pressure inside increases to a degree that might cause brain



(a) (b) (c) (d)

Figure 1: The four most common causes of epilepsy; (a) Brain Infection, (b) Head Trauma, (c) Brain Tumor and (d) Stroke

damage and might be life-threating. Being infected with a brain tumor may be a side effect to chemical exposure, exposure to radiation, family history or increasing age [13]. Fourth reason shown in Figure 1(d) is the stroke, which is caused by the lack of blood flow to the brain leading to Oxygen shortage and cell death [14].

A survey was done by the World Health Organization in the Eastern Mediterranean Region in 2010 to see the most common causes of epilepsy [3]. Nine countries from the region responded to the survey. The result shown in Figure 2 shows that "Trauma" was the most common of all the causes with all the countries voting for it which agrees with what was stated before. After "Trauma" comes the "Central Nervous System Infection" in the second place with a voting of 85% from the responding countries. We can also see "Tumors" as a main contributing factor to the epilepsy disease.

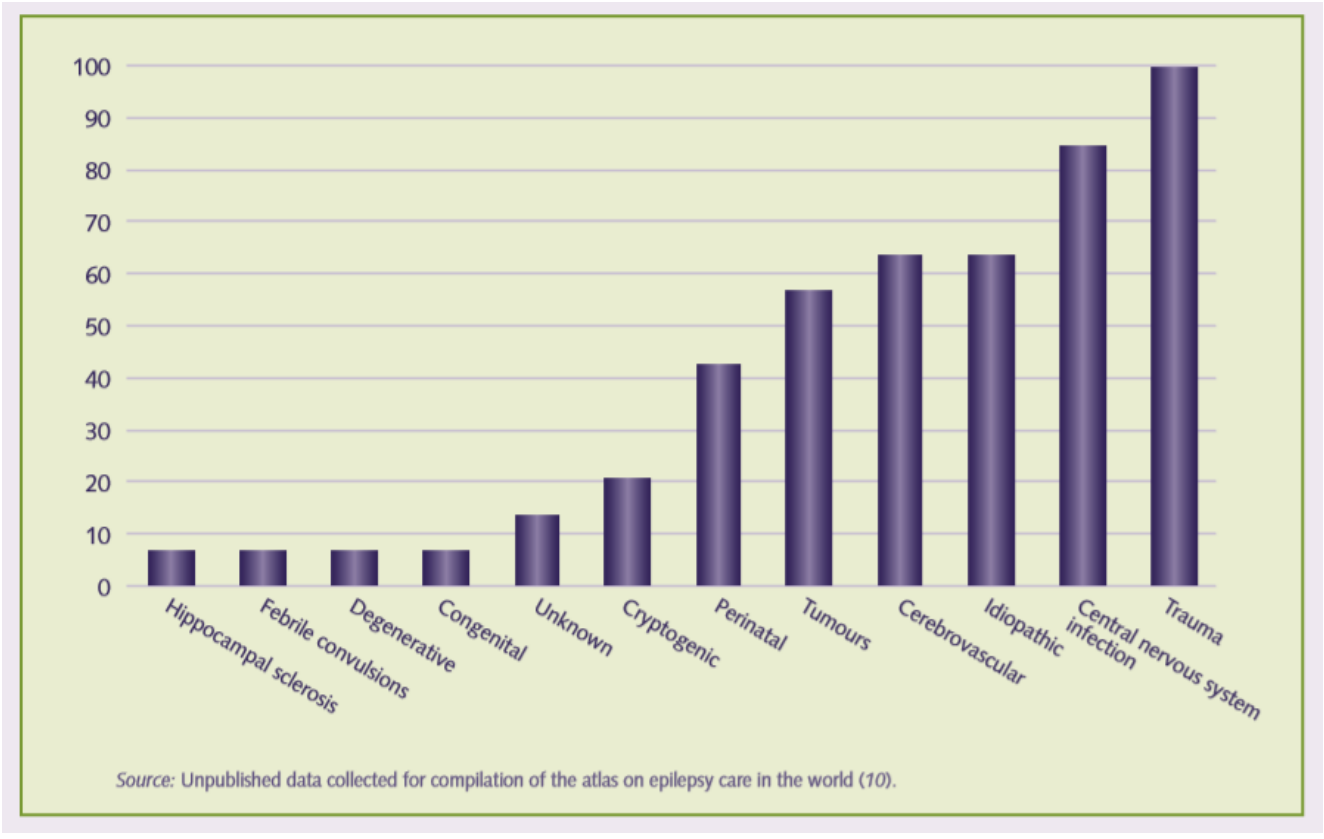


Figure 2: Most frequently reported causes of epilepsy disease in the Eastern Mitterrandian Region [3].

Types of Seizures

The most known and main symptoms of having the epilepsy disease are seizures. Seizures are different from patient to another in which different parts of the brain cannot function

normally [10]. They could be either mild which lasts several seconds during which you lose awareness and it is difficult to recognize as a seizure, or stronger seizures which can last from seconds to several minutes during which your body experience muscle twitches and involuntary movements, in addition to losing consciousness and getting confused [11].

Partial Seizures

Seizures are divided into two main different types as shown in Figure 3, first type is the partial (focal) seizures and second type is the generalized seizures. Partial seizures refer to

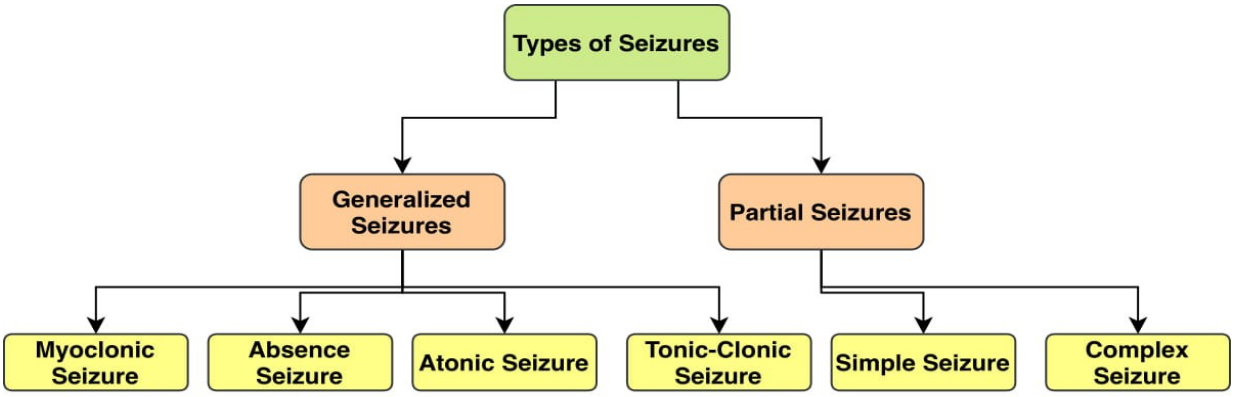


Figure 3: Types of seizures are divided into "Generalized" and "Partial" seizures.

seizures occurring in only one part of the brain as shown in Figure 4 which may spread later to other parts of the brain and they are further divided into two forms:

- **Simple Partial Seizures:** where patients may suffer from involuntary movements (limbs tingling and twitching) and abnormal sensations; however, they stay aware during the seizure with no loss of consciousness and can communicate with the surroundings.

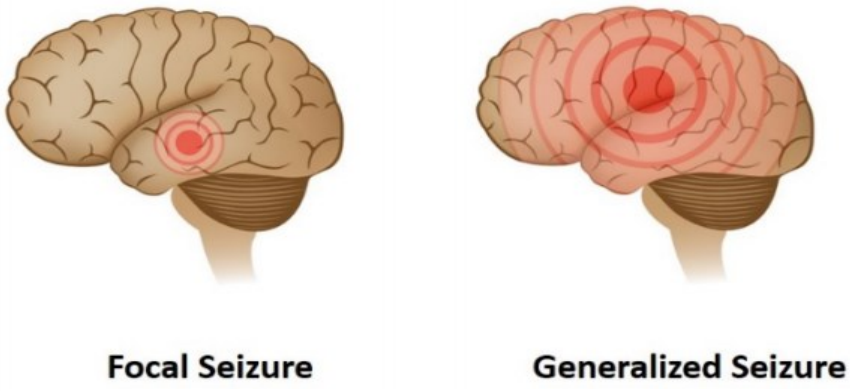


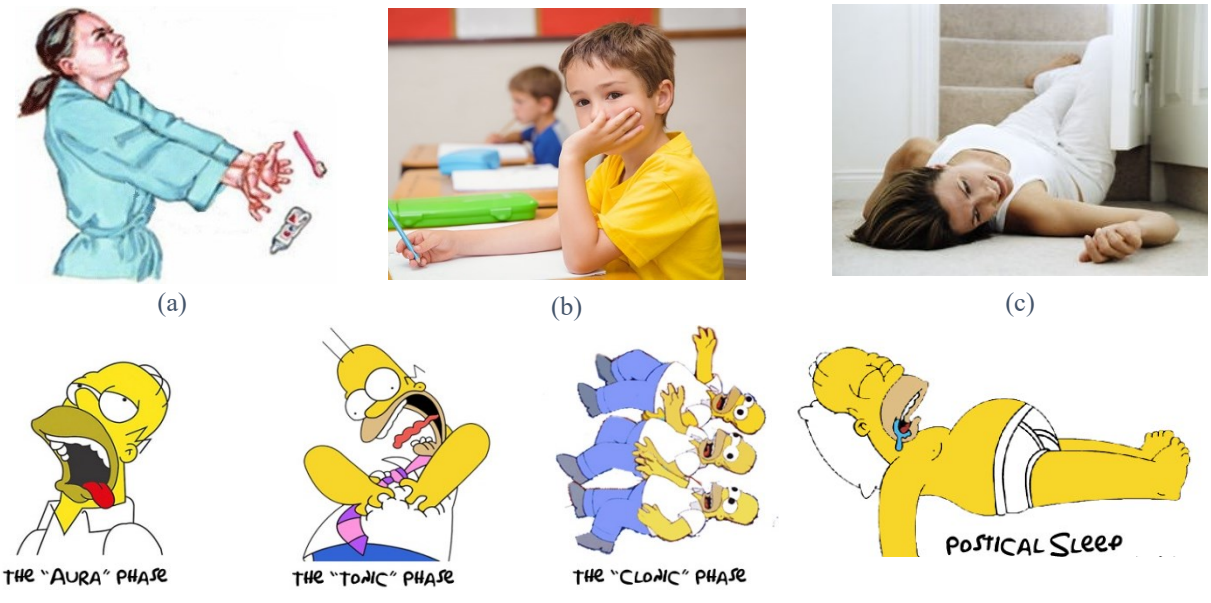
Figure 4: Focal seizures (left) affecting one part of the brain and generalized seizures (right) affecting the whole brain.

- **Complex Partial Seizures:** where patients experience seizures that spread to certain parts of the brain causing loss of consciousness and awareness, in addition, they become unresponsive to the surroundings.

Generalized Seizures

Generalized seizures refer to seizures occurring as a result to the hyperexcitability of the whole brain as shown in Figure 4 which makes it difficult to identify the cause the seizures. They are furtherly divided into four forms:

- **Myoclonic (Muscle -Jerk) Seizure:** This seizure is accompanied by increase in muscle tone and a spontaneous movement of arms and legs as if the patient has an electric shock shown in Figure 5(a).
- **Absence Seizures:** This seizure begins without any warning causing the patient to stare into space with a short loss of awareness for several seconds (~15 sec) shown in Figure 5(b).
- **Atonic Seizures:** This seizure cause the patient muscles to go complete stiff or paralyzed causing sudden falls which may lead to serious injuries shown in Figure 5(c).
- **Tonic-Clonic Seizures:** This seizure goes through four phases, first phase is the "Aura" phase in which the patient feels light dizziness and confusion, second phase



(a) (b) (c) (d)

Figure 5: Generalized seizures forms: (a) Myoclonic, (b) Absence, (c) Atonic, and (d) Tonic-Clonic seizures.

is the “Tonic” phase in which the patient’s limbs and body parts straighten involuntary and he/she starts to lose consciousness, third phase is the “Clonic” phase in which the patient suffers from twitching, rolling and violent shaking, fourth phase is the “Postictal Sleep” phase in which the patients starts to gain consciousness gradually feeling nausea and confusion [10] shown in Figure 5(d).

The symptoms of a seizure may affect any part of the human body; however, its origin is an electrical event inside the brain of the patient. The actual location of that electrical event inside the brain, its spread, how the brain is affected with it and how long it lasts are all factors that determine what type of seizure the patient will have and its impact on him/her.

Epilepsy Statistics

Epilepsy is the 4th most common neurological disorder disease worldwide, about 65 million people around the world suffer from epilepsy (~1% of the world population), about one in every twenty six people in a lifetime is affected by the epilepsy disease [15]. According to the World Health Organization, more than 85% of the cases are from the developing world as shown in Figure 6 and about 4.7 million people are from the Eastern Mediterranean region [3]. It was reported in 2014 that each year about 150,000 Americans are diagnosed with epilepsy and that Americans pay more than 15.5 billion dollars every year on the

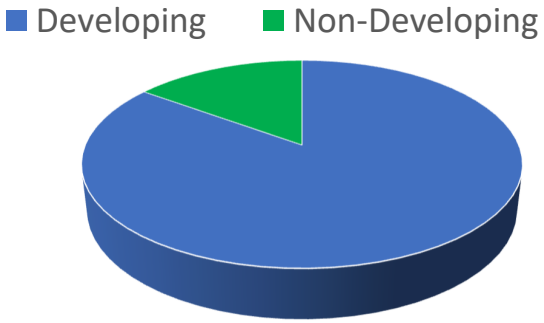


Figure 6: Around 85% of the people diagnosed with epilepsy are from the developing regions.

treatment of epileptic patients [16].

There are two definitions by which the spread of the epilepsy disease is quantized, first is prevalence and second is incidence. Prevalence is the proportion of people with the disease to the whole population in a given period of time. The purpose of prevalence is to determine

exactly the number of people suffering from the disease in order to plan for their treatment. Incidence is number of new cases diagnosed with the disease at a given time. The purpose of incidence is to get a better understanding of the history of epilepsy and the reasons behind the disease [17].

A study made in 2013 on the prevalence and incidence of the epilepsy disease in Egypt [18], it was found that the crude prevalence rate is 12.67/1000, crude means it was calculated directly by dividing the number of cases in a given time period by the population number. We can say that if Egypt's population number in 2013 was around 90 million, then the number of epileptic patients in Egypt in 2013 is estimated to be 1,140,000 patients. The incidence rate was reported roughly to be 1.5/1000 with less conducted studies unfortunately. The reported cases suffering from generalized seizures were more than those suffering from partial seizures by a ration **2.7:1**. It was also noted that the male cases were slightly higher than the female cases, in addition, the cases reported in rural areas were more than this reported at urban areas.

Traditional Treatment Methods for Epilepsy

Owing to the unpredictable nature of epileptic seizure, it represents a major worry and a handicap to the patients, for example the occurrence of an epileptic seizure to a patient driving a car or carrying out a dangerous job like working in a bakery or operating a cutting machine, will cause serious damage and a disastrous accident to the patient. Thus, researchers, practitioners and doctors are paying great attention to every possible way of treating this disease.

Drug Treatment

Long term drug treatments (pharmacotherapy) are usually employed as a first line of defense with patients. These drugs help reduce the frequency of seizures happening but cannot stop a seizure once started, they are typically in the form of tablets absorbed by the stomach, once in the blood stream they travel to the brain affecting the neurotransmitters in a certain way to reduce the electrical activity causing the seizures [11]. Common epilepsy medical drugs are available at the market, however, more than 30% of epileptic patients are drug resistant [19].

Surgery

The alternative when these medications fail is to remove the area of the brain which is responsible for the epileptic seizure. Most often, the part to be removed is the temporal lobe, shown in Figure 7, in a surgery named "Temporal Lobectomy", during the surgery patients are kept awake so that doctors can communicate with them and make sure that there is no part of the brain responsible for important functions in the body is removed [11]. If the part to be removed is large or important, another surgery called "Multiple Subpial Transection" is performed in which the doctors make a cut in the brain to interrupt the pathway of the nerve, thus, keeping the seizure confined to one place.

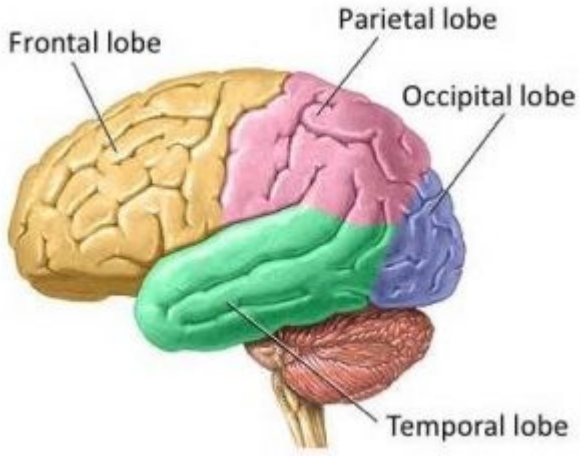


Figure 7: Human brain lopes.

In general, surgery is recommended only if the seizure is confined to one area of the brain. The probability of becoming seizure free after surgery depends on the seizure type, for example in temporal lobe epilepsy surgery it's 75% in lesional cases and only 50% in non-lesional cases, in frontal lope epilepsy it's 60% and 35% in lesional and non-lesional cases respectively [20]. However as shown in previous work [21-23], epilepsy is not confined to a single area of the brain but rather it's epileptic network where different areas of the brain interact synchronously causing these pathological spikes or seizures.

Vagus Nerve Stimulation

For those who are drug-resistant and cannot undergo a surgery, a device like the pacemaker was developed to be inserted in the patient's chest under skin as shown in Figure 8. This device is designed to deliver electrical impulses through the Vagus nerve running through

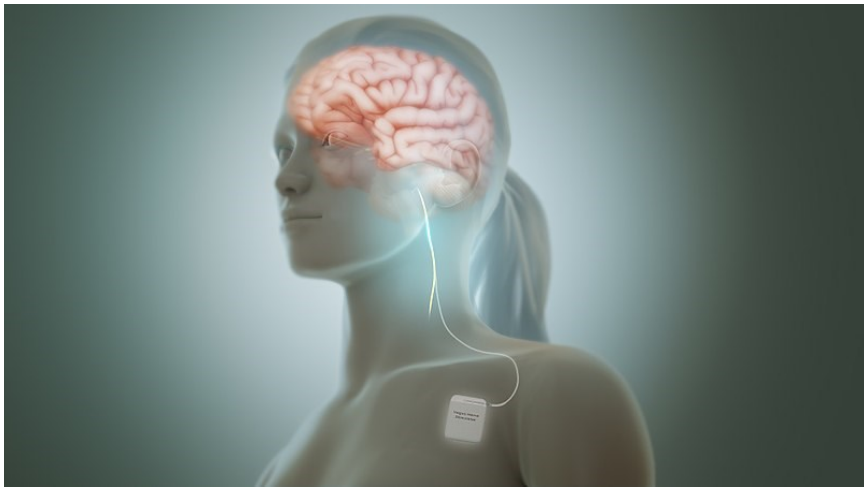


Figure 8: Electrical stimulation of Vagus nerve [5].

your neck, accordingly it prevents brain overexcitation, thus, reducing the likelihood of seizures [10]. 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 [5].

Diet Therapy

Several diet regimes are proved to be beneficial in decreasing the frequency of seizures, one of these regimes is the Ketogenic diet in which patients have to feed on high fat and low carbohydrates food [10]. Approximately half of the patients who cannot respond to medication can benefit from this therapy, however, there is still a valuable portion of the patients who cannot benefit.

Modern Treatment Methods

A new technique was developed in 2006 called "Deep Brain Stimulation" (DBS) where some electrodes implanted in the patient's brain can sense any abnormality in brain's activity, accordingly a generator implanted in the chest will generate electrical pulses to suppress any detected seizures [24]. Researchers and practitioners are paying great attention to developing algorithms that can detect and anticipate the occurrence of a seizure (seizure detection and prediction). Afterwards a stimulus is applied to suppress the seizure

using closed loop or open loop strategies [25, 26]. The algorithms mentioned above employ the use of brain EEG signals which will be explained in more details shortly.

However, there is a great difficulty in being able to differentiate between the seizure signals, pre-seizure signals and the normal brain signals which in turn makes it hard and difficult to build a good and robust detection/prediction model. There are already commercialized devices that can be implemented in the brain to treat these seizures [27]. However, these devices lack accuracy, are very expensive to afford, are hard to implement and they are power hungry which means that the patient having these devices in his brain must undergo a complex and expensive surgery every two years to only change the battery of these devices.

Our contribution, which we will show throughout the rest of the thesis, is that we developed a low power and less complex implementation for the implantable chip inside the brain with no need for a battery (generator) implemented in the chest. We used a machine learning technique called “Support Vector Machine” (SVM) in developing the detection/prediction procedure

Chapter 2

In this chapter, the EEG signal is explained in details, in addition, the description of the database used is presented. Also, the methods for seizure prediction and detection are summarized. Finally, the complete project flow is presented.

