Cairo University

HARDWARE IMPLEMENTATIONS OF MACHINE LEARNING TECHNIQUES FOR NEURAL SEIZURE DETECTION

By

Mohamed Adel Attia Elhady Elgammal

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2018

- *The Student must Return to the Postgraduate Office

HARDWARE IMPLEMENTATIONS OF MACHINE LEARNING TECHNIQUES FOR NEURAL SEIZURE DETECTION

By

Mohamed Adel Attia Elhady Elgammal

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Dr. Ahmed Nader Mohieldien	Dr. Hassan Mostafa Hassan
……………………………….	……………………………….
Associate Professor Department of Electronics and Electrical Communications Faculty of Engineering, Cairo University	Assistant Professor Department of Electronics and Electrical Communications Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2018

- *The Student must Return to the Postgraduate Office

HARDWARE IMPLEMENTATIONS OF MACHINE LEARNING TECHNIQUES FOR NEURAL SEIZURE DETECTION

By
Mohamed Adel Attia Elhady Elgammal

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Approved by the
Examining Committee

____
Prof. Dr. Ahmed Nader Mohieldein, Thesis Main Advisor

____
Prof. Dr. Mohamed Fathy Abu-Elyazeed, Internal Examiner

____
Prof. Dr. Yehya Hassan Ghallab, External Examiner
- Associate professor, Faculty of Engineering Helwan University.

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2018

- *The Student must Return to the Postgraduate Office

Engineer's Name: Mohamed Adel Attia Elhady Elgammal
Date of Birth: 05/12/1993
Nationality: Egyptian
E-mail: Mohamed.adel567@gmail.com
Phone: 01067780710
Address: Faisel st., Giza, 11443
Registration Date: 1/10/2016
Awarding Date: …./…./……..
Degree: Master of Science
Department: Electronics and Electrical Communications Engineering

Supervisors:

Prof. Ahmed Nader Mohieldin
Dr. Hassan Mostafa Hassan

Examiners:

Prof. Yehya H. Ghallab (External examiner)
Prof. Mohamed F. Abu-Elyazeed (Internal examiner)
Prof. Ahmed N. Mohieldein (Thesis main advisor)

Title of Thesis:

HARDWARE IMPLEMENTATIONS OF MACHINE LEARNING TECHNIQUES FOR NEURAL SEIZURE DETECTION

Key Words:
Seizure Detection; Machine Learning; Support Vector Machine; Artificial Neural Network; Accelerator.

Summary:

In this thesis an automatic seizure detection is proposed. For features extraction, more than 20 linear and nonlinear features are software implemented and tested to measure their efficiency in seizure detection. For classification block, two different algorithms are implemented: Artificial Neural Network (ANN) and Support Vector Machine (SVM). Support Vector Machine (SVM) training accelerators are also implemented using two different techniques: Gradient Ascent (GA) and Sequential Minimal Optimization (SMO). Finally, a new EEG dataset is extracted from rats in collaboration with a research team from the Faculty of Science, Cairo university and ONE lab.

Acknowledgments

In the beginning of this thesis, I would like to thank many people who supported me and encouraged me to give more effort to reach that output.

First, to my advisors, Dr. Ahmed Nader Mohieldein and Dr. Hassan Mostafa, I would like to express my sincere gratefulness and appreciation for their excellent guidance, caring, patience, and immense help in planning and executing the work in a timely manner. Their great personality and creativity provided me with an excellent atmosphere for work, while their technical insight and experience helped me a lot in my research. Their support at the time of crisis will always be remembered.

Of course, I cannot find words enough to express the gratitude and appreciation that I owe to my family. My father, my mother, my sisters and my brother Mahmoud are my main supporters and they owe everything I reach. Their tender love and support have always been the cementing force for building what was achieved.