Electroencephalography (EEG) Signal

Electroencephalograph (EEG) is a method used to monitor and measure the activity of the brain, it is based on the principle that the brain consists of neurons that communicate with each other using electrical signals. To understand how this electrical signal is produced and how its measurement can describe the behavior of the brain, we must understand how neurons communicate. A typical neuron is shown in Figure 9, these neurons form the main building block of the brain, they affect your brain activity when you feel something on your skin or feel hunger, fear and other senses, they communicate with each other via what is called action potential, an action potential is an electrical signal generated by neurons to

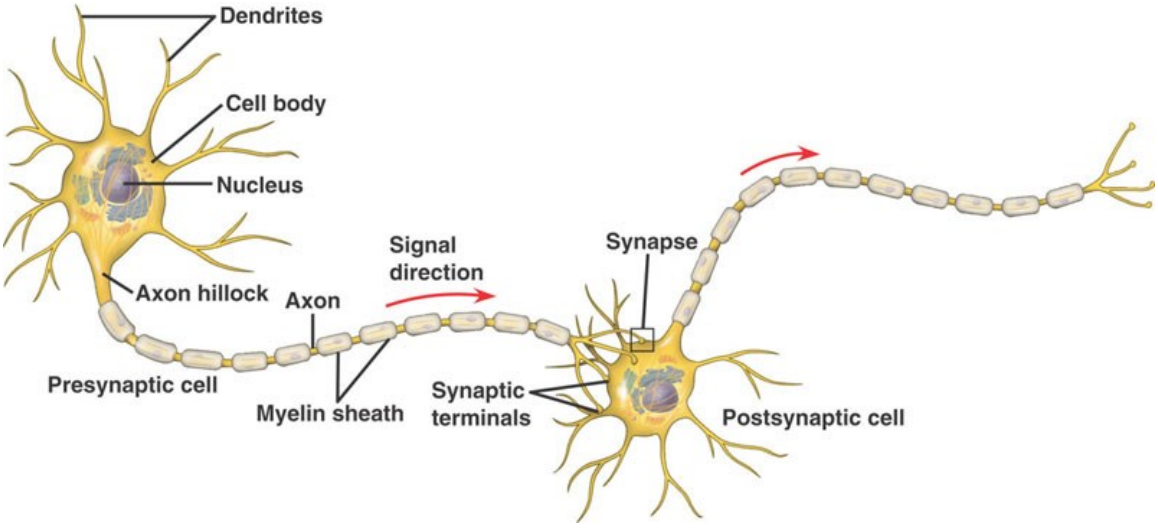


Figure 9:neurons communication [2]

stimulate a communication with each other, when this signal reaches the end of a neuron it leads to the release of a neurotransmitter (chemical compounds) that affects the connection between the neurons leading to binding or joining unconnected neurons. The electrical signal produced by the action potential leads to the production of an electrical signal that is detected and measured by the EEG. a

EEG Signal Types

There are two types of EEG signal measurements, scalp EEG which is a non-invasive measurement with the electrodes placed on the outside of the brain and intercranial EEG which is an invasive measurement with the electrodes placed inside the brain. The most common approach is the scalp EEG because it doesn't involve a brain surgery to place the electrodes inside the brain, however, the accuracy of the intercranial EEG measurements is much higher than the scalp EEG measurements.

EEG Signal Features

There are several features that characterize an EEG signal namely the frequency of the signal, the voltage of the signal, its morphology, synchrony and periodicity.

Frequency:

It measures the repetitive activity of the EEG signal in Hz, there are different properties of the frequency of the EEG, it can be rhythmic in which the EEG waves have constant frequency, arrhythmic in which the EEG waves have variable random frequency and dysrhythmic which is a rarely seen pattern of frequency in a small group of people.

There are four distinct frequency patterns that appear in an EEG signal which are delta[0-3Hz], theta [3.5 to 7.5 Hz], alpha [7.5 to 13 Hz] and beta [>13 Hz] as shown in Figure 10.

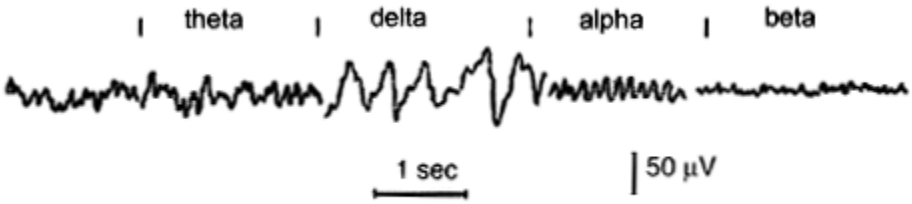


Figure 10: Brain waves frequency *[1]*

Voltage:

There are several terms that describe the average voltage of the EEG signal and are used to characterize the signal, the attenuation which happens when the voltage of the signal is blocked, Hypersynchrony which is an increase in the voltage of the brain due to the increase in the number of neurons producing the signal and Paroxysmal which refers to a sudden rise in the voltage followed by an abrupt return to the normal voltage which is usually associated with epilepsy.

Morphology:

Which refers to the shape of the signal which could be monomorphic, polymorphic, sinusoidal, transient or spikes.

Synchrony:

Refers to the synchronous appearance of a pattern of EEG signals on different areas or the same areas of the brain.

Periodicity:

Which refers to a phenomenon when a certain type of an EEG signal happens periodically in the brain.

Electrode Placement

To generalize the placement of the electrodes on the brain, different standards have been introduced to capture most of the brain signals in a standard manner, a typical standard used in scalp EEG measurements is the 10/20 system, in which the electrodes are placed 10% then 20% away from the consecutive areas of the brain as shown in *Figure 11*. This method was the same used in collecting the data we are working on from [8].

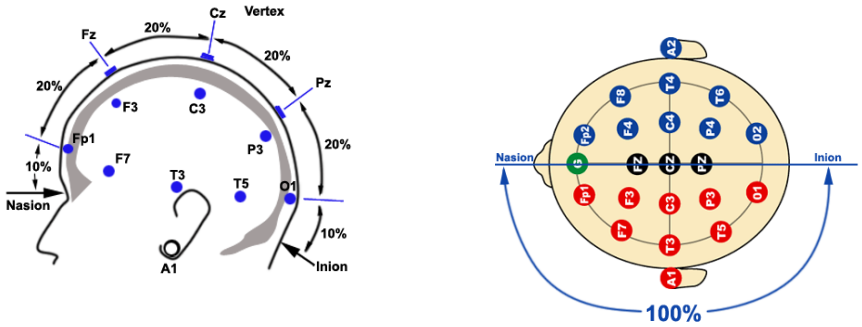


Figure 11: 10/20 electrode placement system [1]

CHB-MIT Scalp EEG Database

The EEG database we used in this project was collected at the Children's Hospital of Boston, it was part of a PhD work done by Ali Shoeb [8]. It consists of EEG recordings to patients with characterized by hard-to-control seizures. The data collected was for 22 patients divided into 5 males (aged 3 years – 22 years) and 17 females (aged 1.5 years – 19 years). Unfortunately, due to hardware limitations, the recorded files for each patient are not continuous with a maximum recording time of 4 hours. This limitation or drawback

made it hard for us to use the data in building and adjusting our prediction model, as a continuous recording of the EEG signal is required and this will be explained soon enough.

All the EEG signals were recorded and sampled at 256 Hz (samples per second) with 16-bit resolution. The 10/20 system mentioned in Figure 11 was used in placing the measurement sensors (electrodes) on the patients' heads giving us 23 EEG Scalp signals one for each channel. All the database for the 23 patients contain a total of 182 seizures. We only used 5 patients in our work due to the limitations of the recording procedures.

There are two main reasons we chose this database specifically; first reason is that the database is available online for free without any charges unlike the Freiburg EEG database, second reason is that we were able to compare our model results, enhancements done and out additions to the work of Ali Shoeb in [8]. We are recording our own data base as explained in the next section for future work.

ONE Lab Rats' EEG Database

Currently and with the help of some Master students at the Faculty of Science, Ain Shams University and under the supervision of ONE Lab organization in the Faculty of

Figure 12: Procedures of recording our new EEG database from rats.

Engineering, Cairo University, we are recoding our own database using rats for future use in building a more accurate and robust prediction/detection model. The photos of recording the intercranial EEG database using rats are shown in Figure 12.

Detection/Prediction Methods

Both detection and prediction are two possible ways leading to the same target which is the suppression of electric seizures, however, the difference between them is that prediction is an early detection of the seizure onset so it is more efficient and safer for researchers to employ prediction than detection. However, implementation of prediction is much harder and costly than detection.

EEG Signal Periods

The brain's EEG signal of a an epileptic patient is divided mainly into 4 differet periods or states by which we can charaterize an eplieptic patient from a normal person [28]. The 4 states are shown in Figure 13 and explained below:

(I) The **ictal state** which is the actual seziure period and usually ranges from 1 to 3 minutes, it could be less and could be more than that.

(II) The **preictal state** which is the period directly before the seziure onset, it usually ranges from 30 to 60 minutes and differes completely from one patient to another.

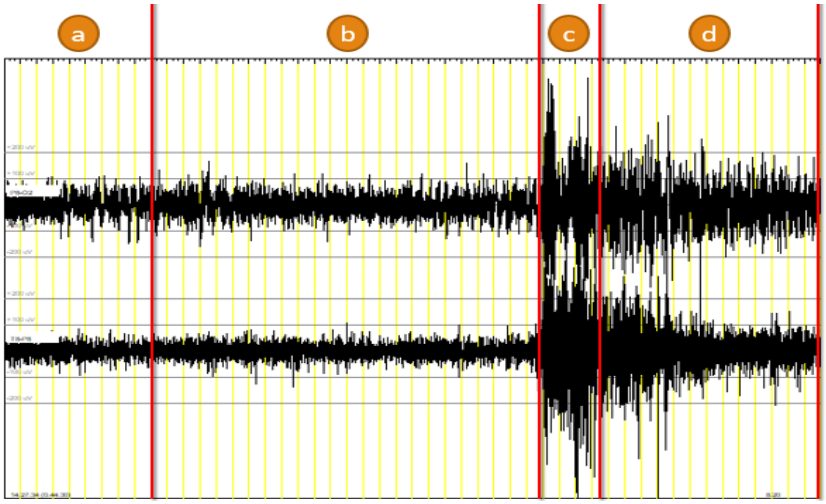


Figure 13: Raw EEG signal obtained from "CHB-MIT Scalp EEG Database" for patient No.3 to show the different stages of the brain signal of an epileptic patient: a) Interictal stage, b) Preictal stage, c) Ictal stage and d) Postictal stage.

(III) The **post-ictal state** which is the period directly after the seziure, it ranges usually from 30 to 60 minutes.

(IV) The **interictal state** which is the period between the post-ictal state and the next pre-ictal state in which the brain of the patient acts normally like a healty person.

Detection Versus Prediction

The seziure detection is considerd a classification problem between two classes, where the first class is the ictal period when a true seziure happens and the second class is the non-ictal period including post-ictal, pre-ictal and interictal periods. On the other hand, the seziure prediction is also a classification problem between two classes, but the first class is the preictal period and the second class is the non-preictal period including interictal, ictal and post-ictal periods as shown in Figure 14.

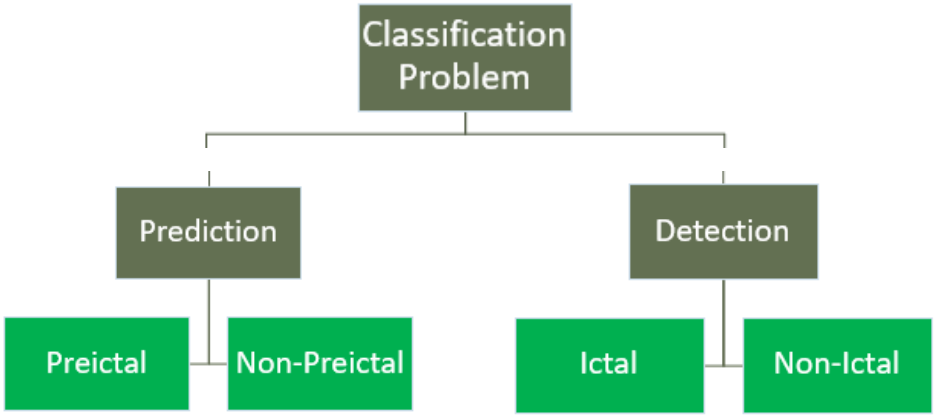


Figure 14: Prediction Versus Detection.

The interictal stage, preictal stage, ictal stage and post-ictal stage are represented respectively in Figure 13(a), (b), (c) and (d). It is clear that the ictal stage, when the seziure happens, is the most period among the other periods that can be visually recognized even by naked eye. Also, it is clearly visible that the preictal stage resemble the interictal stage in many ways such as the amplitude equavilance. That's why the prediction is way too difficult to achieve and implement more that the detection. Concequently, the prediction is more prone to error and false alarms than detection if the features extracted are not characteristic enough to the preictal period.

As we can see above that the implementaton of the prediction approach require a knowledge of the preictal stage period from the reocrded EEG data. Based on [29], the preictal period may vary from 5 minutes to 30 minutes to 120 minutes according to the patient and the seizure type with no definite time period. So, a continous EEG signal recoding is needed for prediction implementiation which is difficult to acieve with the database in hand as stated before due to its limitations in recoding continuity. In addition, the database we are using is a scalp database which will give less accurate results than intercranial data.

Project Flow

In this section, the main system idea is summarized and then our part in the whole system is explained with some details.

Whole system

The whole system for a seizure prediction/detection implantable chip that can treat the epilepsy disease by giving a stimulation pattern according to a decision function is shown in Figure 15. The system is adopted from [30] and consists of:

1) Analog to Digital (ADC) and Digital to Analog (DAC) Converters as an interface between the chip and the implantable electrodes inside the human brain. ADC is to receive the analog EEG signal and then digitize them for processing, and DAC is to convert the digital stimulation pattern to analog signal for stimulating the brain with an electric signal.
2) A storage memory for storing the digitized EEG signals in the form of window samples (epochs) to make them ready for processing.
3) The prediction/detection block for processing the EEG time samples and then classify the signal to either a seizure or non-seizure in case of detection and to either a pre-seizure or non-seizure in case of prediction. After that, a decision is taken and goes as an output in the form of a stimulation pattern.

4) The wireless interface is to communicate with the outer world and give a timely report on the status of the patient, the number of seizures suppressed and can also be used for wireless charging of the device.

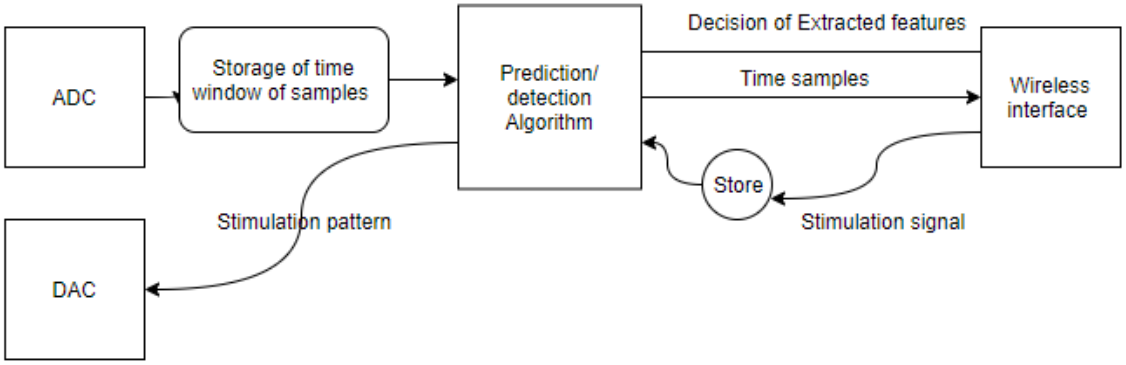


Figure 15: The whole system of seizure detection/prediction implanted in the brain

Our part

Zooming into our part in this project which is the detection block, we can see that it is divided into several steps shown in Figure 16. Firstly, the input data, raw EEG signals, are obtained from CHB-MIT Scalp EEG Database in the form of (European Data Format) file. The files are then processed and converted into readable data. Secondly, discriminant features (Hjorth, spectral power…etc.) are extracted from the signal and are used as inputs to the feature selection block. Thirdly, the extracted features are selected according to their ability to discriminate between the different periods of the signal using inspection, afterwards the selected features are used as inputs to the support vector machine trainer in MATLAB. Fourthly, the support vector machine output is regularized to decrease the number of false alarms. And finally, a decision function produces the correct class when given the trained SVM classifier and the new features. Stimulation pattern is the final output of this detection block.

Figure 16: The detection process flow.

Graduation Project Flow

The main flow of the graduation project is summarized briefly in Figure 17 into 6 main steps. First step is to survey the different machine learning techniques available and which is better for the implementation of the classification process and then decide which way this technique is going to be implemented. Second step is to start building the software blocks for the detection algorithm and then integrate these blocks into one software

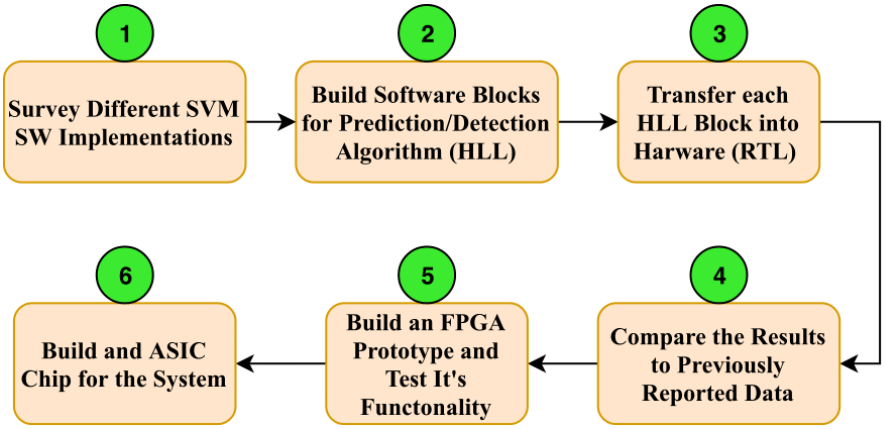


Figure 17: Main project flow.

platform. Third step is to transfer the software (HLL) blocks into hardware (HDL) blocks using hardware description language (HDL) programs. Fourth step is to compare the classification output results to previously reported data and results which in our case found in [8]. Fifth step is to build an FPGA prototype, test its functionality and determine the power consumed. Sixth and last step is to build an ASIC layout for our system ready for fabrication.

Implementation methods

The system to be implemented has different ways of integration to perform the required tasks, the different forms of implementation are shown below:

1. Training and Classification Online.

   Full system on chip (SoC) that's implantable and requires no data from outside and all processing is done inside the implanted chip.
2. Online Feature Extraction and Classification

   Training, that will be done once in a very long time, is performed on PC or FPGA outside the brain (hardware acceleration). The model parameters of the trainer are

saved into the classifier, then, classification will be done in real time inside the brain using the features extracted also from inside. This is the approach we are taking because training process needs a lot of hardware and power that is not suitable for in vivo conditions.

3. Online Feature Extraction Only.

   The implanted part are the electrodes and hardware of feature extraction only, then these extracted features are sent wirelessly via a low power transmitter to the classifier and trainer and all the classification is done outside, then the decision is sent back to the stimuli electrodes [31].

We would like to make sure of the point that our system is divided into offline parts and online parts which means that part of the system will be online (implemented in the patient's brain) and another part will be offline outside the patient's brain using hardware acceleration. The reason is that the SVM training part consumes a lot of power and hardware complexity and is better implemented offline.

Chapter 3

In this chapter, the main steps and concepts followed through the whole project are discussed in detail. In addition, the detailed work flow of the developed platform is added.

Machine Learning:

To be able to differentiate between the Ictal and non-Ictal states, we are going to employ machine learning (ML), machine learning is the field in computer science using statistical methods and artificial intelligence (AI) to continuously learn and Identify different patterns in data, trying to identify a relationship between the inputs and outputs of the system. There are two main types of machine learning: supervised machine learning and unsupervised machine learning. In supervised machine learning, the computer is provided with the inputs and outputs of the system, where the outputs are labeled so that the ML algorithm can differentiate between the labeled outputs and can map the given inputs to their correct label, for example in binary classification problems the outputs are labeled with 0 and 1 or true and false so each input is provided with its corresponding correct outcome, so each possible outcome from each set of inputs is already known, here the machine learning process is supervised by the scientist as he provides the algorithm with the correct labels. On the other hand, in the unsupervised machine learning, the scientist provides the algorithm with the inputs and outputs without labeling the outputs, so the algorithm must draw a relationship between the inputs and the outputs based on its ability to learn and figure out the patterns.

There are different machine learning techniques used for classification and regression analysis. However, given that our goal is to differentiate between ictal and non-ictal states we are facing a binary classification problem in which the output belongs only to two classes, the most commonly used algorithms in seizure detection are neural network and support vector machine due to their accuracy and robustness.

Support Vector Machine (SVM):

Support vector machine is a supervised machine learning technique that is based on statistical learning theory (SLT) It was first introduced by Vladimir Vapnik in 1979 [32] usually used in binary classification problems, given a set of training data belonging to one of two categories marked either -1 or 1, the SVM builds a model that can assign new inputs

to one of the two categories. Meaning it can generalize and assign categories to novel data from the constructed model based on the learning algorithm.

One of the advantages of the SVM is that the resolution principle of the SVM is a quadratic programming tool whose local optimal value is just its global value. Accordingly, the SVM can provide a unique solution and is a strongly regularized method that seeks a global optimized solution and avoid poor generalization problem (namely the over-fitting). However, one of the limitations of the SVM approach is that the accuracy of model depends on the choice of the defined kernel and its parameters. These parameters play a crucial role and should be optimized to yield better generalization performance.

Hard Margin:

In our binary classification problem, we have a set of inputs x_i, where x_i Is a multidimensional vector consisting of a number of m features and each vector corresponds to an output label y_i where y_i is either -1 or 1, for 2 classes of linearly separable data as shown in Figure 18. There is an infinite number of boundaries or Hyperplanes that can separate between the two classes of data where the data points on one side correspond to the +1 label and the data points on the other side correspond to the -1 label. The SVM tries to find the hyperplane with the maximum separation between the data, the points on each side which are closest to this hyperplane influence the position of this hyperplane thus they are called **support vectors** and the lines cutting these support vectors parallel to the Hyperplane are called canonical hyperplanes [33]. So, if you have a vector $\vec{w}$ perpendicular

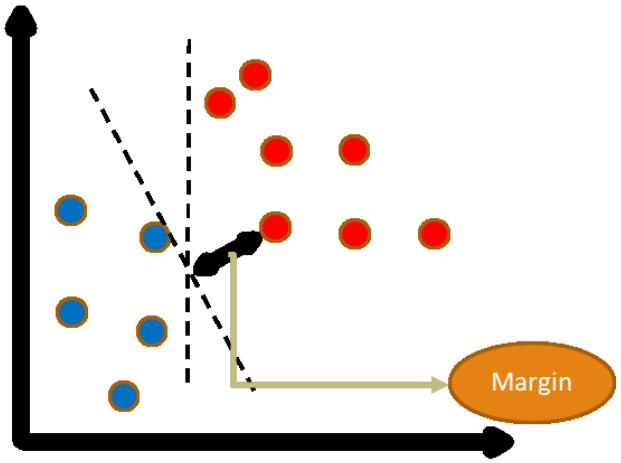


Figure 18: Two classes of linearly separable data with different hyperplanes separating between them, red is +1 and blue is -1.

to the hyper plane and a vector $\vec{u}$ going from the origin and you take the projection of the of $\vec{u}$ on $\vec{w}$, if that value is bigger than a constant then you have a positive sample, if it's smaller than that constant then you have a negative sample this is defined as the decision function.

Thus, the above formulation can be written as the follows [34]

$$f(x) = sign(\vec{w} \cdot \vec{x}_i + b) \tag{3.1.1}$$

For positive samples

$$\vec{w} \cdot \vec{x}_i + b \geq 1 \tag{3.1.2}$$

And for negative samples

$$\vec{w} \cdot \vec{x}_i + b \leq -1 \tag{3.1.3}$$

Thus, in general we can write $y_i(\vec{w} \cdot \vec{x}_i + \mathrm{b}) \geq 1$ where $y_i = +1$ for positive samples and -1 for negative samples.

To define the distance between the canonical hyperplanes, we observe that $y_i(\vec{w} \cdot \vec{x}_i + \mathrm{b}) = 1$ for the points on the canonical hyperplanes or the support vectors and $\vec{w} \cdot \vec{x}_i + \mathrm{b} = 0$ for the separating hyperplane as shown in Figure 19. So, to calculate the distance between

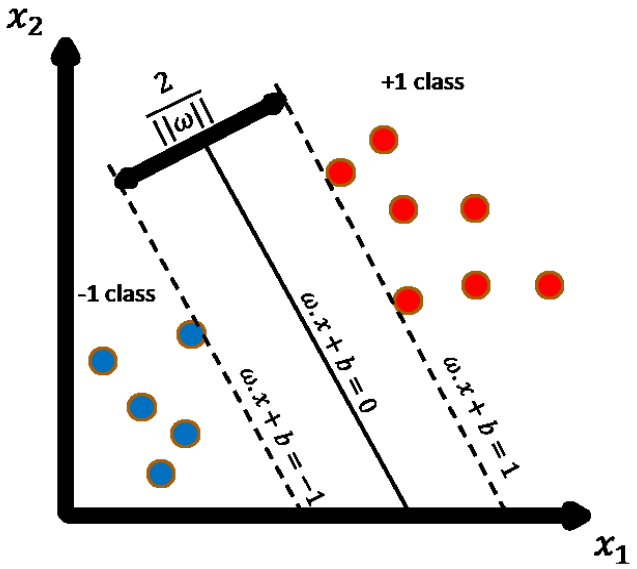


Figure 19: Hyperplanes separating between the two classes showing the bias, weights and the maximum distance.

the 2 canonical hyperplanes $(\vec{w} \cdot \vec{x}_+ + \mathrm{b}) - (\vec{w} \cdot \vec{x}_- + \mathrm{b}) = 1 - (-1) = 2$ thus $\vec{w} \cdot (\vec{x}_+ - \vec{x}_-) = 2$, where $\vec{x}_+$ is a support vector on the positive canonical hyperplane and $\vec{x}_-$ is a support vector on the negative canonical hyperplane. Normalizing this equation with respect to *w* unit vector we can write this equation as follows:

$$\frac{\vec{w}}{\|w\|} \cdot (\vec{x}_+ - \vec{x}_-) = \frac{2}{\|w\|} \quad (3.1.4)$$

This value is the margin between the canonical hyperplanes that we want to maximize, or equivalently minimize

$$\frac{1}{2}\|w\| \quad (3.1.5)$$

Subject to

$$y_i(\vec{w} \cdot \vec{x}_i + \mathrm{b}) \geq 1 \quad (3.1.6)$$

This function is a constraint optimization problem where you minimize (3.1.5) subject to (3.1.6), this can be reduced to the minimization to a Lagrange function defined as follows,

$$L(w, b) = \frac{1}{2}\left|\|\vec{w}\|\right|^2 - \sum_{i=1}^{N} \alpha_i [y_i(\vec{w} \cdot \vec{x}_i + \mathrm{b}) - 1] \quad (3.1.7)$$

This equation consists of the objective function minus the constraint multiplied by α which is the Lagrange multiplier. To minimize equation (3.1.7) we differentiate it with respect to w and b and equate the differentiation to zero:

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \quad (3.1.8)$$

And

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i(\vec{x}_i) = 0 \quad (3.1.9)$$

Thus,

$$w = \sum_{i=1}^{N} \alpha_i y_i (\vec{x}_i) \tag{3.1.10}$$

And

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.1.11}$$

Substituting back into equation (3.1.7) we get the dual formulation

$$\min_{\alpha} \sum_{i=1}^{N} \alpha_i - 0.5 \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j (\vec{x}_i . \vec{x}_j) \alpha_i \alpha_j \tag{3.1.12}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{for } \alpha_i > 0 \tag{3.1.13}$$

Thus, instead of dealing with the maximization of the margin we are minimizing the dual form of α_i. And the objective function of both forms can be shown to have the same value as there is a one to one relationship between the Lagrange multiplier, the margin, and the bias. Once α_i is calculated, we can calculate the margin as $w = \sum_{i=1}^{N} \alpha_i y_i (\vec{x}_i)$ and the bias as $b = y_i - \vec{w} . \vec{x}_i$.

Soft Margin

In the above formulation we assumed that the data are linearly separable, However, to extend the technique to non-linearly separable data as shown in Figure 20, in 1995 Cortes & Vapnik [35] introduced what's called a Loss function or a slack variable .

$$\xi_i = \max\left(0, 1 - y_i (\vec{w} \cdot \vec{x}_i + b)\right) \ \forall i \tag{3.1.14}$$

This function is zero if x lies on the correct side of the hyperplane and a value proportional to the distance from the canonical hyperplane if it lies on the wrong side of the hyperplane. Using the slack variable, we can modify the equations in the hard margin approach as follows:

$$y_i(\vec{w} \cdot \vec{x}_i + \mathrm{b}) \geq 1 - \xi_i \tag{3.1.15}$$

Now we are minimizing the margin in addition to the error subject to the following equation,

$$\min[\frac{1}{2}\|\mathrm{w}\| + \mathrm{C}\sum_{\mathrm{i}=1}^{\mathrm{N}} \xi_{\mathrm{i}}] \tag{3.1.16}$$

Where C**:** is a parameter capturing the tradeoff between having a wide margin with a small number of margin failures. Thus, the LaGrange formulation can be written as follows:

$$L(w, b, \xi, \alpha_i) = \frac{1}{2}\left|\left|\vec{w}\right|\right|^2 + C\sum_{i=1}^{N} \xi_i \ - \sum_{i=1}^{N} \alpha_i[y_i(\vec{w} \cdot \vec{x}_i + \mathrm{b}) - 1 + \xi_i] - \sum_{i=1}^{N} r_i\xi_i \tag{3.1.17}$$

Where α_i handles the constraint equation, and $r_i > 0$ handles $\xi > 0$ requirement.

The derivative with respect to w, b, ξ gives

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \tag{3.1.18}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.1.19}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - r_i = 0 \tag{3.1.20}$$

Substituting with $w = \sum_{i=1}^{N} \alpha_i y_i x_i$ in $L(w, b, \xi, \alpha_i)$ we obtain the same dual formula in equation (3.1.12), however, the equation is subject to

$$0 < \alpha_i < C \tag{3.1.21}$$

As $C - \alpha_i - r_i = 0$ and $r_i \geq 0$ and $\alpha_i \geq 0$, equation (3.1.21) is known as box constraint because it constrains alpha to be in a box with a maximum of C and a minimum of 0.

Kernel Trick

When dealing with non-linearly separable SVM it's better to transform the data into a higher dimension space in which they are linearly separable and then perform the dot product as shown in Figure 20. However, there is an easier way using the kernel trick, the kernel trick employs a kernel function that compute the dot product of the data in lower dimension space, this dot product is identical to the output of the dot product of the two

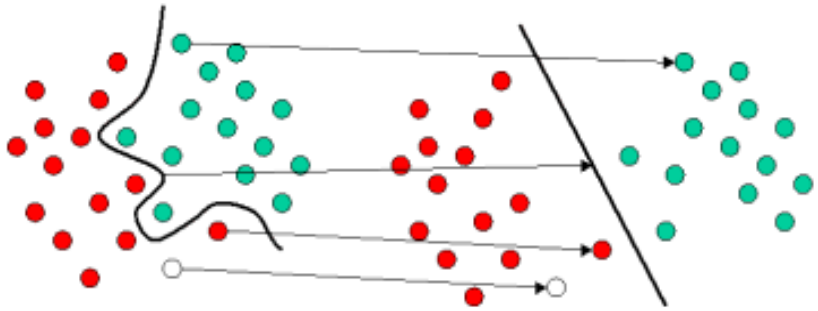


Figure 20 Non-Linearly separable data and the transformation to a higher dimension space.

data points after they are transformed to the higher dimension space.

The kernel function is formulated using the following symbol $K(x_i, x_j)$ where $K(x_i, x_j) = \Phi(\mathrm{x_i}).\Phi(\mathrm{x_j})$ where $\Phi(\mathrm{x_i})$ is the value of x_i after the transformation into the higher dimension space. There are many forms of the kernel function including but not limited to: 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 $K(x_i, x_j) = e^{-\psi\|x_i - x_j\|}$, and last but not least the Gaussian kernel or Radial Basis Function(RBF) $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma}}$. Using the kernel trick the minimization optimization problem is transformed into the following equation

$$\min_{\alpha} \sum_{i=1}^{N} \alpha_i - 0.5 \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j k(\vec{x}_i, \vec{x}_j) \alpha_i \alpha_j \qquad (3.1.22)$$

And dual formulation is represented by the following equations:

$$L(w, b, \xi, \alpha_i) = \frac{1}{2}\left|\left|\vec{w}\right|\right|^2 + C\sum_{i=1}^{N} \xi_i \; -$$

$$\sum_{i=1}^{N} \alpha_i [y_i(\boldsymbol{k}(\vec{w}, \vec{x}_i) + \mathrm{b}) - 1 + \xi_i] - \sum_{i=1}^{N} r_i \xi_i \qquad (3.1.23)$$

And the decision function or the output of the SVM is

$$f(x) = \sum_{i=1}^{N} y_i k(\vec{x}_i, \vec{x}) \alpha_i + b \tag{3.1.24}$$

To solve for the Lagrange multiplier in the optimization minimization problem and calculate the bias used in equation (3.1.24) there are several algorithms developed to calculate an optimal solution, these algorithms include but not limited to Gradient descent, gradient Ascent, Empirical risk minimization, and Sequential Minimal Optimization (SMO).

Sequential Minimal Optimization (SMO):

SMO is an algorithm proposed by Platt in 1998 [4] for training support vector machine as an alternative to the chunking algorithm proposed by Vapnik [36] and the algorithm proposed by Osuna [37]. Because SVM requires the solution of very large quadratic programming (QP) optimization problems to calculate alpha in equation (3.1.22), these QP optimization problems are time consuming and require a lot of memory, SMO breaks the large QP problem into smaller and easier to solve QP problems. It can handle very large data sets as the amount of memory required by the algorithm is linearly proportional to the training data set.

Since SMO breaks the QP problem into smaller problems, it solves for the smallest QP optimization problem that involves two Lagrange multipliers obeying the linear equality constraint. At every step, the SMO chooses two random Lagrange multipliers

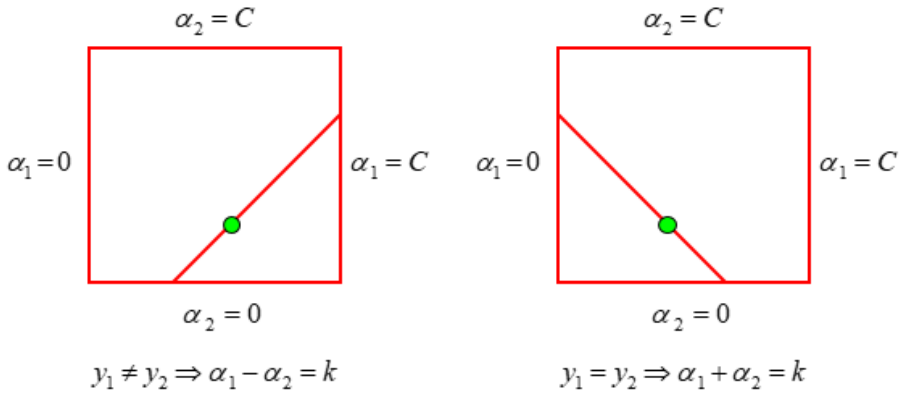


Figure 21: Lagrange multipliers obeying the box constraint and the Linear equality constraint causing them to lie on a diagonal [4].

corresponding to two values, optimize them simultaneously and update the SVM with the optimized values as shown in Figure 21. These Lagrange multipliers can be found by solving the small QP problem analytically with no need for complex numerical computations which is a great advantage of the SMO.

It should be noted that the kernel function must obey Mercer's conditions [38] in order for the QP problem to be positive definite, also the Karush-Kuhn-Tucker (KKT) [39] conditions are necessary and sufficient conditions to find an optimal solution to a QP problem that is positive definite [40]. The KKT conditions for our QP problem are:

$$\alpha_i = 0 \leftrightarrow y_i f(x_i) \geq 1 \tag{3.1.25}$$

$$0 < \alpha_i < C \leftrightarrow y_i f(x_i) = 1 \tag{3.1.26}$$

$$\alpha_i = C \leftrightarrow y_i f(x_i) \leq 1 \tag{3.1.27}$$

Calculating the Two Lagrange Multipliers:

The SMO algorithm calculates the constraints for the two Lagrange multipliers then solves for the minimum of the constraint, the box constraint causes the Lagrange multipliers to be bound in a box and the Linear equality constraint causes the two Lagrange multipliers to lie on a line as shown in Figure 21.

The algorithm first calculates the second Lagrange multiplier α_2 then solves for the bounds of the diagonal line segment inside the box constraint in terms of this Lagrange multiplier, if $y_1 \neq y_2$ then it applies the following bounds

$$L = \max(0, \alpha_2 - \alpha_1), H = \min(C, C + \alpha_2 - \alpha_1) \tag{3.1.28}$$

If $y_1 = y_2$ then the following bounds apply

$$L = \max(0, \alpha_2 + \alpha_1 - C), H = \min(C, \alpha_2 + \alpha_1) \tag{3.1.29}$$

We can express the second derivative of the objective function along the diagonal by

$$\eta = K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) - 2K(\vec{x}_1, \vec{x}_2) \tag{3.1.30}$$

The objective function will be positive definite under normal conditions and we can calculate a minimum along the diagonal linear equality constraint and η will be >0. Thus, the minimum is calculated to be

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta} \tag{3.1.31}$$

Where $E_i = f(x_i) - y_i$, then the constrained minimum is calculated by clipping the unconstrained minimum via the following set of equations

$$\alpha_2^{new,clipped} = \begin{Bmatrix} H \; if \; \alpha_2^{new} \geq H \\ \alpha_2^{new} \; if \; L < \alpha_2^{new} \leq H \\ L \; if \; \alpha_2^{new} \leq L \end{Bmatrix} \tag{3.1.32}$$

To calculate α_2^{new} form the clipped alpha,

$$\alpha_2^{new} + \alpha_1 + y_1 y_2(\alpha_2 + \alpha_2^{new,clipped}) \tag{3.1.33}$$

Under normal conditions η will be positive, however if the kernel function does not obey Mercer's conditions, it will introduce a negative η which may lead to an indefinite objective function. If the same vector x is common between more than one training sample, η may be zero even with a kernel function obeying Mercer's conditions. The SMO algorithm computes the objective function through the following set of equations even if η is negative

Let $s = y_1 y_2$

$$f_1 = y_1(E_1 + b) - \alpha_1 K(\vec{x}_1, \vec{x}_1) - s\alpha_1 K(\vec{x}_1, \vec{x}_2), \tag{3.1.34}$$

$$f_1 = y_2(E_2 + b) - s\alpha_1 K(\vec{x}_1, \vec{x}_2) - \alpha_2 K(\vec{x}_2, \vec{x}_2), \tag{3.1.35}$$

$$L_1 = \alpha_1 + s(\alpha_2 - L) \tag{3.1.36}$$

$$H_1 = \alpha_1 + s(\alpha_2 - H) \tag{3.1.37}$$

$$\Psi_L = L_1 f_1 + L f_2 + \frac{1}{2} L_1^2 K(\vec{x}_1, \vec{x}_1) + \frac{1}{2} L^2 K(\vec{x}_2, \vec{x}_2) + s L L_1 K(\vec{x}_1, \vec{x}_2) \tag{3.1.38}$$

$$\Psi_H = H_1 f_1 + H f_2 + \frac{1}{2} H_1^2 K(\vec{x}_1, \vec{x}_1) + \frac{1}{2} H^2 K(\vec{x}_2, \vec{x}_2) + s H H_1 K(\vec{x}_1, \vec{x}_2) \quad (3.1.39)$$

The bias b is calculated at each step, to calculate b_1 the following equation is used which is valid when its Lagrange multiplier is not at the boundaries as it forces the SVM output to be y_1 if the input value is x_1

$$b_1 = E_1 + y_1(\alpha_1^{new} - \alpha_1)K(\vec{x}_1, \vec{x}_1) + y_2\left(\alpha_2^{new,clipped} - \alpha_2\right)K(\vec{x}_1, \vec{x}_1) + b \quad (3.1.40)$$

Similarly, b_2 is computed

$$b_2 = E_2 + y_1(\alpha_1^{new} - \alpha_1)K(\vec{x}_1, \vec{x}_2) + y_2\left(\alpha_2^{new,clipped} - \alpha_2\right)K(\vec{x}_1, \vec{x}_2) + b \quad (3.1.41)$$

Both of these values are equal if the Lagrange multiplier α_1 & α_2 are at the boundaries and $L \neq H$, in such case, the SMO algorithm choses the bias to be $\frac{b_1+b_2}{2}$.

A special case when dealing with Linear kernel SVM, instead of updating the margin w every time instead of all the zero values corresponding to non-zero Lagrange multipliers

$$\vec{w}^{new} = \vec{w} + y_1(\alpha_1^{new} - \alpha_1)\overrightarrow{x_1} + y_2\left(\alpha_2^{new,clipped} - \alpha_2\right)\overrightarrow{x_2} \quad (3.1.42)$$

Heuristics for choosing the Lagrange multipliers:

The algorithm employs two different heuristics for each Lagrange multiplier. The first one is for choosing the first multiplier, it iterates over the whole training set to determine whether each training example violates the KKT conditions or not, thus, this heuristic defines the outer loop of the algorithm, if a training example violates the KKT conditions then it can be chosen for optimization, after one iteration through the entire training set, the outer loop then iterates over the examples whose Lagrange multipliers are not 0 or C, meaning that they are not boundary multipliers, then it chooses the examples that violates the KKT conditions, this approach is repeated until all the non-bound examples obey the KKT conditions within a threshold ε which is usually chosen to be 10^{-3}, then the outer loop goes back to the entire training set not just the non-boundary training examples, the

outer loop keeps alternating between the entire data set and the non-bound training examples until all the training examples obey the KKT conditions.

In the second heuristic, the second Lagrange multiplier is chosen in a way that maximizes the step size taken in simultaneous optimization, the step size is approximated to be $|E_1 - E_2|$, in some cases the SMO algorithm progress cannot be made like when the two training examples have the same input x, this case causes the objective function to become semi-definite rather than positive definite, thus, the SMO algorithm searches in the non-boundary Lagrange multipliers training examples until a positive progress is made, if the non-bound examples doesn't satisfy the requirement of positive progress then it searches through the entire training examples, the iterations in the non-bound and entire examples start at random locations, if none of the non-bound examples or the entire example set allow the SMO algorithm to advance in a positive progress the example chosen in the first heuristic is skipped and the next example is evaluated and the entire process is repeated.

Pseudocode for the Algorithm:

The pseudo code is adapted from [4] and found in Appendix I.

Flowcharts for SMO Algorithm

An abstract of the flowchart followed by a detailed flowchart are shown in Figure 22 and Figure 23 respectively.

Artificial Neural Network:

The second most common machine learning method used in epileptic seizure detection is Artificial neural network (ANN). ANN were inspired from the biological neurons and proven to be a powerful tool for learning by constructing a nonlinear mapping between a given set of input and output data [41]. The ANN approach is a well-known classification and regression method, due to its inherent pattern recognition capabilities and its ability to handle noisy data [42]. The patterns are represented in the connection weights between the network layers that are updated during the learning process based on the backpropagation error signal that represents the difference between the desired output and the actual outputs of the network.