Dedication

This thesis is dedicated to my father and my mother.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviation	Description
AED	Anti-Epileptic Drugs
ANN	Artificial Neural Network
CNS	Central Nervous System
ECG	Electrocardiogram
FD	Fractal Dimension
FFT	Fast Fourier Transform
FNPS	False Negatives Per Seizure
FPPS	False Positives Per Seizure
GA	Gradient Ascent
MAV	Mean Absolute Value
ML	Machine Learning
PPM	Partial Products Matrix
QP	Quadratic Programming
RBF	Radial Basis function
RMS	Root Mean Square
SD	Standard Deviation
SDA	Seizure Detection Algorithm
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
VNS	Vagal Nerve Stimulations
WHO	World Health Organization
WT	Wavelet Transform

Abstract

Epilepsy is one of the most common neurological disorders that affects lives of millions of people around the world. Therefore, automatic seizure detection systems has been introduced.

The proposed work in the thesis aims to design and implement an implantable chip that helps in seizure detection. The system of automatic seizure detection consists of 4 stages: preprocessing, feature extraction, feature selection and classification. For features extraction, more than 20 linear and nonlinear features are software implemented and tested to measure their efficiency in seizure detection. Then, an exhaustive search is performed to choose the best features.

For the classification block, different machine learning techniques are hardware implemented to classify seizure and non-seizure epochs. The classifier block is implemented using Artificial Neural Network (ANN) and Support Vector Machine (SVM). A comparison is performed between the two classifiers on the performance, area and energy consumption. A modification is proposed on ANN to improve performance.

As the neural seizure detection is a very complex problem, support vector machine (SVM) training accelerators are implemented to speed up the training phase. The implementation of the accelerator is done using two different algorithms: Gradient Ascent (GA) and Sequential Minimal Optimization (SMO).

Moreover, a new EEG dataset is extracted in collaboration with a research team from the Faculty of Science, Cairo University and ONE lab. The new dataset is extracted from rats before, during and after seizures. This dataset is extracted using commercial industrial amplifier and a BioBench based software.

Chapter 1 : Introduction

Human brain is the main part of the central neural system (CNS). It is a very complex system that consists of billions of neurons organized in a huge network. It is responsible on receiving and collecting measurements from sensors all over the body and taking decisions to make humans behave as they do. This great system –the human brain- is divided into multiple regions. Each region is responsible on a specific task. Understanding how human brain works is a very interested research topic that has been studied at different spatial scales: microscopic and macroscopic. It is found that different neurons and regions communicate with each other through this network. Many Disorders affect human brain and consequently cause malfunction in human behavior.

1.1. Motivation

Epilepsy is a central nervous system (CNS) disorder resulting from abnormal activities. It is one of the chronic diseases the affects people from all ages. According to World Health Organization (WHO), more than 50 millions around the world have epilepsy [1]. Epilepsy causes seizures on infrequent basis. Epileptic seizures vary in type, strength and duration. People who have epilepsy face many obstacles in their daily life such as driving a car and cooking. Epileptic seizure is a large-scale phenomenon in which a large portion of the brain is involved in the abnormal activity not only one neuron. Thus, having a very large number of neurons and a dense network among these neurons are the main conditions for epileptic seizures. These conditions are satisfied in the human brain in the normal activity [2].

Epilepsy is classified into some generalized categories: focal seizures, non-focal seizures and continuous seizures. In focal epilepsy, a specific part of the brain is the main source of the seizures due to some damaged neurons. These damaged neurons start the abnormal activity then this activity spreads to a large portion of the brain.

In non-focal seizures, sometimes called generalized seizures, the epileptic activity starts at the whole brain simultaneously. Scientists suggests that the cause of generalized seizures is due to brain properties rather than some damaged neurons [2].

In continuous seizures, there is almost no recovery between the seizures. It is the most dangerous type of seizures as it might threat patient’s life.

1.2. Proposed Work

In this thesis proposal, an automatic seizure detection system is proposed to measure the EEG signal of a seizure patient. The system extracts some discriminating features from the EEG. Then, different classification techniques are proposed to classify the seizure and non- seizure time epochs. Hardware implementations of support vector machine (SVM) classifier and artificial neural network (ANN) are proposed and compared. Moreover, a hardware implementation of an accelerator of support vector machine learning is implemented using two different techniques. The two techniques are: gradient ascent (GA) and sequential minimal optimization (SMO).