The main limitation of the ANN is that these networks are not so great when it comes to knowledge presentation. The user can't extract the knowledge from the weights that are

distributed throughout the whole network. This makes neural network like a black box to the user. Another drawback is tedious parameter tuning of the network structure. Moreover,

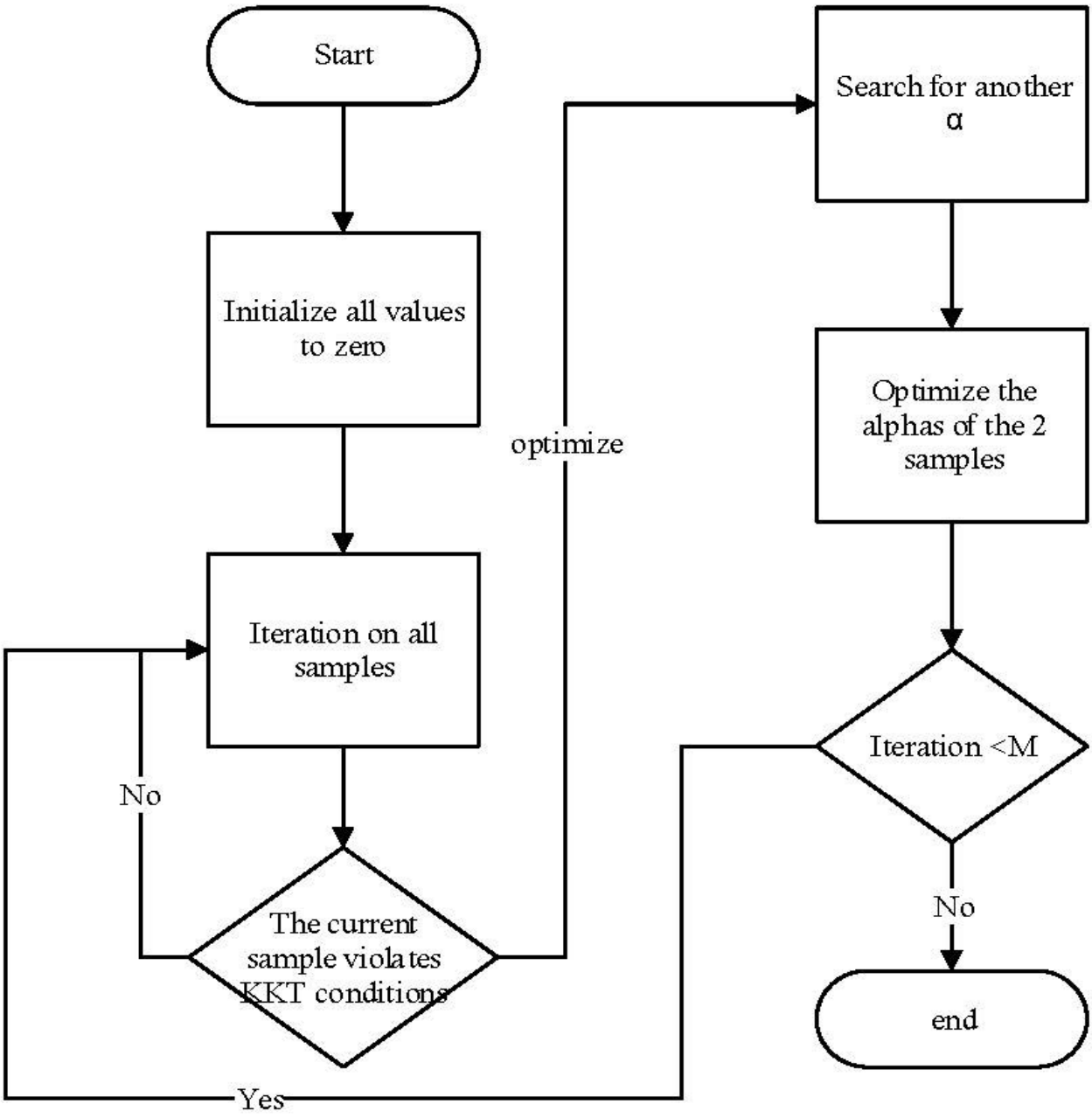


Figure 22: Flowchart overview for SMO algorithm.

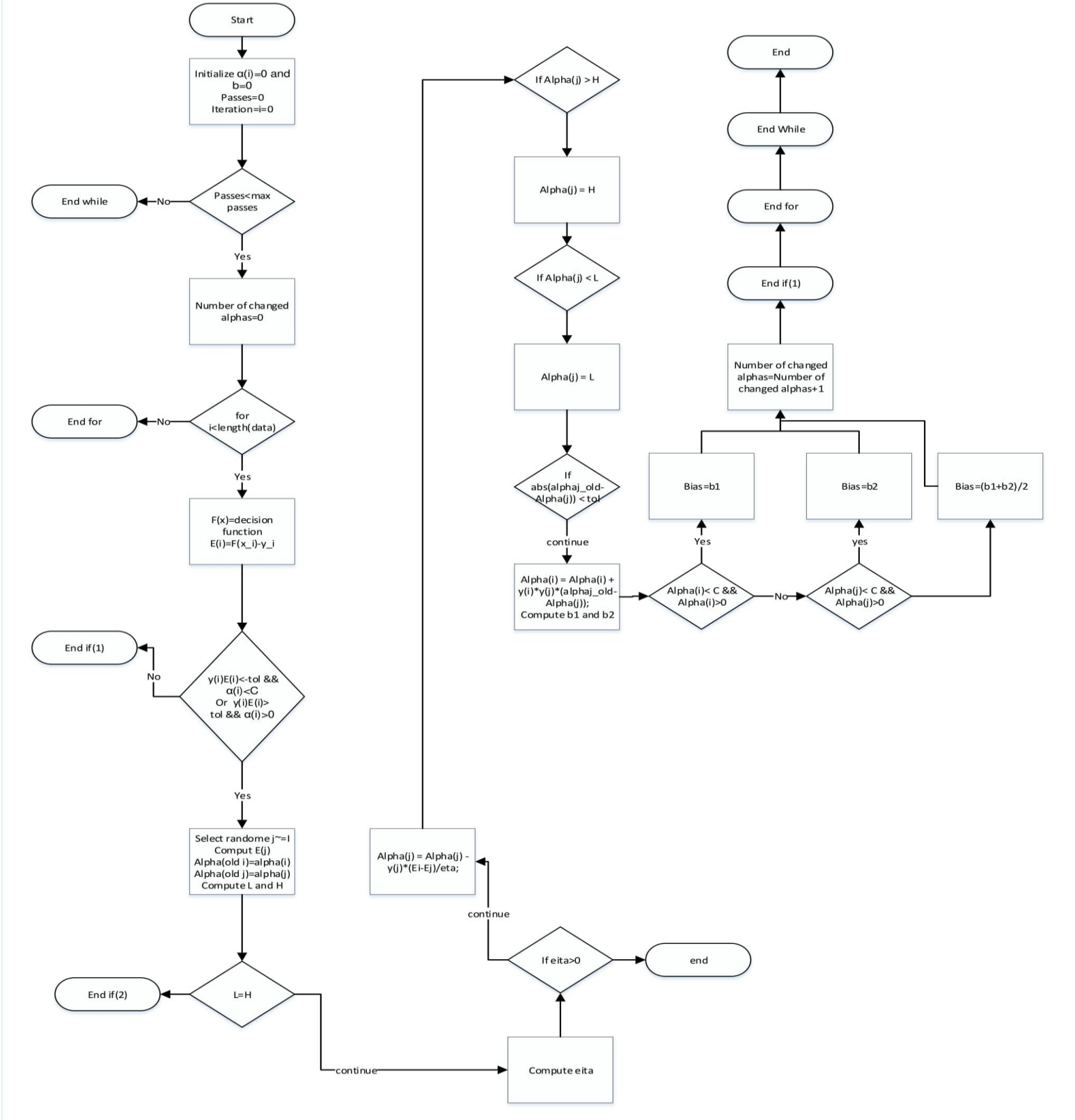


Figure 23: SMO detailed flowchart.

the ANN algorithm is based on the principle of empirical risk minimization, and this can lead to local minima and hence poor minimization performance.

A typical ANN consists of a set of neurons that receive a value from an input, according to this value it alters its internal state or the activation value and produces an output depending on the input and the activation function. A neuron ***a*** typically has an activation value denoted by $a_j(t)$, it has a threshold value θ_j and an input $x_j(t)$, this neuron can compute its activation value at the next discrete time interval via an activation function, where, $a_j(t+1) = f(a_j(t), x_j(t), \theta_j)$. In addition to the neurons, an ANN consists of a set of connections from neuron i to neuron j where each connection has a weight denoted by w_{ij}. To compute the input to neuron j from the output of neuron i a propagation function is used $p_j(t) = \sum_i O_i(t) w_{ij}$ where $O_i(t)$ is the output of neuron i. The ANN learns by randomly varying the number of neurons, the number of hidden layers, the weights and the thresholds to map a set of inputs to an output with the least error value.

A nonlinear weighted sum of the network can be represented by $f(x) = K * \sum w_i g_i(x)$ where $g_i(x)$ is a composition of other function like the propagation, K is usually denoted by activation function, typical activation functions that are widely used are the hyperbolic tangent, the sigmoid function, the rectifier function and the SoftMax function.

In the learning process, the function $f(x)$ tries to map the inputs to the output, this is achieved by applying what is called by a cost function $C = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$ where N is the number of samples and y_i is the target output, Figure 24(a) shows a typical neural network with only one hidden layer and one output while Figure 24(b) shows a neural network with multi hidden layer perceptron and multiple output values for a multiclass artificial neural network problem.

Different types of ANNs are used in classification and regression problems, including but not limited to feedforward neural network, backpropagation neural network and convolutional neural network, these networks are used depending on the application.

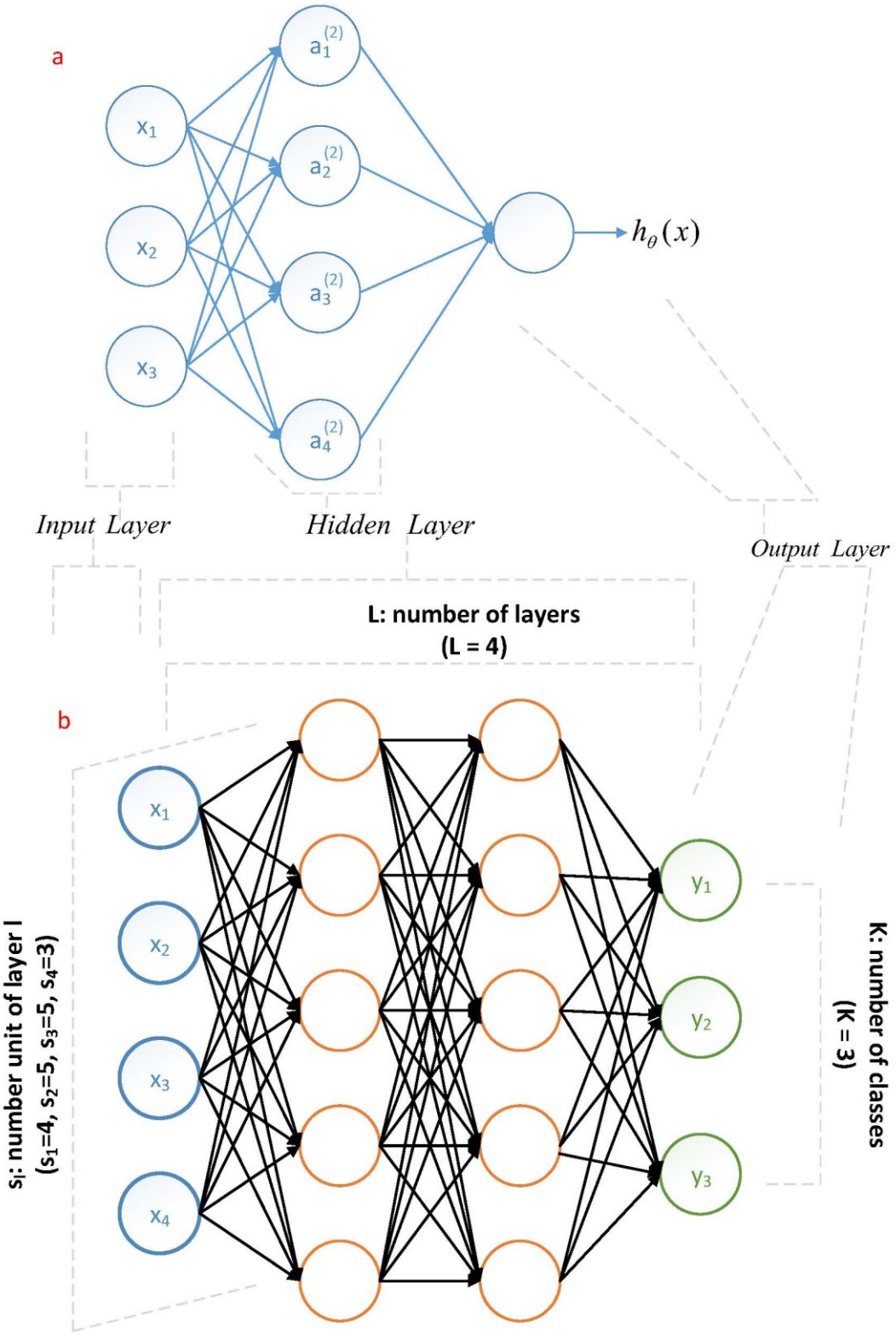


Figure 24 a) single layer perceptron with only one output, b) multilayer perceptron with multiple outputs.

Features

Feature Extraction

To be able to differentiate between the different categories of the input signal and allow the SVM algorithm to differentiate between the seizure (ictal or preictal in prediction) and non-seizure (non-ictal) states, raw signal data is converted into a set of distinctive features via feature extraction. Through these features, the model will predict the occurrence of a seizure if the features extracted are similar or comparable to the features of the ictal period in the training data set. Throughout the project, we investigated many types of features in order to get the optimal set of features that offer a reasonable hardware complexity while maintaining the required performance.

Features are can be classified into two categories, frequency domain features and time domain features. Table 1 provides a summary for the features adopted in our study. Features may be applied on raw signal data coming from the EEG or after formation of epochs.

Table 1: Time and frequency domain features used.

Frequency Domain	Time Domain	
- Power spectral density using filters for different bands. - Power spectral density using Fast Fourier Transform (FFT)	- Coast Line	$x(i+1)-x(i)$
	- Energy	$E = x^2$
	- Variance	$\sigma^2 = \frac{\sum(x-\mu)^2}{N}$
	- Skewness	$k = \frac{E(x-\mu)^3}{\sigma^3}$
	- Kurtosis	$k = \frac{E(x-\mu)^4}{\sigma^4}$
	- Hjorth Mobility	$HM = \sqrt{\frac{var\left(\frac{dx(t)}{dt}\right)}{var(x(t))}}$

	- Hjorth Complexity	$HC = \frac{HM\left(\frac{dx(t)}{dt}\right)}{HM(x(t))}$

Frequency Domain Features

Literature identified certain frequency components in EEG signals that appear the most during preictal period which are five known bands: delta-band (0.5–4 Hz), theta-band (4–8 Hz), alpha-band (8–13 Hz), beta-band (13–30 Hz) and gamma-band (30–128 Hz) were reported to be useful in predictive models as the power is transferred from low frequency components to high frequency components during preictal period [43]. For ictal period, frequencies up to 20 Hz are widely considered [31]. There are two methods to extract the spectral powers of these bands explained below.

i. Using filters

This approach places several band-pass filters as shown in Figure 25 centered around different frequencies. The output of the filters is accumulated as shown

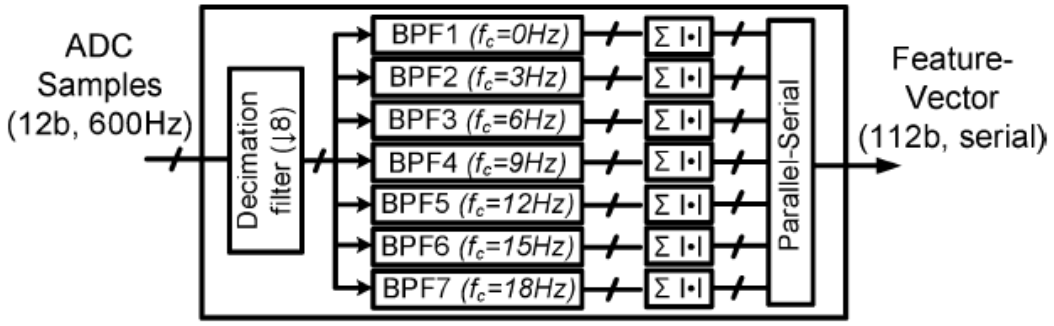


Figure 25: The filter bank followed by magnitude summation to calculate the spectral power of certain bands.

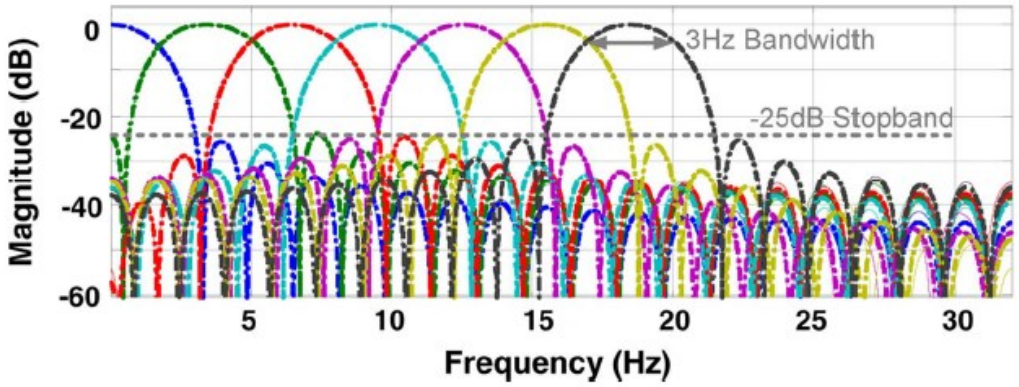


Figure 26: Specifications of the filter bank.

in Figure 26. Typical filters are Type-I FIR filters whose orders may reach 50 [31]. Intuitively, such bank of filters will be very power hungry when implemented.

ii. Relative spectral power using Fourier transform

$$P_i = \sum_i PSD(x), \quad np_i = \frac{P_i}{P_{tot}} = \frac{\sum_i PSD(x)}{\sum_{tot} PSD(x)} \tag{3.2.1}$$

iii. Where p_i is the spectral power of i^{th} sub-band, x is the windowed raw EEG signal, i indexes the i^{th} frequency sub-band, and *PSD(x)* is the Power Spectral Density of the signal. The normalized spectral power feature for a given sub-band was computed by dividing the spectral power of the sub-band by the total power as in (3.2.1). This implementation will need an FFT core to compute *PSD(x)*.

Time domain features

This type of features is obtained by performing direct time domain operations on the raw EEG signal data or after windowing.

i. Coast Line

It's one of the simplest features that can be obtained yet it proved efficient. It aims at finding the absolute difference between each two samples $(x\ (i), x\ (i+1))$ then windowing those differences into the required epoch length.

$$x(i+1) - x(i) \tag{3.2.2}$$

ii. Hjorth Energy

Another simple and easy way to extract a feature is the Hjorth energy feature which simply gets the summation of the squares of data points, or even time epochs.

$$Energy = \sum_i x(i)^2 \tag{3.2.3}$$

Where $x(i)$ can be a data sample or a windowed epoch.

iii. Hjorth Mobility (HM)

The mobility parameter represents the mean frequency of the power spectrum of the signal x(t).

$$HM = \sqrt{\frac{var\left(\frac{dx(t)}{dt}\right)}{var\big(x(t)\big)}} \tag{3.2.4}$$

iv. Hjorth Complexity (HC)

It represents the frequency variation in each signal x(t), where x(t) is a window from the raw data.

$$HC = \frac{HM\left(\frac{dx(t)}{dt}\right)}{HM\big(x(t)\big)} \tag{3.2.5}$$

v. Variance

Variance describes how the data points in a sample or a population deviate from their mean value. So, if the variation is large, so does the EEG fluctuations which resembles the high rhythmic activity during a seizure (ictal period).

$$\sigma^2 = \frac{\sum(x-\mu)^2}{N} \tag{3.2.6}$$

Where μ is the sample mean and N is number of data points in the sample.

vi. Skewness

Skewness is a measure of the asymmetry of the data around the sample mean.

$$k = \frac{E(x-\mu)^3}{\sigma^3} \tag{3.2.7}$$

Where $E(x)$ is the expected value of x.

vii. Kurtosis

Kurtosis is a measure of how outlier-prone a distribution is and also how the distribution deviates from the normal bell-shaped distribution as shown in Figure 27.

$$k = \frac{E(x-\mu)^4}{\sigma^4} \tag{3.2.8}$$

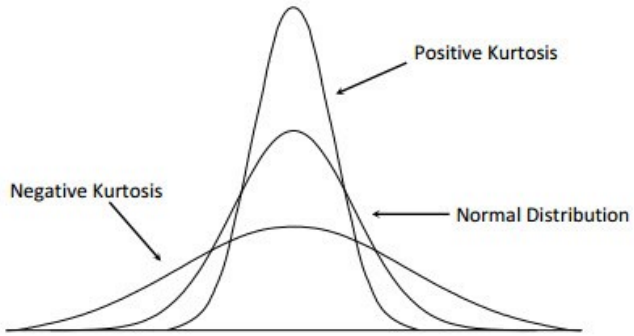


Figure 27 Kurtosis meaning

Online Algorithm for Statistical Moments

For the statistical features (variance, skewness, kurtosis), they depend on the sample mean so we need to wait for the required samples to calculate them. That would introduce large latencies in the hardware implementation of these features, so we need online algorithms that update the value of the produced quantity for every new data point.

Terriberry proposed an algorithm [44] that calculates the statistical moments, M_k, using online incremental algorithm shown below.

$$M_k = E\left[x - E[x^k]\right] \tag{3.2.9}$$

$$\delta = x - m$$

$$m' = m + \frac{\delta}{n}$$

$$M_2' = M_2 + \delta^2 \frac{n-1}{n}$$

$$M_3' = M_3 + \delta^3 \frac{(n-1)(n-2)}{n^2} - \frac{2\delta M_2}{n}$$

$$M_4' = M_4 + \delta^4 \frac{(n-1)(n^2-3n+3)}{n^3} + \frac{6\delta^2 M_2}{n^2} - \frac{4\delta M_3}{n}$$

Skewness

$$g_1 = \frac{\sqrt{n}M_3}{M_2^{3/2}}$$

Kurtosis

$$g_2 = \frac{nM_4}{M_2^2} - 3$$

The pseudocode for the online algorithms is found in Appendix I.

This is an incremental (online) algorithm proposed by Terriberry to obtain the crucial central moments needed to calculate mean, variance, skewness and kurtosis. This the algorithm we used in writing the hardware for these features and is explained further in the hardware section.

Normalization

Because there are no limits on the voltage levels of the EEG signal and data may fluctuate rapidly at a high voltage level (or low) that will be considered as a whole within a certain class by the classifier, so testing and training data must be normalized using exactly the same steps. We tested three different methods of normalization.

a. Standard Score

$$y = \frac{x - \mu}{\sigma} \tag{3.2.10}$$

The mean of the training data is subtracted from every point of training data and testing data then divided by the standard deviation of training data. Note that we cannot use the mean or standard deviation of testing data on normalizing the testing data because we assume that we don't have all the testing data as it enters the system and got classified in real time.

b. Feature Scaling

This type of normalization scales all the data between to levels a and b.

$$y = a + \frac{(x - x_{min})(b - a)}{x_{min} - x_{min}} \tag{3.2.11}$$

In order to confine the data in the rang [0,1] we use $a = 0$ and $b = 1$.

c. Mean of the Absolute.

$$y = \frac{x - mean(X)}{mean(|X|)} \qquad , x \in X \tag{3.2.12}$$

Normalization is also done on raw data in order to reduce the number of bits needed in the fixed-point system we used in hardware.

Feature and Channel Selection

It is reported that 60% of the patients suffer from epileptic seizures that are related to a specific region of the brain which is temporal lope [45]. So, multivariate features which explore the spatial dependence is chosen to be able to characterize differences in the EEG signal from the different regions of the brain leading to a better prediction performance. The bivariate approach is used on the relative spectral power features of the 5 bands mentioned above obtained for each channel of the 23 channels used to read the EEG signal. It was implemented using the method found in [46]. For mono variate features, we selected six channels manually that came from the most influenced areas of the brain.

Smoothing

Due to the roughness and the great variations and artifacts in EEG data, it is suitable to smooth the data by using a moving window average (3.2.13). So, each epoch amplitude now depends on the previous epochs, hence no sudden change will happen as shown in Figure 28.

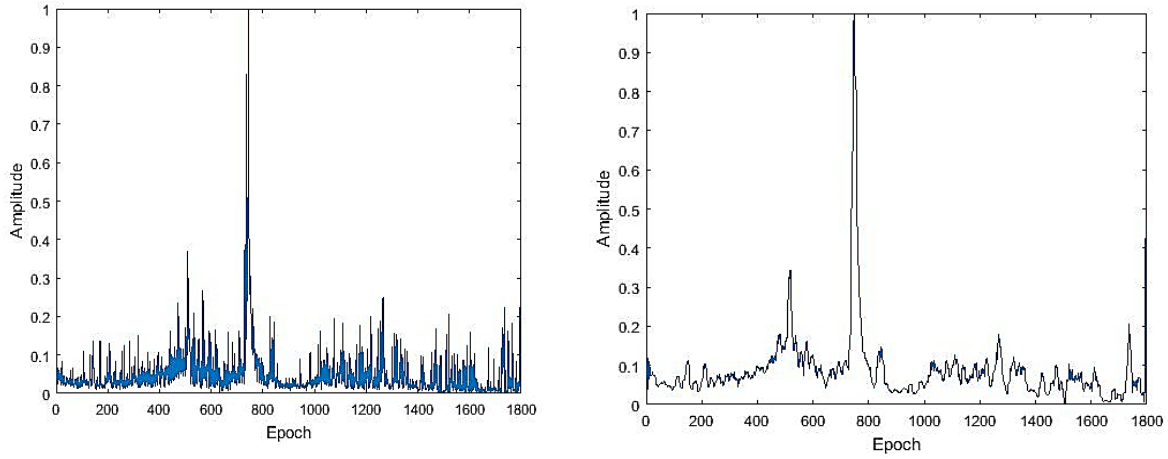


Figure 28: Before smoothing (Left). After of smoothing (Right).

$$\bar{a}_k = \frac{1}{m+1}\sum_{j=0}^{m} a_{k-j} \tag{3.2.13}$$

Performance Metrics:

Measuring the performance of the model developed by the machine learning algorithm is critical to be able to compare it to other models developed by other researchers and to reach the best model that can produce the best results compared to the actual data. There are two main metrics used to measure the model's performance, the epoch-based metric and the event-based metric [47].

Epoch-Based Metric

This approach doesn't depend on the application of the machine learning algorithm, but rather it assesses the performance of the output based on the individual epoch. In classification problems, each epoch is compared to the corresponding epoch in the actual class, for example binary classification problems in which the output of the machine learning algorithm is either 0 or 1 (true or false), the classifier output can be represented by a confusion matrix as shown in Figure 29. The confusion matrix consists of four categories, true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

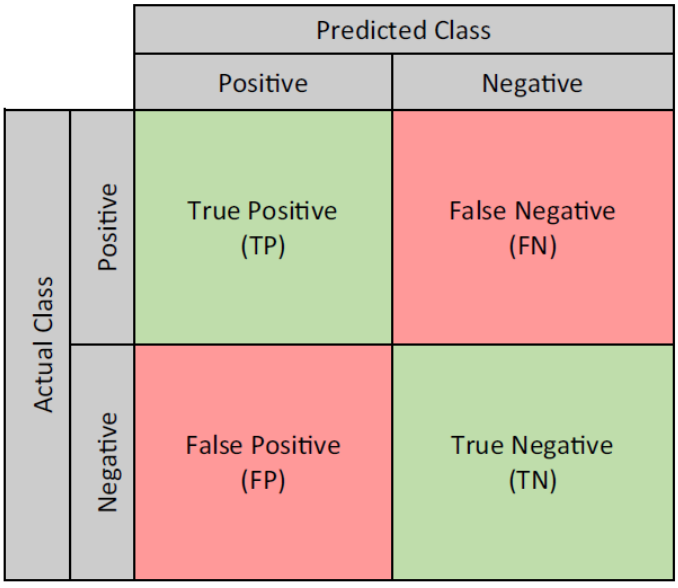


Figure 29: Binary classification confusion matrix [6]

Where TP represents the epochs labeled true when the actual value is also true, TN represents epochs labeled false when the actual output is also false, FP represents epochs labeled true when the actual output is false, and finally FN represents epochs labeled as false when the actual output is true, these values are used to calculate other metrics that are widely used in performance metrics, namely the sensitivity, the accuracy, the specificity and the false detection rate (FDR). The sensitivity is defined as $\frac{TP}{TP+FN}$ which represents the ratio of the correctly identified seizure -in the case of seizure detection- to the correctly identified epochs, the specificity is defined as $\frac{TN}{TN+FP}$ which represents the truly identified non-seizures from all the non-seizures or zero classification, the FDR represents the amount of false identified seizures per hour, and the accuracy is defined as $\frac{TP}{TP+FP}$ which represents the rate of correct seizure epochs from the epochs identified as true .

Another metric used in epoch-based performance metric is the receiver operator characteristic (ROC) curve as shown in Figure 30, which shows how the sensitivity changes with respect to the specificity, the area under the curve can be used to compare between the performance of different algorithms, where a random classification will lead to an area of 0.5 and a perfect classification will produce an area of 1.

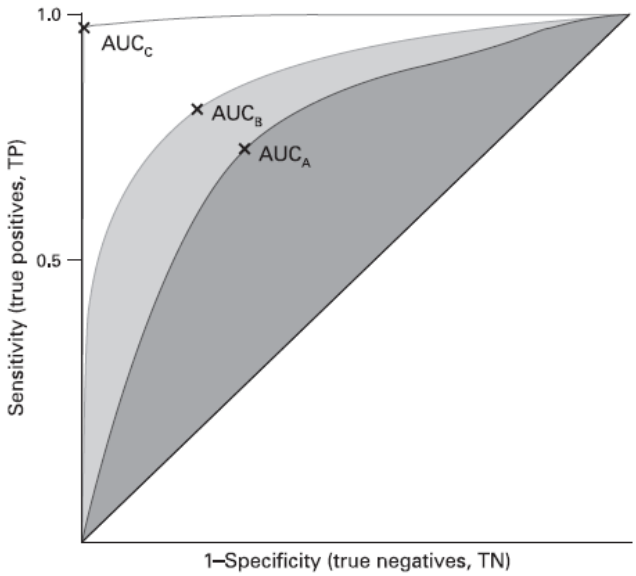


Figure 30: Figure 30: ROC curve [7]

Event-Based Metric:

An application specific metric used in seizure detection performance assessment, it doesn't judge the validity of the classifier based on the epoch only but rather a combined decision of subsequent epochs which lead to an event, in seizure detection two metrics are taken into account, good detection rate (GDR) and false detection rate (FDR). A seizure is marked as detected if it was detected at any time between the start and the end of the seizure, a GDR is the number of detected seizures compared to the total number of seizures. Another metric is introduced specific for seizure detection shown in Figure 31 is latency which is defined as the difference between the time when the seizure was detected and the actual start of the seizure as shown in Figure 31, FDR is defined as the number of false detected seizures per hour, in a non-seizure or ictal free hour.

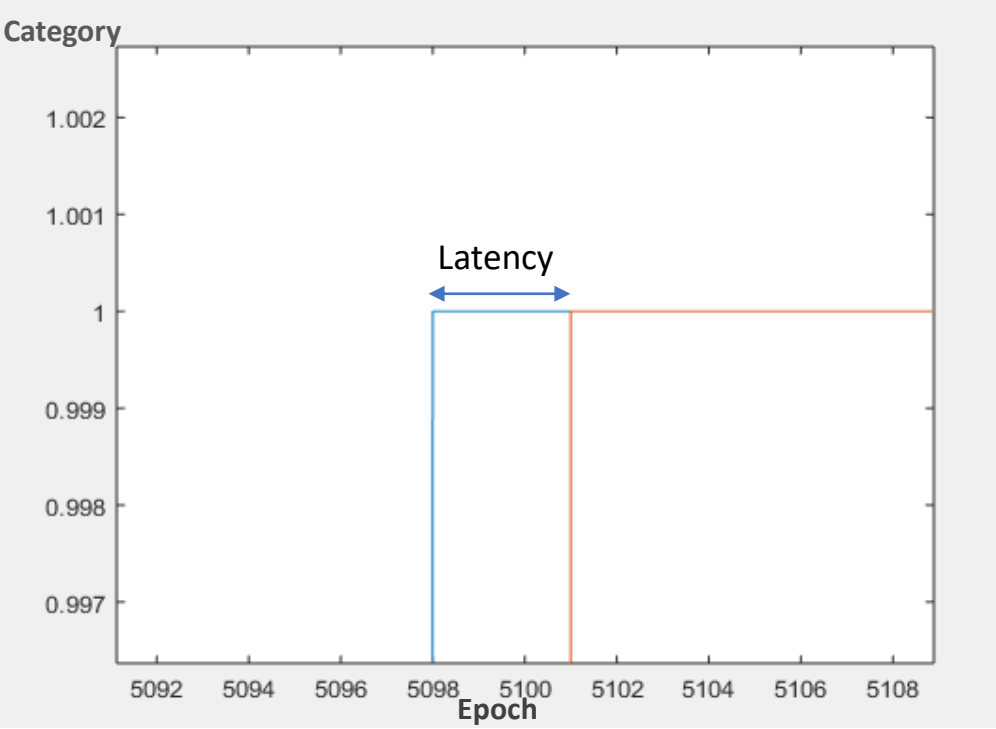


Figure 31: Latency

Throughout our work we are going to employ the event-based metric, reporting the number of correctly identified seizures, the latency and the false detection rate.

Seizure Detection workflow

The patients' files are organized in European data format (EDF) files, these files contain the EEG signals documented and organized with hours containing seizure and seizure-free hours. We start by reading the patients files in MATLAB using "Readedf" function to convert the EDF files to mat format which can be processed by MATLAB, then we separate the seizure hours from the seizure-free hours, then we mark the onset of the seizure and the ending of the seizure in the hours containing seizures. We then we start by extracting the features from a number of hours containing seizures and seizure free hours to give the trainer a wide set of training data to produce unbiased results, each hour contains a matrix $23 * 921600$, each row corresponds to a channel and each column corresponds to an hour. Given that the data are sampled at 256 sample per second, the matrix is divided by 256 to convert it into seconds and they are further converted into epochs each epoch is 2 second long.

Six discriminant channels are chosen from the 23 channels, thus, the input raw data from which the features will be extracted is a $6 * 1800$ matrix with 1800 epochs from 6 channels, we then extract each feature from the 6 channels. The output feature matrix is summed and divided by the number of features to calculate an average feature from the 6 channels, then these features are normalized using the min max normalization discussed above, thus, the final input matrix to the trainer is the number of features*number of hours*1800 where the number of features represent the columns and 1800*number of hours represent the rows of the matrix.

The input to the SVM trainer is the extracted features and a label matrix created by marking the beginning and ending of the seizures, a set of parameters are tuned in the SVM trainer to produce the best classifier that can discriminate between ictal and non-ictal states in an input test vector.

To assess the performance of the classifier we first generate a test input matrix from the hours of the same patient that were not used in the training, we produce the test matrix the same way the training input matrix is generated, also we generate the labels which mark the ending and beginning of the seizure the same way as the input labels are generated

where the seizure epochs are marked by 1 and the seizure free epochs are marked by 0 or -1.

The input to the SVM classifier are the test hours and the actual labels and the output of the classifier are the labels generated by the classifier to categorize the seizures.

The output from the classifier and the actual classification matrices are compared against each other to assess the performance of the classifier according to our performance metrics.

The total workflow is summarized neatly and briefly in Figure 32.

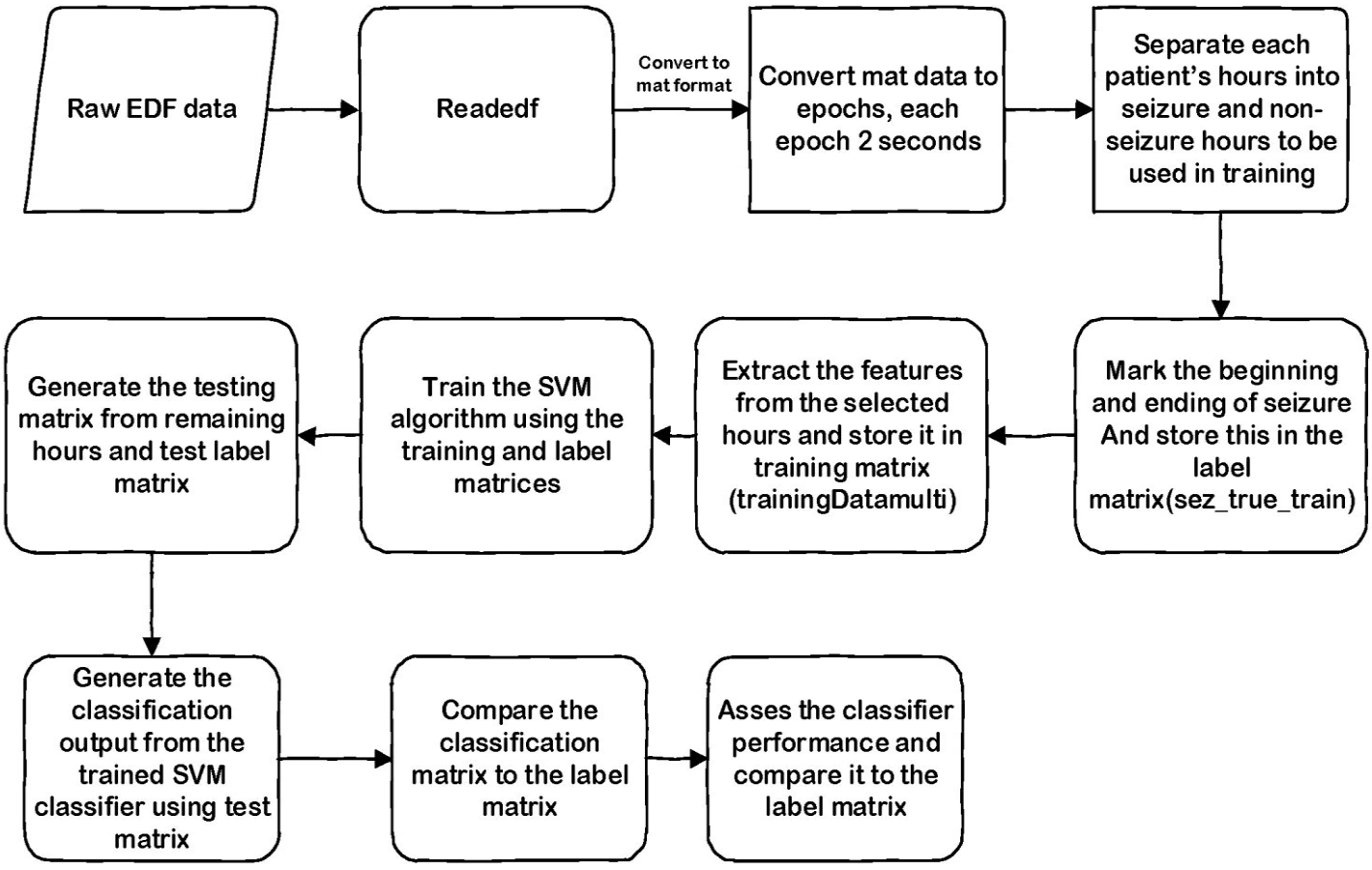


Figure 32: MATLAB workflow

Chapter 4

In this chapter a detailed explanation of the hardware techniques used is presented. In addition, the hardware blocks developed so far are presented in the form of neat block diagrams. The designed ASIC chip layout and power consumption estimations are also presented in this chapter.

CORDIC Approximation

One of the most crucial parts in the SMO SVM is the kernel function, due to the high cost of the RBF-kernel, most literatures resort to linear kernel functions as they can be implemented by just a multiplier accumulator (MAC). The problem with RBF-kernel is that it contains exponential function which is not straight forward to be implemented as a hardware RTL. A mathematically efficient and accurate method called CORDIC (Coordinate Rotation Digital Computer) is used to calculate many elementary functions such as trigonometric, hyperbolic and exponential functions. It needs only a shifter and an adder in addition to a small look up table.

It was first developed by Jack Volder in 1959 [48] as a new method for computing trigonometric functions in digital applications using only add and shift function in addition to a look up table.

An example of using the CORDIC algorithm in the rotational mode for computing a trigonometric function like the Sine or Cosine of an angle in radian is shown below in Figure 33.

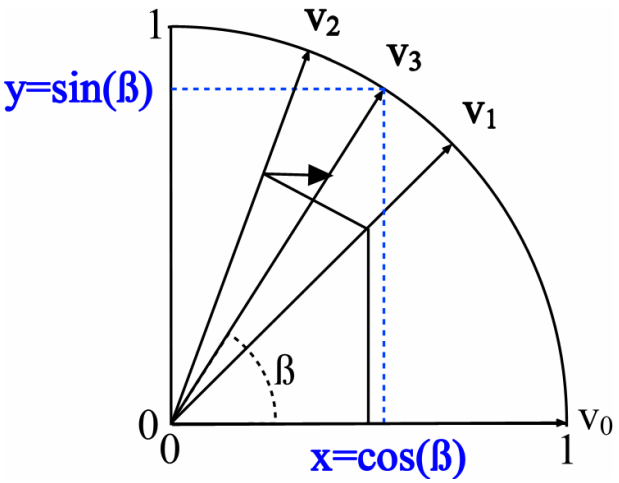


Figure 33: Angle rotation.

Starting with an arbitrary value for x and y, $x_0 = 0$ and $y_0 = 1$ and a vector $v = \begin{matrix} x \\ y \end{matrix}$, where $v_0 = \begin{matrix} 1 \\ 0 \end{matrix}$ like binary search we are searching for our angle β. We start by rotating the vector by 45° which is half the angle between y and x, then we compare the vector's angle to β if β is less than 45, after that we rotate the vector by half 45 to the left, if β is more than 45, then we rotate the vector by half 45 to the right and then compare β to the new angle and repeat the same procedure; rotate by half the new angle to the left or the right until the rotated angle of the vector $v = \beta$ to a certain tolerance, at this point the x value of the vector represents $\cos(\beta)$ and the y value represents $\sin(\beta)$. The angles rotation could be stored in a look up table and we can further improve it by making the angles correspond to multiples of 2 so that the multiplication or division is reduced to a simple shift, a typical look up table is as follows $45, 26.565, 14.036$ which corresponds to $1, 0.5, 0.25$ and so on.

The above algorithm could be represented by the following equation:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = A_i \begin{bmatrix} 1 & \sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix}$$

Where σ is either positive or negative one according to the direction of rotation, and A is a constant that approaches 0.60725 for very large i. We can compute $exp(x)$ as $cosh(x) + sinh(x)$ that are computed in a similar way as sin and cos [49]. Another approach for computing $y = exp(x)$ is described below.

First, we choose $k < x$ such that $exp(k)$ is a "nice number";

$exp(-k)\ y = exp(x - k)$ increment k till $x - k = 0$

$exp(-x)\ y = exp(x - k) = exp(0) = 1$

In other words, when we subtract k from x, we multiply y by $exp(k)$, where $exp(k)$ will be chosen to be easily multiplied that will be in the form of $2^n\ or\ 1 \pm 2^n$, n is integer so that the multiplication is done easily using add 1 and shift. (e.g. = 2, 3/2, 33/32…etc.).

Example:

let's calculate y = $exp(5)$:

Figure 34: Look up table(LUT) for exponential CORDIC.

Initially: y = 1, x=5;

$x = 5 - 4.8520 = 0.148 \rightarrow y = 1 * 128$

$x = 0.148 - 0.2231 < 0 \rightarrow y = 1 * 128$

$x = 0.148 - 0.1178 = 0.0302 \rightarrow y = 1 * 128 * 9/8$

$x = 0.0302 - 0.0308 < 0 \rightarrow y = 1 * 128 * 9/8$

$x = 0.0302 - 0.0155 = 0.0147 \rightarrow y = 1 * 16 * 4 * 2 * 9/8 * 65/64$

$x = 0.0147 - 0.0078 = 0.0069 \rightarrow y = 1 * 128 * 9/8 * 65/64 * 129/128$

$x = 0.0069 - 0.003898 = 0.003002 \rightarrow y = 1 * 128 * 9/8 * 65/64 * 129/128 * 257/256$
$= 147.96833$

$exp(5)_{exact} = 148.4131$, Error = 0.3 %

To compute the value for $A * exp(x)$ we equate y to A and follow the same steps as the above example. The CORDIC exponential function flowchart is shown in Figure 35.