1.3. Organization of the thesis

The reminder of the thesis is organized as follows: Chapter 2 introduces basic concepts for the epilepsy treatment techniques, the EEG signal, automatic seizure detection system and machine learning techniques. It also introduces a literature review of the previous work done on the literature. Chapter 3 presents detailed analysis of the proposed feature extraction and selection process. It also tabulates the results of the feature extraction and selection and the best features found. Chapter 4 presents a detailed analysis of the SVM training procedure and two different algorithms are presented and hardware implemented. Chapter 5 presented a detailed analysis of different classifiers techniques and their proposed hardware implementations. Chapter 6 shows the work done to generate a new EEG dataset from rats to be used in testing. Finally, appendices illustrates the MATLAB codes used for software simulations and the detailed results of feature selection process.

Chapter 2 : Literature Review.

2.1. Diagnosis and Treatment of Epilepsy

The presence of abnormal or damaged neurons in the brain does not necessarily cause seizures. To diagnose an epileptic seizure, many imaging of the brain should be taken. Also, medical history of the patient should be reviewed.

After diagnosis an epilepsy and determining its type, different treatment techniques such as Anti-Epileptic Drugs, Surgical resection and Electrical stimulation are used.

Anti-epileptic drugs (AEDs) is one of the most common methods to treat epilepsy. AEDs attempt to treat epilepsy by changing the chemistry of the brain. Hence, AEDs aim to control seizures and they work well with almost two-thirds of epilepsy patients. On the other hand, they have many side effects as they affect the whole brain. Another drawback of the AEDs is that they are totally experimental. Doctors start to try a combination of drugs that shows good results with other patients who have the same age, gender and medical history. Then, they try different combinations and doses till they get the right combination that gives the best result with that patient. That best mixture of drugs should balance between controlling the seizures and minimizing the side effects as much as possible. A great research is being done on AEDs and is achieving good results [3].

The second technique that is used in epilepsy treatment is surgical resection [4]. This solution is used specially for focal seizures and when a mixture of more than 3 AEDs could not control seizures [2]. A surgery of removing the damaged neurons and resection it from the brain network is performed. This surgery causes that the abnormal activity of the damaged neurons could not be transferred to the whole brain. Hence, no seizures occur. Many tests should be done on the brain before starting the surgery to determine the portion of the brain that causes seizures. Also, the removed portion should not be responsible of one of the main functions of the patient like memory, vision, hearing, speaking or moving. The large amount of redundancy in human brain neurons made it possible to remove a small portion without facing a great effects on human’s daily life.

When the first two techniques could not help in epilepsy treatment, Doctors think of alternative ways to control and limit seizures for this patient. One of these ways is electrical stimulation. Many people may think that electrical stimulation for neurons may cause more seizures not reducing them. However, it is proven that electrical stimulation causes a reduction in seizures in many cases [5].

Vagal nerve stimulation (VNS) is one of the most common treatments of epilepsy based on electrical stimulation [6]. VNS includes implanting stimulating electrodes on the brain cortex and an electrical battery on the chest cavity. These electrodes are used to give electrical stimulation to specific regions in the brain lobe to reduce seizures [7]. The clinical experiments of VNS have showed a reduction by 50% of the total number of seizures. Also, the implanted device stays working for years after activation [2]. VNS also has the advantage of not having the side effects caused by AEDs. However, VNS has some drawbacks. First, it is a way to reduce seizures not eliminate them. Second, VNS affects a large portion of the brain not the required portion only.

The way the electrical is applied to the brain is under great research. Traditionally, the electrical stimulation was used continuously on an on-off modes. In slow on-off mode, the stimulation is used for 30 seconds. Then, it is being off for 5 minutes. While in fast mode the stimulation is used for 7 seconds and then being off for 12 seconds [8]. The choice of a specific mode, period and shape of an electrical stimulation used for a specific patient is usually empirical.