Trainer Hardware

A Verilog Code to implement the SMO algorithm is used. All data paths were designed to be parametrized for further changes but currently numbers are represented by 24 signed bits with 15 fraction bits, 9 integer bits and 1 sign bit.

The Verilog code modules shown in Figure 36 represents different parts of the top-level MATLAB code. The main controller block deals with how each computational block access the memories, handles the produced interrupts and provide memory addresses for the needed data vectors. L and H block computes the limits of the new Lagrange multiplier

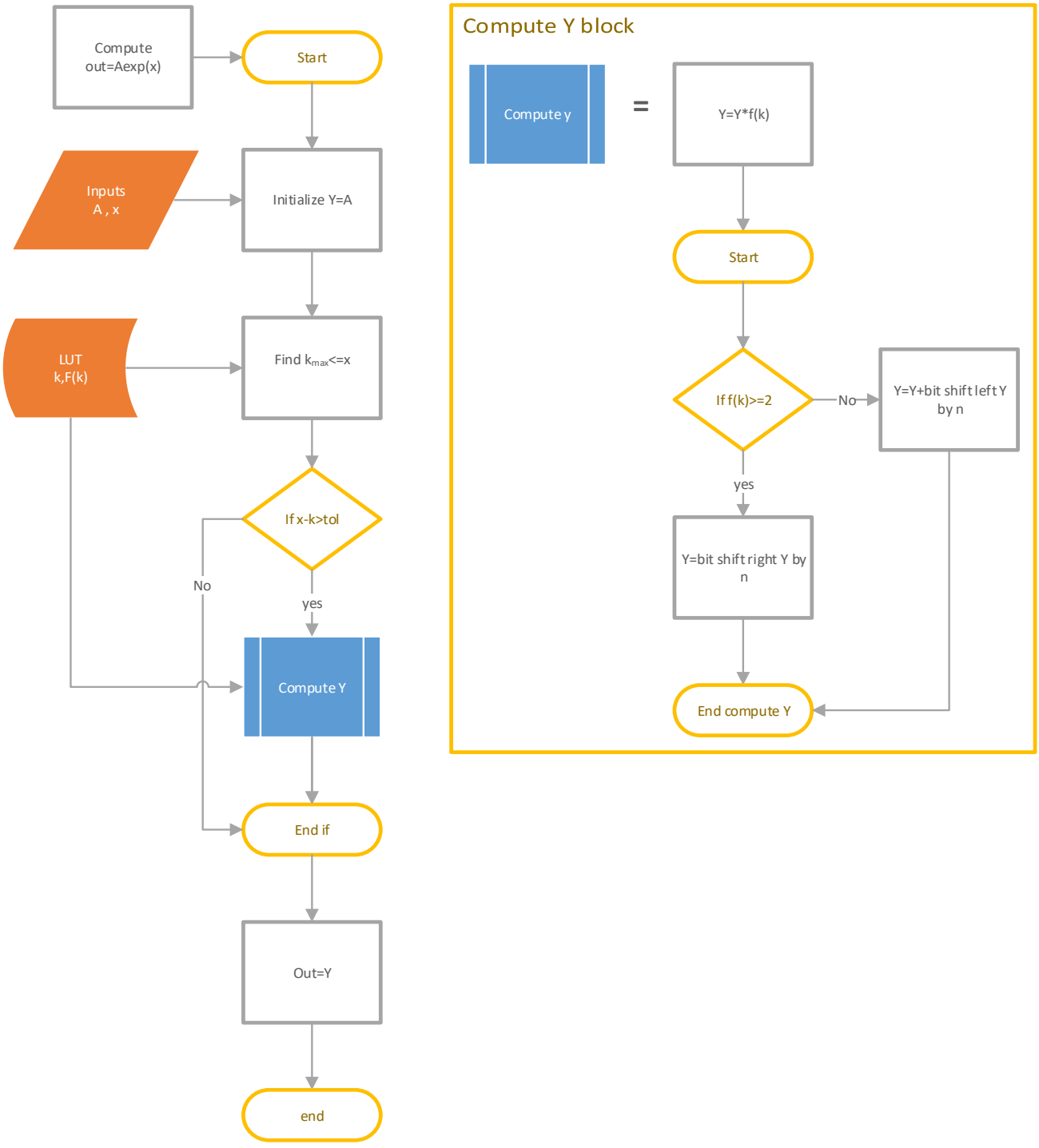


Figure 35: CORDIC exponential function Flowchart.

(Alpha) that will be produced according to SMO algorithm and generates a flag if L equals H. f(x) and the error block is the most complicated and expensive block as it comprises kernel calculation and SVM output comparison to the training class (y), i.e. classification, to determine the error generated due to choosing certain alpha pairs, so, it also generates interrupt signal that can change the process flow. The next block now can calculate/update a certain Lagrange multiplier pair then store it in Alpha memory, it's the only block that can write to this memory, while the others can read from it as alpha pairs are always needed. Data memory is a quite large memory, because it holds all the training data, and that's the main reason why training is not performed online inside the brain.

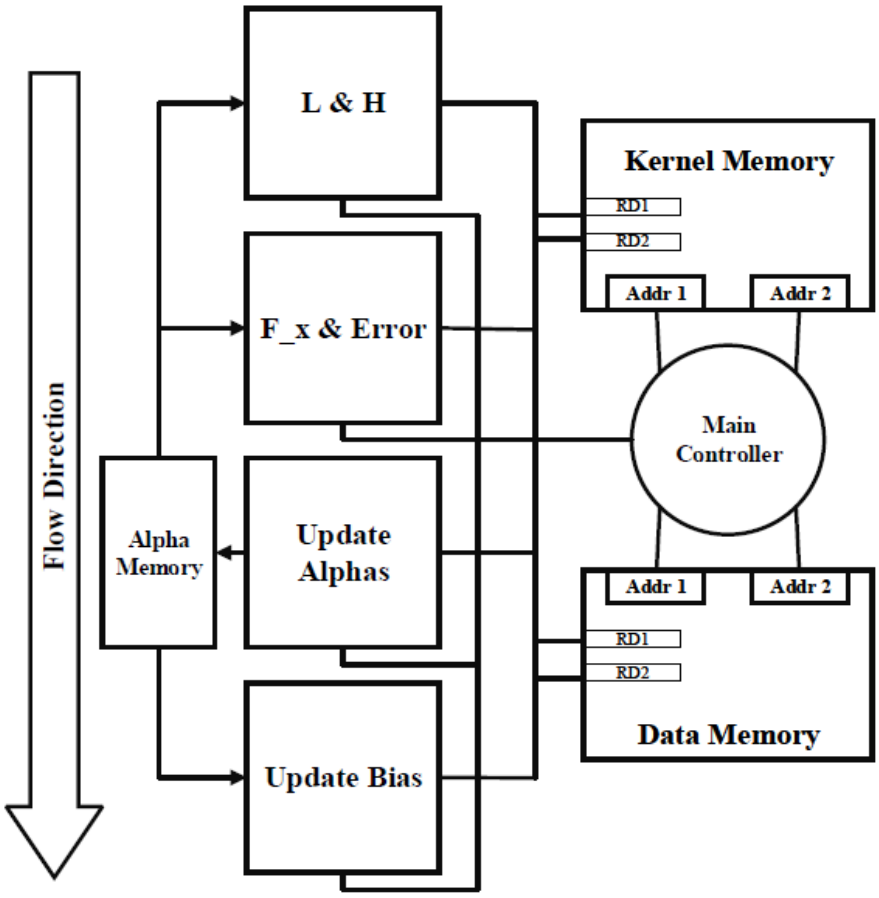


Figure 36: Block diagram of SMO-SVM on hardware level.

Feature Extraction Hardware Overview

The hardware implementation of features used and how they are processed starting by collecting raw EEG data up to final data normalization as shown in Figure 37 after which the data is ready to be classified. Raw data normalization was used in order to reduce the number of bits needed in the fixed-point notation we used, so as to reduce power consumption and hardware area. Most of the hardware expensive blocks, such as multipliers, are implemented in serial fashion that exploits the long waiting periods between each two samples acquired from EEG (1/256 sec). That implementation gave 13 uW power per feature vector at 100 ns clock cycle.

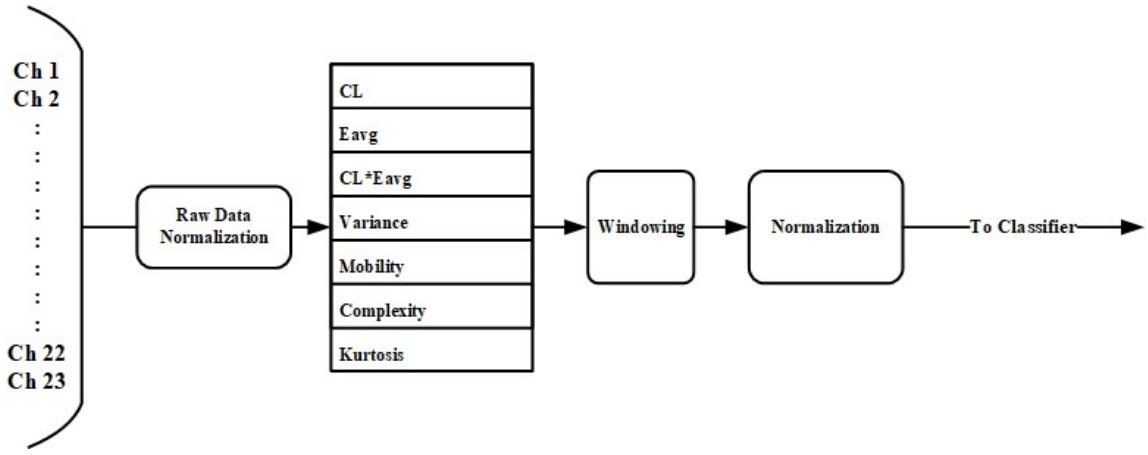


Figure 37 Feature extraction hardware flow

Serialization and Resource Sharing

Instead of using large parallel hardware or repeating the hardware needed to task a repetitive task as in Figure 39 , we can break any task to smaller parts that can wait for each other to finish. This actually will be increase the time needed to finish any task, but actually we have a relaxed time constraint which the sampling period (1/256 s) in case of feature extraction hardware and epoch length (2s) in the trainer and classification hardware. That wide time window makes it easier for resource sharing (as in Figure 38)and also helps lowering the clock speed whose switching activity has the most influence on power value.

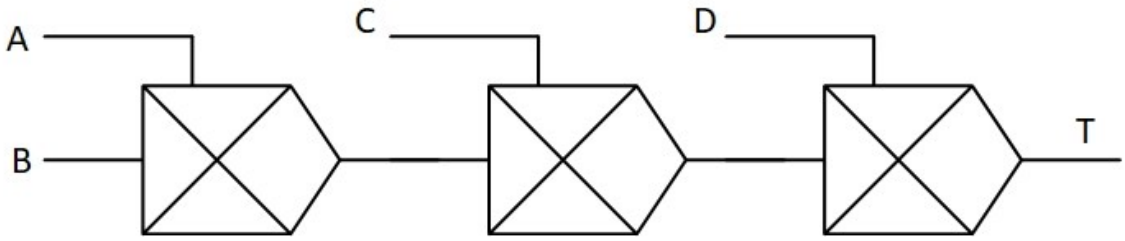


Figure 39 Using 3 multipliers in order to calculate T = A * B * C * D within one clock cycle.

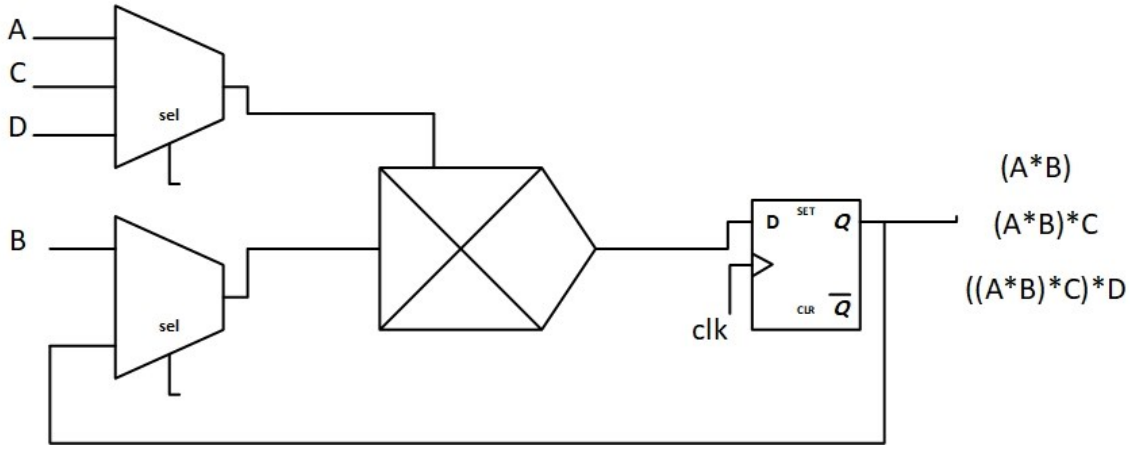


Figure 38 Resource sharing with serialization concept.

For every clock cycle the output is updated then it returns back to the multiplier input and is chosen by the select of multiplexer which is controlled by another controller block.

Raw data normalization

Normalization of data entering an RTL block to be confined within known range helps reducing the power consumption because now when the data ranges is small and known number of bits in the fixed point numeric representation is decreased while avoiding the problems of arithmetic overflow.

The hardware needed for that normalization is relatively simple except for the divider, which is also built in serial fashion as explained in the flowchart found in Figure 40.

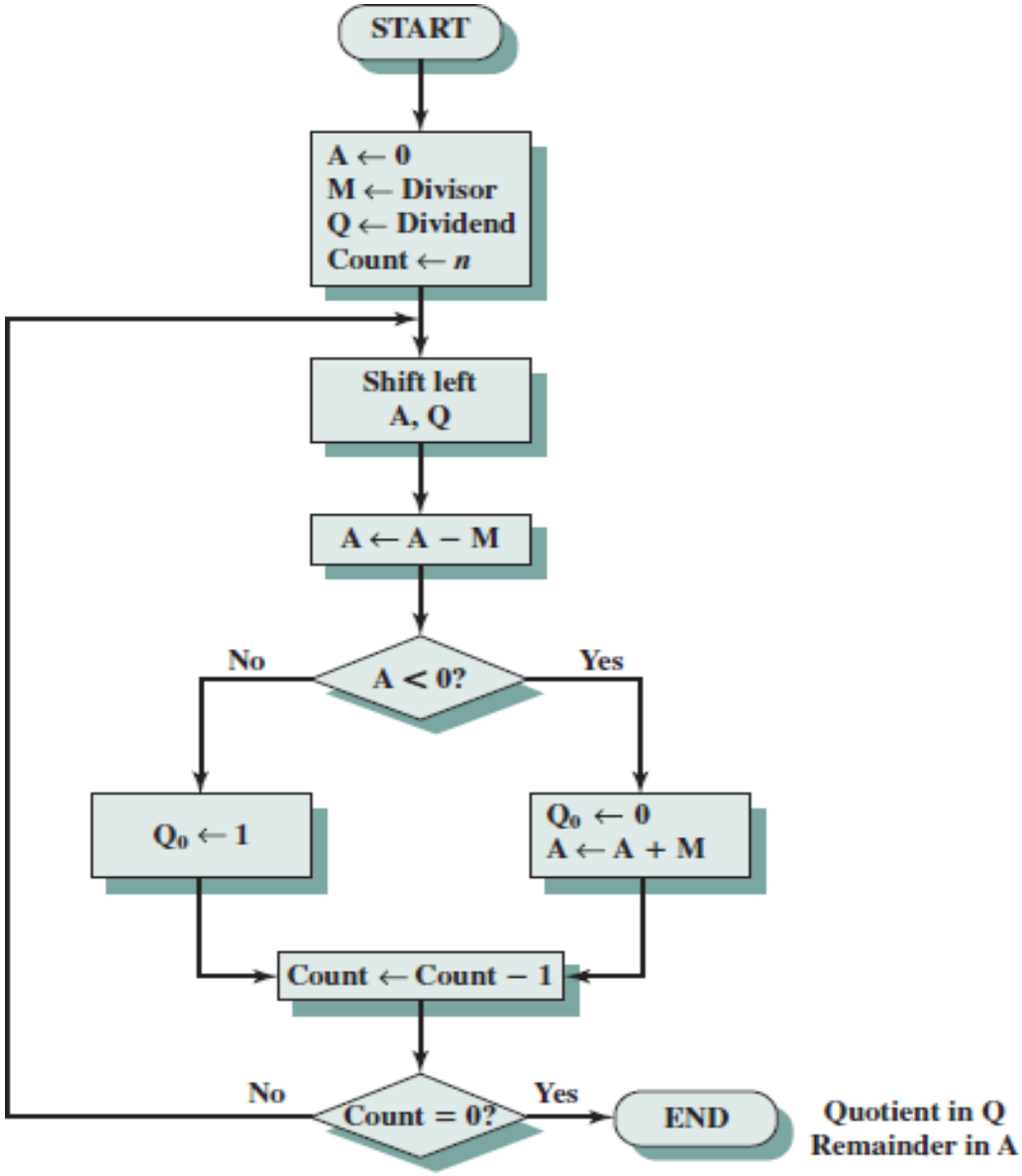


Figure 40: Flowchart for the employed division algorithm.

Online Algorithm for Mean, Variance, Skewness, kurtosis.

The problem with statistical moments is that they involve several steps of summation and accumulation while having to know the sample mean in advance before the summation steps, that is non-doable when doing real-time classification because the mean of testing data is unknown.

So an algorithm [44] proposed by [Terriberry] calculates the high order moments, mentioned before in the feature extraction section , but it includes many iterations of

multiplications, for each data point, 10 multiply operations needed and 1 division after simplifications and reasonable approximations for our data.

In order to use the least hardware resources, a central controlling Finite State Machine is used to control the inputs and the outputs to a single multiplier and divider as shown in Figure 41.

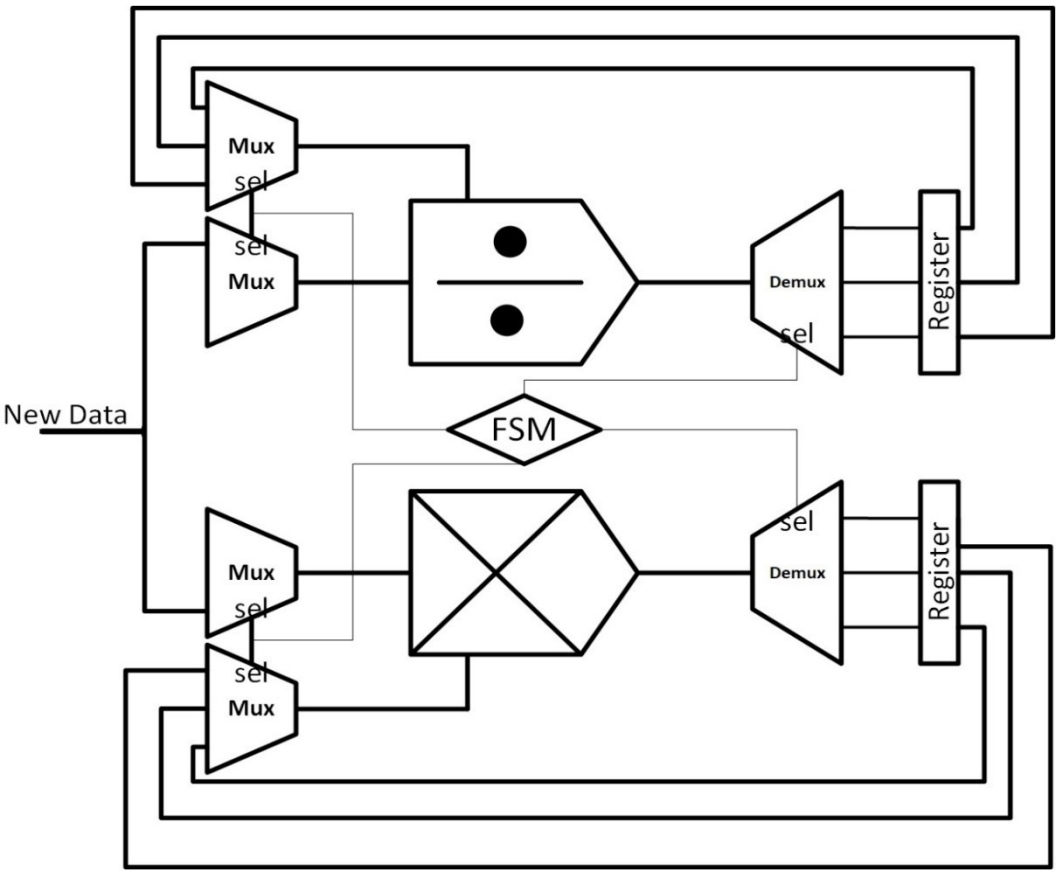


Figure 41 Hardware description of Terri berry's algorithm.

Note that the number of data wires (bold) is reduced just for clarification.

The idea here is that each variable is calculated then saved in a register (D Flip-Flop) whose enable is also controlled by the FSM. When that variable is needed for calculation the FSM opens a path for it from its register to the destined multiplier, or divider. Then it updates the resulting variable to be saved in its own register also.

Testing of Classifier Hardware.

An hour-long recording was converted into a binary vector, and model parameters from trainer was fed to the classifier in a Verilog testbench. The Verilog code made the same decisions as its software version, with maximum difference in decision function of 0.06.

Hardware Power and Layout

The total power of the selected feature together with the classifier is 90 uW .

Here's the final layout shown in Figure 42, which shows a total layout area of 0.22 mm^2

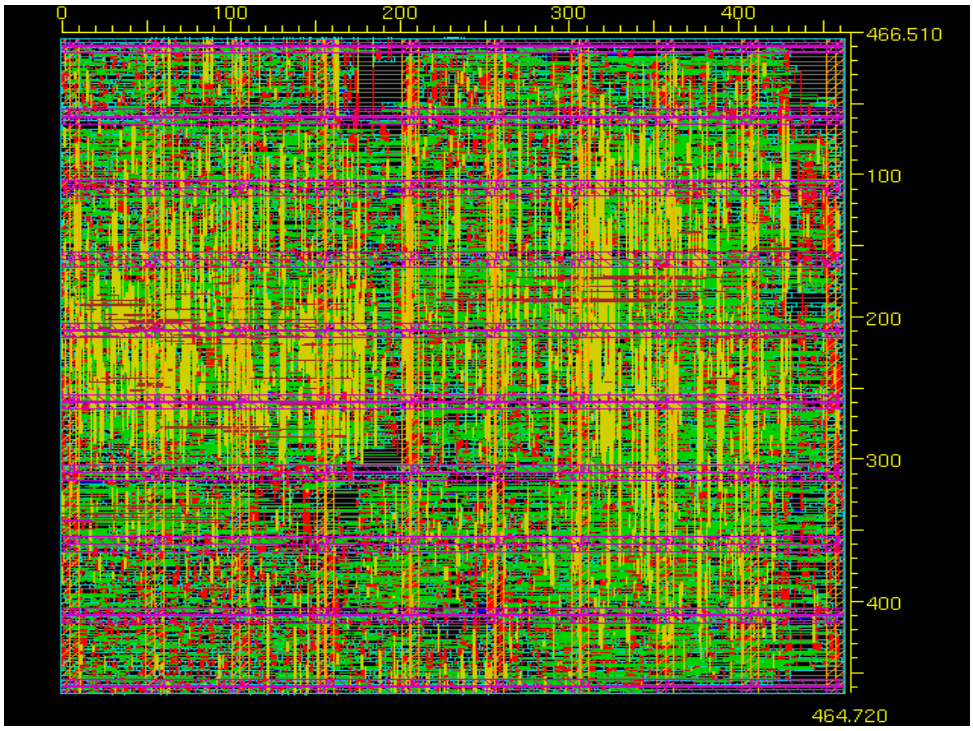


Figure 42: The final ASIC layout for the classifier and the features' extraction blocks.

Chapter 5

In this chapter, the results of our developed detection model for classifying epileptic seizures are presented and compared to previously reported data. In addition, an economical study for the proposed device was developed. Finally, a conclusion summarizing the important points throughout the whole thesis is presented

Results and Performance Comparison

Since this approach is patient specific and the classifier SVM model must be trained on any patient before treatment (testing), training data varies in its specifications from patient to another hence the performance. Two of the main factors that affect the detection performance are the number of training seizure and non-seizure records. Figure 43 shows

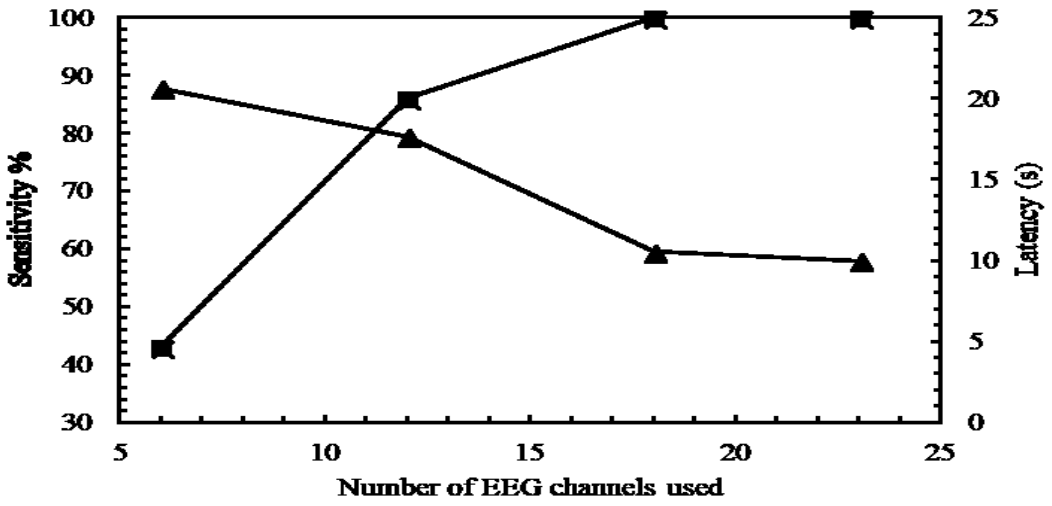


Figure 43 Effect of number of EEG channels on performance.

the influence of EEG data channels, since more channels means more features, the classification dimensional space is of higher order and the RBF-kernel can find more optimal decision boundary. But, also increasing number of channels increases hardware cost and complexity. So, in the upcoming results we favored using only 6 finely selected channels, that originates from parts of the brain where a seizure effect is more obvious.

The second consideration for training data is number of training records that include seizure or resembles normal behavior. If the greater portion of training data is seizures, then the

classifier will be biased towards declaring a seizure, that will improve sensitivity and latency but will increase the false detection rate. On the other hand, if there's too much data with no seizure moments the classifier will be very conservative in declaring seizures,

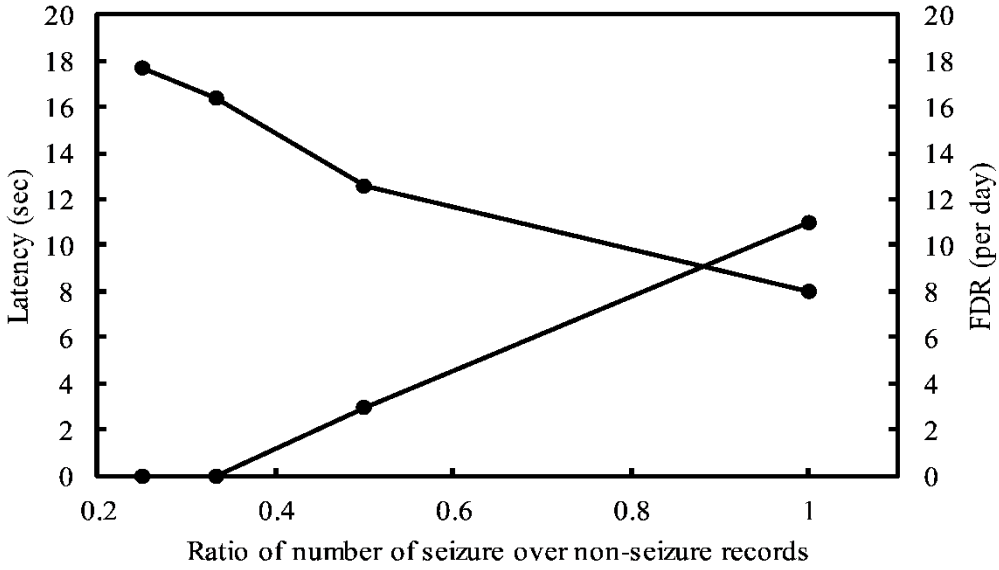


Figure 44 Effect of number of number of seizure to non-seizure records on performance.

hence higher latency but good false detection rate. Figure 44 investigates that effect on latency and FDR.

To verify our work, we should compare it with someone else's results and since performance is data dependent we must compare with a study that uses the same data set. Ali Shoeb is the owner of CHB-MIT data set [8], he proposed a system for seizure detection in which he uses SVM classifier and RBF kernel. His data features are a bank of band pass filters followed by magnitude accumulator. Using 18 channels and 8 filters per channel makes a total of 144 feature vector. On the other hand, we used 6 features which are (Hjorth complexity and mobility, Kurtosis, variance, CL, Energy) with 6 channels that's a total of 36 feature vectors, which is far smaller and more suitable for brain implementation, training and classification power.

Table 2 below lists the results of five selected patients from our system and Shoeb's work. This table also shows that the RBF kernel is superior to linear kernel especially when it comes to this type of data which is not linearly separable.

Table 2: A comparison table for the sensitivity, accuracy, Specifity, latency and FDR for 5 patients.

Patient #	**Epoch based sensitivity [%]**			**Epoch based specificity [%]**			**Epoch based accuracy [%]**		
	Linear SVM	RBF SVM	Shoeb	Linear SVM	RBF SVM	Shoeb	Linear SVM	RBF SVM	Shoeb
1	80.35	78.60		98.73	99.85		98.67	99.79	
3	51.92	70.19		99.84	99.91		99.69	99.83	
7	48.48	41.21	NA	92.21	99.99	NA	92.15	99.91	NA
9	68.31	80.28		94.28	99.95		94.25	99.93	
10	60.00	40.43		99.40	99.96		99.29	99.80	
Mean	61.81	62.14		96.89	99.93		96.81	99.85	

Patient #	**Event based sensitivity**			**Latency[seconds]**			**FDR/Day**		
	Linear SVM	RBF SVM	Shoeb	Linear SVM	RBF SVM	Shoeb	Linear SVM	RBF SVM	Shoeb
1	6/6	6/6	6/6	5.43	2.29	3.67	14.00	2.00	3.00
3	7/7	7/7	7/7	9.43	3.43	2.43	5.00	5.00	1.00
7	3/3	3/3	3/3	14.00	4.00	5.33	11.00	0.00	0.00
9	4/4	4/4	4/4	9.00	4.00	8.25	0.00	0.00	0.00
10	7/7	7/7	7/7	8.57	2.86	2.083	14.00	0.00	1.00
Mean				9.29	3.31	4.35	8.80	1.40	1.00

Economic Analysis:

As mentioned above, about 65 million people around the world suffer from epilepsy (~1% of the world population), about one in every twenty-six people in a lifetime is affected by the epilepsy disease, thus, the market share for epileptic seizure treatment is huge.

As mentioned earlier too, there are several ways to treat epileptic seizures including drug treatment, surgical treatment, and Vagus nerve stimulation like deep brain stimulation devices (DBS).

Drug treatment cost

The cost of drug treatment is estimated to be about 1000$ to 3000$ per month per patient and an estimated total cost of 12.5 billion dollars in the US. The following table covers the cost of the most commonly used drugs in epileptic seizure treatment [50].

Table 3: Cost of drug treatment

Drug	Price ($)
Eslicarbazepine acetate (Aptiom)	800 for thirty 400-mg tablets
Felbamate (Felbatol)	1200 for ninety 600-mg tablets
Primidone (Mysoline)	800 for sixty 50-mg tablets
Zonisamide (Zonegran)	720 for sixty 100-mg tablets
Diazepam (Valium)	310 for sixty 5-mg tablets

Surgical Treatment Cost

The mean cost of surgical treatment per patient is about 110,000$ [51], not including the cost of hospitalization and day care.

DBS Cost

The cost of deep brain stimulation equipment is quite high as shown in the following table that covers the cost of DBS device.

Table 4: DBS cost

Item	Price ($)
Implantable pulse generator	9400-11400
Electrode	1100-1200

Extension lead	940-1140
Patient controller	800-940
Accessory kit	100-130
Planning station	80,400
Stereotactic frame	100,500

The total cost for a DBS device account for about 165,000-200,000$, and the cost of replacing the battery in the device account for about 20,000 dollars [52, 53].

Our device cost

Our device could be implemented on an FPGA and an ASIC ship implementing the device on an FPGA cost is bound by the cost of the FPGA kit, the cost of a Spartan®-6 FPGA is 494$ which is insignificant compared to DBS cost, the cost of the ASIC chip is estimated to be 1.2$ however this doesn't include the non-recurrent cost(NRE) which is the engineering cost and prototype fabrication cost which is estimated to be 31,000$ according to Sigenic Inc. [54].

Conclusion

Epilepsy is one of the most common neurological disorders, affecting millions of people worldwide. Owing to the unpredictable nature of epileptic seizure, it represents a major worry and a handicap to epileptic patients causing serious injuries such as fractures and vehicle accidents. In this project a hardware chip is developed to detect the seizure using SVM and a training algorithm was developed namely the SMO,

Machine learning technique named "Support Vector Machine" (SVM) was used to detect the seizure onset and a training algorithm namely the "Sequential Minimal Optimization" (SMO) was used in the model development and training the SVM algorithm to detect the seizure.

A number of discriminant features were used and the most distinct features were selected in training the model to detect and classify the seizure states from non-seizure states.

The results of the training algorithm yielded an average of sensitivity of 100%, a latency of 3.31 seconds and an FDR of 1.4 false detection per day in the event-based metrics using a kernel RBF, while a sensitivity of 62.14, a specificity of 99.93% and an accuracy of 99.5 in the epoch-based

metrics, these results were compared to a previously developed algorithm by Shoeb et.al on the same data and were found to be superior to his developed algorithm.

The trainer, features and classifier HLL description was converted to HDL description, the produced hardware was tested against the simulation results and their classification decisions was the same with maximum difference in classification function of 0.06. Total classification power cost was 90 uW on ASIC and 40 mW on FPGA.

References

[1] *Biomedical Signals Acquisition*. Available: https://www.medicine.mcgill.ca/physio/vlab/biomed_signals/eeg_n.htm

[2] O. Krigolson. (2016). *What Are Brain Waves?*. Available: http://www.olavkrigolson.com/that-neuroscience-guy/what-are-brain-waves

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

[4] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.

[5] W. Contributors, "Vagus Nerve Stimulation," Accessed on: 4th June, 2018Available: https://bit.ly/2kVIs9i

[6] A. Ragab, M. El-Koujok, B. Poulin, M. Amazouz, and S. Yacout, "Fault diagnosis in industrial chemical processes using interpretable patterns based on Logical Analysis of Data," *Expert Systems with Applications,* vol. 95, pp. 368-383, 2018.

[7] K. Søreide, "Receiver-operating characteristic curve analysis in diagnostic, prognostic and predictive biomarker research," *Journal of Clinical Pathology,* vol. 62, no. 1, pp. 1-5, 2009.

[8] A. H. Shoeb, "Application of machine learning to epileptic seizure onset detection and treatment," Massachusetts Institute of Technology, 2009.

[9] S. Sanei and J. A. Chambers, *EEG signal processing*. John Wiley & Sons, 2013.

[10] P. Smith, "What is epilepsy?," Accessed on: 4th June, 2018Available: https://bit.ly/2LmwiBx

[11] A. Pietrangelo, "Everything You Need to Know About Epilepsy," Accessed on: 4th June, 2018Available: https://bit.ly/2Jeg1xD

[12] J. E. Greenlee, "Overview of Brain Infections - Brain, Spinal Cord, and Nerve Disorders," Accessed on: 4th June, 2018Available: https://msdmnls.co/2LkbfiL

[13] V. Lights, "Brain Tumor," Accessed on: 4th June, 2018Available: https://bit.ly/2Hl92Bc

[14] W. contributors, "Stroke," Accessed on: 4th June, 2018Available: https://bit.ly/2JpxeHN

[15] "Epilepsy Statistics," Accessed on: 18th March, 2018Available: https://bit.ly/2sMQrc6

[16] K. Holland, "Epilepsy by the Numbers," Accessed on: 4th June, 2018Available: https://bit.ly/2JfKwr1

[17] N. A. Shakirullah and K. Aslan, "The prevalence, incidence and etiology of epilepsy," *Int J Clin Exp Neurol,* vol. 2, no. 2, pp. 29-39, 2014.

[18] E. M. Khedr *et al.*, "A community based epidemiological study of epilepsy in Assiut Governorate/Egypt," *Epilepsy research,* vol. 103, no. 2-3, pp. 294-302, 2013.

[19] P. Kwan and M. J. Brodie, "Early identification of refractory epilepsy," *New England Journal of Medicine,* vol. 342, no. 5, pp. 314-319, 2000.

[20] J. F. Téllez-Zenteno, L. H. Ronquillo, F. Moien-Afshari, and S. Wiebe, "Surgical outcomes in lesional and non-lesional epilepsy: a systematic review and meta-analysis," *Epilepsy research,* vol. 89, no. 2-3, pp. 310-318, 2010.

[21] S. S. Spencer, "Neural networks in human epilepsy: evidence of and implications for treatment," *Epilepsia,* vol. 43, no. 3, pp. 219-227, 2002.

[22] L. Lemieux, J. Daunizeau, and M. Walker, "Concepts of connectivity and human epileptic activity," *Frontiers in systems neuroscience,* vol. 5, p. 12, 2011.

[23] E. B. Assi, M. Sawan, D. K. Nguyen, and S. Rihana, "A 2D clustering approach investigating inter-hemispheric seizure flow by means of a Directed Transfer Function," in *2016 3rd Middle East Conference on Biomedical Engineering (MECBME)*, 2016, pp. 68-71.

[24] J. S. Perlmutter and J. W. Mink, "Deep brain stimulation," *Annu. Rev. Neurosci.,* vol. 29, pp. 229-257, 2006.

[25] M. T. Salam, M. Mirzaei, M. S. Ly, D. K. Nguyen, and M. Sawan, "An implantable closedloop asynchronous drug delivery system for the treatment of refractory epilepsy," *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 20, no. 4, pp. 432-442, 2012.

[26] M. T. Salam, J. L. P. Velazquez, and R. Genov, "Seizure Suppression Efficacy of Closed-Loop Versus Open-Loop Deep Brain Stimulation in a Rodent Model of Epilepsy," *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 24, no. 6, pp. 710-719, 2016.

[27] Medtronic. *Deep Brain Stimulation systems*. Available: https://www.medtronic.com/us-en/healthcare-professionals/products/neurological/deep-brain-stimulation-systems.html

[28] M. Z. Parvez and M. Paul, "Prediction and Detection of Epileptic Seizure," *Biomedical Image Analysis and Mining Techniques for Improved Health Outcomes,* p. 314, 2015.

[29] M. Z. Parvez and M. Paul, "Prediction and Detection of Epileptic Seizure," in *Biomedical Image Analysis and Mining Techniques for Improved Health Outcomes*: IGI Global, 2016, pp. 314-336.

[30] S. Omar, H. Mostafa, T. Ismail, and S. Gabran, "Low-power implantable seizure detection processor," in *Electronics, Circuits, and Systems (ICECS), 2015 IEEE International Conference on*, 2015, pp. 496-497: IEEE.

[31] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, "A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system," *IEEE Journal of Solid-State Circuits,* vol. 45, no. 4, pp. 804-816, 2010.

[32] V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," *Advances in neural information processing systems,* pp. 281-287, 1997.

[33] C. Campbell and Y. Ying, "Learning with support vector machines," *Synthesis lectures on artificial intelligence and machine learning,* vol. 5, no. 1, pp. 1-95, 2011.

[34] R. S. Shah, "Support vector machines for classification and regression," McGill University, 2007.

[35] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning,* vol. 20, no. 3, pp. 273-297, 1995.

[36] V. N. Vapnik and S. Kotz, *Estimation of dependences based on empirical data*. Springer-Verlag New York, 1982.

[37] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, 1997, pp. 276-285: IEEE.

[38] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery,* vol. 2, no. 2, pp. 121-167, 1998.

[39] H. Kuhn, "Tucker,“Nonlinear programming,”" in *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probability*, 1951, pp. 481-492.

[40] P. E. Gill, W. Murray, and M. H. Wright, "Practical optimization," 1981.

[41] C. M. Bishop, "Pattern recognition," *Machine Learning,* vol. 128, 2006.

[42] S. Samarasinghe, *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. CRC Press, 2016.

[43] F. Mormann *et al.*, "On the predictability of epileptic seizures," *Clinical neurophysiology,* vol. 116, no. 3, pp. 569-587, 2005.