Nowadays, research is done to detect seizures and apply electrical stimulation once a seizure has begun instead of applying it continuously. This will minimize the side effects greatly. Moreover, the battery life will be extended greatly. However, many challenges face researchers. Automatic seizure detection is very challengeable and many research is being done for the automatic detection and prediction of epileptic seizures with different approaches. One approach is to analyze the muscles movement to detect epileptic seizures [9]. Another approach is studying the electrocardiogram (ECG) signal of the heart [10]. A third approach is electroencephalogram (EEG) analysis.

2.2. Electroencephalogram (EEG) signal

As mentioned above, Analysis of EEG signal is one of the most common approaches used for seizure detection and prediction. EEG is an electrical record of what is happening inside the brain. Traditionally, Electrical voltage was first measured from monkeys on 1875. However, there was almost no meaningful benefit from it until 1920s [11].

EEG signal is the electrical signals generated by human brain. These electrical signals' amplitude are less than 300μV. The frequency response of these signals are spanned to 100Hz. Because of the very low amplitude of the EEG signals, the process of EEG measurement is a very challengeable task.

EEG measurements are made at various scales. First type is scalp EEG where measurement electrodes are added on the skull. The scalp electrodes can be easily attached. However, recordings from scalp EEG are highly attenuated as the skull acts as a filter so a very large portion of the brain should be involved in the seizure to be able to detect seizures from EEG. However, the performance of EEG measurement using scalp electrodes can be enhanced by using more electrodes. In practice, more than 20 electrodes are used and placed on patient's skull. However, some research has proposed more electrodes up to 256 electrodes to increase measurement performance [12]. The placement of the electrodes on the skull follows many standards as 10-10 and 10-20 system. A typical EEG signal measured from 4 different scalp electrodes are shown in Figure 1.

The second type of EEG measurement is intra-cranial EEG where electrodes are implanted on the cortex in a surgery. This type of measurement is more accurate and can record measurement of a smaller scale of neurons [13].

The EEG signal frequency domain is divided into multiple frequency bands:

- The Delta bands contains signals with frequencies less than 4 Hz.
- The Theta band contains signals with frequencies between 4-7 Hz.
- The Alpha band contains signals with frequencies between 8-12 Hz
- The Beta band contains signals with frequencies between 12-30 Hz
- The Gamma band contains signals with frequencies between 30-100 Hz

These bands are shown in Figure 2.

Each frequency band contains a specific kind of information. Research is performed to extract information from each frequency band. Cantero et al. proved that the Theta band contains information about the transition from sleeping to waking up [14]. Palva et al. proved that the Alpha band contains information about making a calculation [15]. The second type of information that can be extracted from EEG signals is the transient information. In transient analysis, different spikes are measured and analyzed. These spikes can be caused due to a neurological disease like epilepsy or due to other artifacts. These artifacts exist due to different causes like biological or environmental reasons. It is so important to remove such artifacts before processing the EEG signal to detect seizure.

2.3. Seizure Detection

One of the main problems that is obstructing the research for epilepsy treatment is the absence of a perfect way to detect seizure. In the pre-computer era the reading of EEG was performed by experienced encephalographers who, based on their experience, decided whether the recording was a seizure or not. Nowadays, even with the great computational power, the EEG analysis by expert encephalographers remains one of the most powerful approaches for seizure detection. However, the EEG analysis by experts are very subjective and very time-consuming. The purpose of seizure detection algorithms (SDA) is to replace this old-fashioned way of EEG analysis by another process that automatically detect seizures. In order to compare the performance of different detection methods some of the following important performance measures can be used. The first measure is the percentage of missed seizures in 24h. However, as noted by P. Buteneers [12], it is probably more relevant to look at the false negatives per seizure (FNPS), as this measure allows a fair comparison between different EEG recordings. The same applies for another measure, namely the number of false positives, where the false positives per seizure (FPPS) can replace the number of false positives during 24h. From a more practical point of view the time necessary for the detection of the seizure, also called the detection delay, is an important parameter as well.

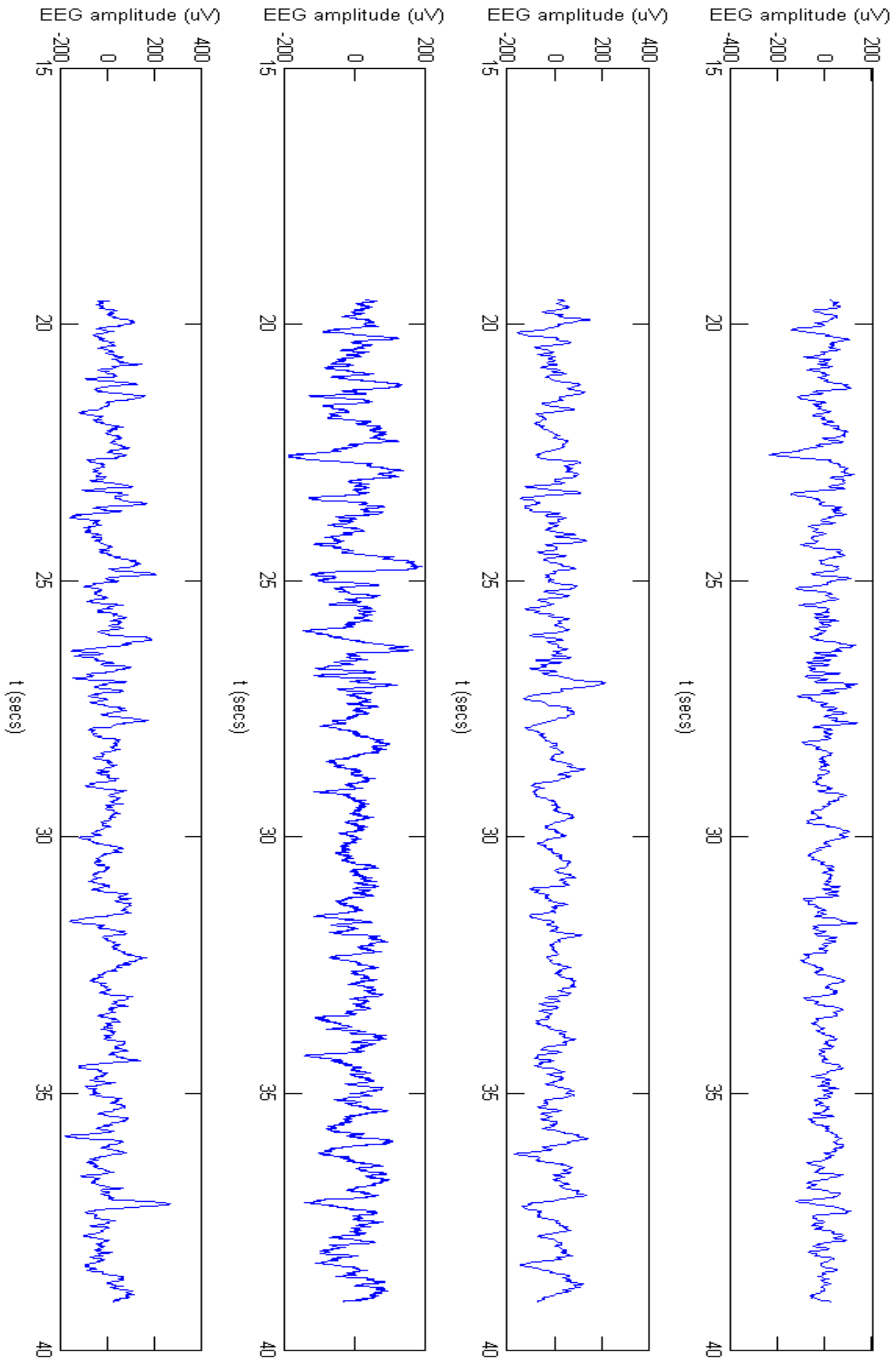


Figure 1 Typical EEG signal measured from 4 different electrodes

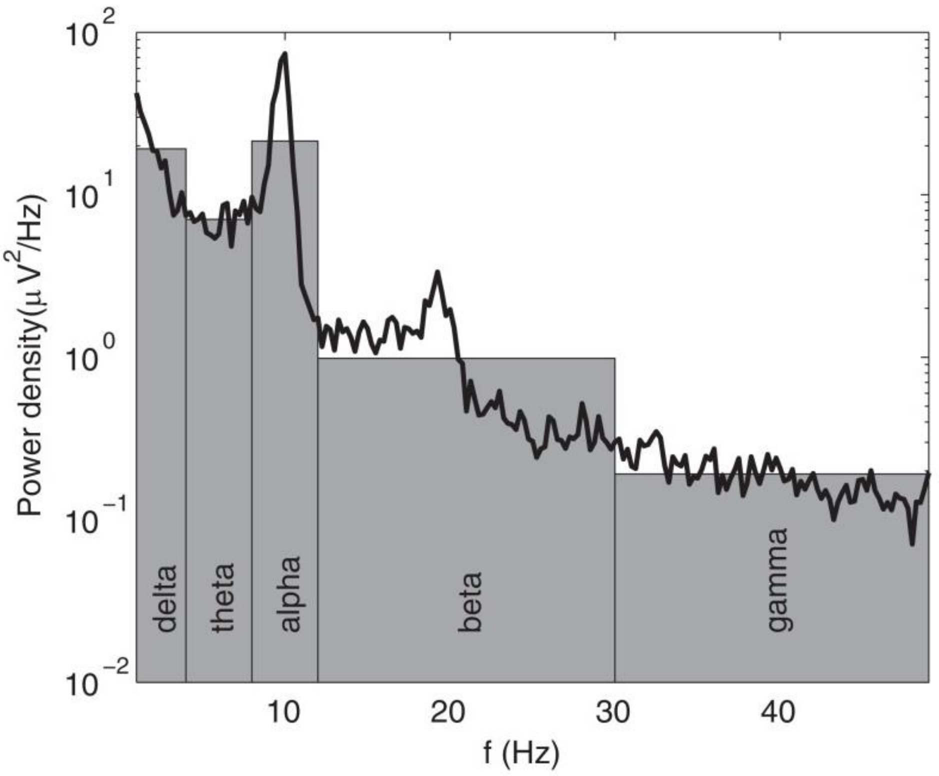


Figure 2 - EEG frequency spectrum bands.

2.4. Automatic seizure detection system

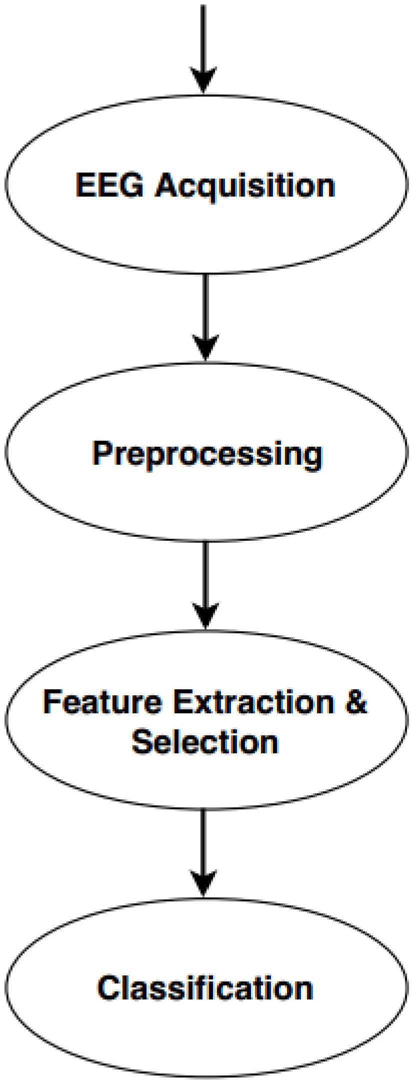


Figure 3 - Automatic seizure detection system block diagram.

Figure 3 shows the block diagram of the automatic seizure detection system. The system mainly consists of 4 stages.

2.4.1. EEG Acquisition

The first stage is the Multi-channel EEG signal acquisition. In this stage, Different electrodes are used to sense and measure EEG signals from different spatial positions on the skull or the cortex. The efficiency of the electrodes affects the overall performance of seizure detection greatly. The positioning of the measurement electrodes on the skull follows different standards. One of these standards are the 10-20 system shown in Figure 4.