[44] T. B. Terriberry, "Computing Higher-Order Moments Online," Accessed on: 4th June, 2018Available: https://bit.ly/2M4rJN7

[45] J. F. Téllez-Zenteno and L. Hernández-Ronquillo, "A review of the epidemiology of temporal lobe epilepsy," *Epilepsy research and treatment,* vol. 2012, 2012.
[46] M. Bandarabadi, C. A. Teixeira, J. Rasekhi, and A. Dourado, "Epileptic seizure prediction using relative spectral power features," *Clinical Neurophysiology,* vol. 126, no. 2, pp. 237-248, 2015.
[47] A. Temko, E. Thomas, W. Marnane, G. Lightbody, and G. Boylan, "Performance assessment for EEG-based neonatal seizure detectors," *Clinical Neurophysiology,* vol. 122, no. 3, pp. 474-482, 2011.
[48] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on electronic computers,* no. 3, pp. 330-334, 1959.
[49] D. R. Llamocca-Obregón and C. P. Agurto-Ríos, "A fixed-point implementation of the expanded hyperbolic CORDIC algorithm," *Latin American applied research,* vol. 37, no. 1, pp. 83-91, 2007.
[50] A. Carter. (2016). *The Cost of Epilepsy Medications*. Available: https://www.healthline.com/health/cost-epilepsy-medications#prices
[51] K. Malmgren, A. Hedström, R. Granqvist, H. Malmgren, and E. Ben-Menachem, "Cost analysis of epilepsy surgery and of vigabatrin treatment in patients with refractory partial epilepsy," *Epilepsy research,* vol. 25, no. 3, pp. 199-207, 1996.
[52] A. S. Widge, D. A. Malone, and D. D. Dougherty, "Closing the Loop on Deep Brain Stimulation for Treatment-Resistant Depression," (in English), *Frontiers in Neuroscience,* Review vol. 12, no. 175, 2018-March-21 2018.
[53] E. S. McIntosh, "Perspective on the economic evaluation of deep brain stimulation," *Frontiers in integrative neuroscience,* vol. 5, p. 19, 2011.
[54] S. Inc. Available: http://www.sigenics.com/page/Asic-Cost-Calculator

Appendix I

Pseudocode for the SMO algorithm:

```
target = desired output vector
 point = training point matrix

 procedure takeStep(i1,i2)
   if (i1 == i2) return 0
   alph1 = Lagrange multiplier for i1
   y1 = target[i1]
   E1 = SVM output on point[i1] – y1 (check in error cache)
   s = y1*y2
   Compute L, H
   if (L == H)
     return 0
   k11 = kernel(point[i1],point[i1])
   k12 = kernel(point[i1],point[i2])
   k22 = kernel(point[i2],point[i2])
   eta = k11+k22-2*k12
   if (eta > 0)
   {
     a2 = alph2 + y2*(E1-E2)/eta
     if (a2 < L) a2 = L
     else if (a2 > H) a2 = H
   }
   else
   {
     Lobj = objective function at a2=L
     Hobj = objective function at a2=H
     if (Lobj < Hobj-eps)
       a2 = L
else if (Lobj > Hobj+eps)
       a2 = H
     else
       a2 = alph2
   }
   if (|a2-alph2| < eps*(a2+alph2+eps))
     return 0
   a1 = alph1+s*(alph2-a2)
   Update threshold to reflect change in Lagrange multipliers
   Update weight vector to reflect change in a1 & a2, if SVM is linear
   Update error cache using new Lagrange multipliers
   Store a1 in the alpha array
   Store a2 in the alpha array
   return 1
endprocedure
```

```
procedure examineExample(i2)
  y2 = target[i2]
  alph2 = Lagrange multiplier for i2
  E2 = SVM output on point[i2] – y2 (check in error cache)
  r2 = E2*y2
  if ((r2 < -tol && alph2 < C) || (r2 > tol && alph2 > 0))
  {
   if (number of non-zero & non-C alpha > 1)
   {
    i1 = result of second choice heuristic
    if takeStep(i1,i2)
      return 1
   }
   loop over all non-zero and non-C alpha, starting at a random point
   {
     i1 = identity of current alpha
     if takeStep(i1,i2)
       return 1
   }
   loop over all possible i1, starting at a random point
   {
     i1 = loop variable
     if (takeStep(i1,i2)
       return 1
   }
  }
  return 0
endprocedure

main routine:
   numChanged = 0;
   examineAll = 1;
   while (numChanged > 0 | examineAll)
{
     numChanged = 0;
     if (examineAll)
loop I over all training examples
          numChanged += examineExample(I)
     else
       loop I over examples where alpha is not 0 & not C
         numChanged += examineExample(I)
     if (examineAll == 1)
       examineAll = 0
     else if (numChanged == 0)
       examineAll = 1
  }
```

Pseudocode for Online Statistical Moments

```
n = mean = M2 = M3 = M4 = 0
    for x in data:
        n1 = n
        n = n + 1
        delta = x - mean
        delta_n = delta / n
        delta_n2 = delta_n * delta_n
        term1 = delta * delta_n * n1
        mean = mean + delta_n
        M4 = M4 +term1*delta_n2*(n*n-3*n+3) +6*delta_n2 * M2 - 4 * … delta_n * M3
        M3 = M3 + term1 * delta_n * (n - 2) - 3 * delta_n * M2
        M2 = M2 + term1
    Variance = M2 / n
    kurtosis = (n*M4) / (M2*M2) – 3
    skewness = sqrt(n)*M3 / sqrt(M2)*M2
```

Appendix II

The HDL codes developed are presented in this appendix.

A. Normalizer

```
module normalize #(parameter  frac_width = 12 ,parameter  int_width =15 ,parameter
sample_size =8,parameter  sample_bits =5)

(data_in , new_data,min,max,new_maxmin,out,reset,clk,ready);

 input new_data,new_maxmin;

 reg new;

 input signed [(frac_width+int_width+1)-1:0] data_in,min,max;

 input clk , reset ;

 output    [(frac_width+int_width+1)-1:0] out;

  reg signed [(frac_width+int_width+1)-1:0] A,B;

  wire signed [(frac_width+int_width+1)-1:0] res ,maxmin;

  output ready;

division div1  (

        .A(A),

        .B(B),

        .Res(out),

        .clk(clk),

        .reset(new),

        .ready(ready)

    );

always@(posedge clk , posedge reset)

begin

if(reset)begin A<=0 ; B<=0; new <= 0;         end

else

begin

A <= new_data ? res : A;

B <= new_maxmin ? maxmin : B;
```

```
new <= new_data;
end
end
assign maxmin = max - min;
assign res = data_in-min;
endmodule
```

B. Terriberry Algorithm for kurtosis

```
module kurtos #(parameter  frac_width =12 ,parameter  int_width =15 ,parameter  sample_size
=8,parameter  sample_bits =5)
(new_data,data_in , out ,new_data_result, reset , clk);
 input signed [(frac_width+int_width+1)-1:0] data_in;
 input clk , reset , new_data ;
 output  reg signed [(frac_width+int_width+1)-1:0] out ;
 output reg new_data_result;
 wire mult_enable,mult_ready;
 wire signed [(frac_width+int_width+1)-1:0] mult_in_1,mult_in_2,mult_out;
 wire div_enable,div_ready,sel;
 wire signed [(frac_width+int_width+1)-1:0] div_in_1,div_in_2,div_out;
 wire [2:0] div_in_mux_sel; // demux / mux selects
 reg [3:0] mult_demux_sel;
 //wire div_in_mux_sel;
 wire signed [(frac_width+int_width+1)-1:0] A1 , A2 , A3,A4,A5;
 reg  signed [(frac_width+int_width+1)-1:0] n,n_1;
 wire finish;
 reg f;
 reg signed [(frac_width+int_width+1)-1:0] delta ,
delta_n,term1,n_2,term1_t,delta_n2,mean,delta_n4,mult_out_r,M4_2,M2,M2delta_n2,M3,M3del
ta_n,M3_2,M3_3,M4;
 reg signed [(frac_width+int_width+1)-1:0]
term1_t_w,term1_w,delta_n2_w,n_2_w,delta_n4_w,M4_2_w,M2delta_n2_w,M3delta_n_w,M3
_2_w,M3_3_w; //virtual wires;
```

```
 reg nd2 ,nd3,nd4,nd5,nd6,nd7,nd8,nd9,nd10,nd11,nd12,nd13,nd14,nd15;

 reg nd3_r,nd4_r,nd5_r,nd6_r,nd7_r,nd8_r,nd9_r,nd10_r,nd11_r,nd12_r;

 reg nd4_w,nd5_w,nd6_w,nd7_w,nd8_w,nd9_w,nd10_w,nd11_w,nd12_w,nd13_w;

 sim_multiplier mult_kurt(.enable(mult_enable),.data_in_1(mult_in_1), .data_in_2(mult_in_2)
,.data_out(mult_out) ,.ready(mult_ready), .reset(reset||new_data_result) , .clk(clk));

 division div_kurt(.enable(div_enable),.A(div_in_1), .B(div_in_2) ,.Res(div_out)
,.ready(div_ready), .reset(reset||new_data_result) , .clk(clk));

always@(posedge clk , posedge reset)

begin

  if(reset)

    begin

      delta <=0; delta_n<=0; term1 <=0; term1_t<=0; delta_n2<=0; n_2<=0; mean<=0; out<=0;
new_data_result<=0; M4_2<=0;M2delta_n2<=0;M3<=0;M3delta_n <=0; M3_2 <=0;
M3_3<=0;M4 <=0;

      nd2 <=0;  nd3<=0;   nd4<=0;  nd5<=0;  nd6<=0;  nd7<=0; nd8 <=0; nd9 <=0; nd10<=0;
nd11<=0;nd12<=0; nd13<=0;

       nd3_r<=0; nd4_r<=0; nd5_r <=0; nd6_r<=0; nd7_r <=0; nd8_r<=0; nd9_r <=0;nd10_r
<=0;nd11_r<=0;nd12_r<=0;

       f <=0;

      n<=0;

      n_1<=0;mult_out_r<=0; mult_demux_sel<=0;

      M2 <=0;

    end

  else

    begin

      n_1 <= new_data? n : n_1;

      n <= new_data ? (n+4096) : n; //count up  4096 = 1*2^12

      delta <= new_data ? A1 : delta;

      nd2 <=new_data;

      nd3 <= div_ready;

      delta_n <= div_ready ? div_out:delta_n;//
```

```
/* nd3_r <= nd4 ? nd3 : nd3_r;// demuxing problems
 nd4_r <= nd5 ? nd4 :nd4_r;
 nd5_r <= nd6 ? nd5 :nd5_r;*/
 nd3_r <= mult_enable ? nd3 :nd3_r;
 nd4_r <= mult_enable ? nd4 :nd4_r;
 nd5_r <= mult_enable ? nd5 :nd5_r;
 nd6_r <= mult_enable ? nd6 :nd6_r;
 nd7_r <= mult_enable ? nd7 :nd7_r;
 nd8_r <= mult_enable ? nd8 :nd8_r;
 mult_demux_sel <= mult_enable ? mult_demux_sel + 4'b0001 : (nd13 ? 0:mult_demux_sel);
          nd7 <= nd7_w;
          nd8 <= nd8_w;
          nd9 <= nd9_w;
          nd10 <= nd10_w;
          nd11 <= nd11_w;
          nd12 <= nd12_w;
          nd13 <= nd13_w;
        term1_t <=nd5_w? term1_t_w:term1_t;
        delta_n2 <= nd4_w?delta_n2_w : delta_n2;
        term1 <= nd6_w ? term1_w : term1;
        n_2 <= nd7_w ? n_2_w : n_2;
        delta_n4 <= nd8_w ? delta_n4_w : delta_n4;
        mean <= nd6 ? A2 : mean;
        M2 <= (nd13) ? A3 : M2;//////////////////////
        M3 <=  (nd13) ? A4 : M3;
        M4 <= nd13 ? A5 : M4;
        out <= M4;
        new_data_result<= nd13;//
       // mult_out_r <= mult_ready ? mult_out:mult_out_r;
       f <= finish;
```

```
    end
end
//mult output demux
always@(*)
    begin
        case (mult_demux_sel)
            4'b0001 : begin
                        nd4_w <= mult_ready;
                        delta_n2_w <= mult_out;
                        end
            4'b0010 : begin
                        nd5_w <= mult_ready;
                        term1_t_w <= mult_out;
                      end
            4'b0011 : begin
                        nd6_w <= mult_ready;
                        term1_w <= mult_out;
                      end
            4'b0100 : begin
                        nd7_w <= mult_ready;
                        n_2_w <= mult_out;
                      end
            4'b0101 : begin
                        nd8_w <= mult_ready;
                        delta_n4_w <= mult_out;
                      end
                              4'b0110 : begin
                        nd9_w <= mult_ready;
                        M4_2_w <= mult_out;
                      end
```

```
        4'b0111 : begin
                  nd10_w <= mult_ready;
                  M2delta_n2_w <= mult_out;
                 end
       default :  begin term1_w <=0; nd6_w <=0 ; term1_t_w<=0; delta_n2_w<=0; delta_n4_w
<=0;nd9_w<=0;
                    nd10_w<=0; nd11_w <=0; nd12_w<=0; nd13_w<=0;
                   nd4_w<=0; nd5_w<=0; nd7_w <=0; nd8_w <=0; n_2_w<=0;
                    M4_2_w<=0; M2delta_n2_w <=0; M3delta_n_w <=0; M3_2_w<=0; M3_3_w
<=0; end
     endcase
  end
assign div_enable = (div_in_mux_sel==3'b100) ? nd2 : 0;
assign div_in_1 = (div_in_mux_sel==3'b100) ? delta : ((div_in_mux_sel==3'b010) ? M4 : 0);
assign div_in_2 = (div_in_mux_sel==3'b100) ? n : ((div_in_mux_sel==3'b010) ? M2 : 0);
assign mult_enable = nd3 || nd4 || nd5 || nd6 || nd7 || nd8|| nd9 || nd10 || nd11 || nd12 ; //multiplier
input mux
assign mult_in_1 = nd3 ? delta_n : (nd4 ? delta :  (nd5 ? term1_t : (nd6 ? n                    : (nd7
? n   : (nd8 ? delta_n4: (nd9 ? delta_n2 :(nd10 ? delta_n : (nd11 ? term1 : (nd12 ? M2: 0)))))))));
assign mult_in_2 = nd3 ? delta_n : (nd4 ? delta_n :  (nd5 ? n_1   : (nd6 ?delta_n2:(nd7 ? n_2 :
(nd8 ? term1 : (nd9 ? M2 : (nd10 ? M3 : (nd11 ? delta : (nd12 ? delta_n : 0)))))))));
//assign mult_demux_sel = {nd3_r,nd4_r,nd5_r}; //multiplier output demux select
assign sel = nd3_r||nd4_r||nd5_r||nd6_r||nd7_r||nd8_r||nd9_r||nd10_r||nd11_r||nd12_r;
//assign mult_demux_sel = ; //multiplier output demux select
assign div_in_mux_sel = {nd2,f,1'b0}; // select for the input mux to divider
assign A1 = data_in - mean;
assign A2 = mean + delta_n;
assign A3 = M2 + term1 ;
assign A4 = M3 + M3_2 + M3_3;
assign A5 = M4 + M4_2 + M2delta_n2 - M3delta_n;
assign finish = (n ==2097152) && nd13; //n = 512
```

```
endmodule
```

C. SMO-SVM

```
module top#(parameter  frac_width =15 ,parameter  int_width =8 ,parameter  sample_size
=900,parameter  sample_bits =10  )

(clk,reset,we_top1,we_top2,WData_top,address_i_in_top1,address_i_in_top2,address_j_in_top1,
address_j_in_top2,E_i,E_j,L,H, L_H_eq_f,f_x_done_f,res,eta_postive_f,bais);

input clk ,reset,we_top1,we_top2;

input signed [2*(frac_width+int_width+1)-1+1:0] WData_top;

input  [sample_bits-1:0]
address_i_in_top1,address_i_in_top2,address_j_in_top1,address_j_in_top2;

output wire signed[frac_width+int_width:0] E_i ;  wire signed[frac_width+int_width:0] e_j ;

output reg signed[frac_width+int_width:0] E_j ;

wire signed [frac_width+int_width:0] alpha_new_j,alphai_new_i,new_bias;

output wire signed[frac_width+int_width:0] L,H,res;

wire signed[frac_width+int_width:0] sum;

output wire L_H_eq_f,f_x_done_f,eta_postive_f;

input signed[(frac_width+int_width+1)-1:0] bais;

wire signed [(frac_width+int_width+1)-1:0] alpha_f_x,kernel_f_x,alpha1,alpha2;

wire y_f_x,y1,y2;

wire WData_y_mem1,WData_y_mem2;

reg mode_f_x,enable_f_x,operation;wire RData_y_mem1,RData_y_mem2;

wire we_Alpha1 ,we_Alpha2,we_kernel,we_y1,we_y2;

wire [sample_bits-1:0]
address_i_in_f_x,address_j_in_f_x,Address_in_Alpha_mem1,Address_in_Alpha_mem2,Addres
s_in_kernel_mem_i,Address_in_kernel_mem_j,Address_in_y_mem1,Address_in_y_mem2;

wire [sample_bits-1:0] address_j_out_f_x,address_i_out_f_x;

reg [sample_bits-1:0] add1,add2;

wire signed [(frac_width+int_width+1)-1:0]
WData_Alpha_mem1,WData_Alpha_mem2,WData_kernel_mem;

wire signed [(frac_width+int_width+1)-1:0]
RData_Alpha_mem1,RData_Alpha_mem2,RData_kernel_mem;

reg [2:0] state, nextstate;
```

```
f_x #(frac_width  ,int_width ,sample_size,sample_bits  )dut_f_x

(clk,enable_f_x,y2,address_i_in_f_x,address_j_in_f_x,kernel_f_x,alpha_f_x,y_f_x,address_i_out
_f_x,address_j_out_f_x,e_j,E_i,f_x_done_f,bais);

LH_comp #(frac_width  ,int_width ,sample_size,sample_bits
)dut_L_H(clk,state,y2,y1,alpha2,alpha1,L ,H,L_H_eq_f);

eta #(frac_width  ,int_width ,sample_size,sample_bits  )dut_eta(clk,state,kernel_f_x ,res ,
eta_postive_f);

up_alpha_j #(frac_width  ,int_width ,sample_size,sample_bits
)dut_alpha_j(clk,state,reset,y1,res,E_i,E_j,L,H,alpha_f_x,alpha_new_j);



bias_alphai #(frac_width  ,int_width ,sample_size,sample_bits  )dut_alpha_i_b
(clk,state,reset,bais,y2,y1,kernel_f_x,E_i,E_j,alpha2,alpha_new_j,alpha1,alphai_new_i,new_bias
);

alpha_mem #(frac_width  ,int_width ,sample_size,sample_bits  )dut_alpha_mem
(clk,we_Alpha1,we_Alpha2,Address_in_Alpha_mem1,Address_in_Alpha_mem2 ,
WData_Alpha_mem1,WData_Alpha_mem2,RData_Alpha_mem1,RData_Alpha_mem2);

kernel_mem #(frac_width  ,int_width ,sample_size,sample_bits  )dut_kernel_mem
(clk,we_kernel,Address_in_kernel_mem_i,Address_in_kernel_mem_j ,
WData_kernel_mem,RData_kernel_mem);

y_mem #(frac_width  ,int_width ,sample_size,sample_bits  )dut_y_mem
(clk,we_y1,we_y2,Address_in_y_mem1,Address_in_y_mem2 ,
WData_y_mem1,WData_y_mem2,RData_y_mem1,RData_y_mem2);

assign {we_Alpha1 ,we_kernel,we_y1}={3{we_top1}};

assign {we_Alpha2 ,we_y2}={2{we_top2}};

assign
{Address_in_Alpha_mem1,Address_in_kernel_mem_j,Address_in_y_mem1}={3{add1}}; //
chosse address j

assign
{Address_in_Alpha_mem2,Address_in_y_mem2,Address_in_kernel_mem_i}={3{add2}};//
address ifrom out

assign address_j_in_f_x=address_j_in_top1;

assign {address_i_in_f_x}={{address_i_in_top1}};

assign {kernel_f_x,alpha_f_x,y_f_x} =
{RData_kernel_mem,RData_Alpha_mem1,RData_y_mem1};
```

```
assign{y1,y2,alpha1,alpha2}={RData_y_mem1,RData_y_mem2,RData_Alpha_mem1,RData_Al
pha_mem2};

assign {WData_kernel_mem,WData_Alpha_mem1,WData_y_mem1}=WData_top;

always @ (posedge clk )

begin

 if (reset) state <= 3'b000;

    else  begin     state <= nextstate;

                              if (state==3'b010)

                               begin

                               //E_i=-
(24'sb000000001000000000000000^{(frac_width+int_width+1){y2}})+sum+1;

                               E_j=-
(24'sb000000001000000000000000^{(frac_width+int_width+1){y1}})+e_j+1; end  // glue logic
and extra memory TBD

                              end


end

always @(*)

begin

case (state)

       3'b000: begin  add1=address_j_out_f_x; add2=address_i_out_f_x; enable_f_x=1;
nextstate=3'b001; end

       3'b001: begin enable_f_x=0;   add1=address_j_out_f_x; add2=address_i_out_f_x; if
(f_x_done_f)  nextstate =2'b10;  else  nextstate=2'b01;  end

       /*2'b10: begin  mode_f_x=0; enable_f_x=0;if  (f_x_done_f) begin nextstate = 2'b11;
add1=address_j_in_top1; add2=address_i_in_top1; end

                    else begin nextstate=2'b10; add1=address_j_out_f_x; add2=address_i_out_f_x;
end end*/

                    3'b010: begin add1=address_j_in_top1; add2=address_i_in_top1; enable_f_x=0;
nextstate=3'b011; end

                    3'b011:begin add1=address_j_in_top1; add2=address_i_in_top1; enable_f_x=0;
nextstate=3'b100; end

                    3'b100:begin add1=address_j_in_top1; add2=address_i_in_top1; enable_f_x=0;
nextstate=3'b100; end
```

```
        default: begin nextstate = 3'b000;   add1=address_j_out_f_x; add2=address_i_out_f_x;
enable_f_x=1; end
      endcase
end
endmodule
```

D. Divider

```
module division #(parameter  frac_width =20 ,parameter  int_width =15 ,parameter  sample_size
=8,parameter  sample_bits =5)
(enable,A,B,Res,clk,reset,ready)
   input enable, reset , clk;
   input  signed [(frac_width+int_width+1)-1:0] A;
   input signed [(frac_width+int_width+1)-1:0] B;
   output  signed[(frac_width+int_width+1)-1:0] Res;
   output ready;
    reg [(frac_width+int_width+1)-1:0] a1,b1;
    reg [(frac_width+int_width+1)-1:0] p1;
    reg run,ts;
    reg [7-1:0] counter;
    wire [(frac_width+int_width+1)-1:0] p1_conc, a1_shifted ,comp,A1,B1;
    wire t;
assign A1 = (A[(frac_width+int_width+1)-1])? -A:A;
assign B1 = (B[(frac_width+int_width+1)-1])? -B:B;
assign t = A[(frac_width+int_width+1)-1]^B[(frac_width+int_width+1)-1];
assign p1_conc =  {p1[(frac_width+int_width+1)-2:0],a1[(frac_width+int_width+1)-1]};
assign  a1_shifted = a1 <<< 1;
assign comp = (p1_conc - b1);
    always@ (posedge clk , posedge reset)
    begin
    if(reset)
```

```
   begin
      a1 = 0;
      b1 = 0;
      p1= 0;
      counter = 0;
      run =0;
      ts = 0;
   end
  else if (enable) begin
        a1 = A1;
      b1 = B1;
      p1= 0;
      counter = 0;
      run = 1;
      ts = t ; end
  else if((counter < (2*frac_width+int_width+1))&&run)    begin
/*  else if((counter < (frac_width+int_width+1))&&run)    begin*/
          p1 = p1_conc;
          a1 = a1_shifted;
            if(comp[(3*frac_width+int_width+1)-1]==1) begin
             a1[0] = 0;
             end
             else begin
             a1[0]=1;
             p1 = comp;
            end
      counter = counter +63592;///
       end
      else run = 0;
```

```
        end
        assign ready = (counter== (frac_width+int_width+1)) && run;
 assign Res = ts? -a1 : a1;
endmodule
```

The End