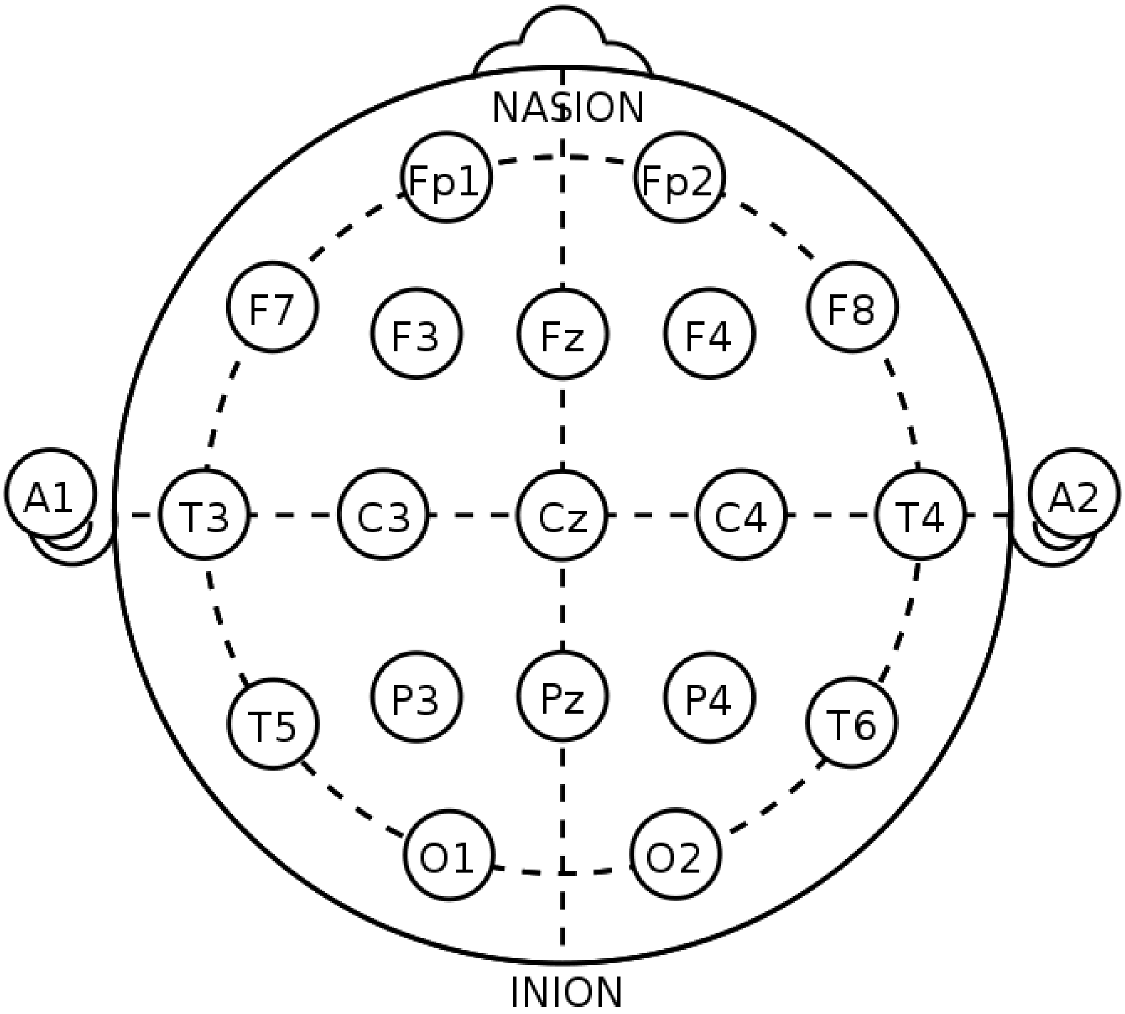


Figure 4 - 10-20 system for EEG measurement.

2.4.2. Preprocessing

The second stage is preprocessing. In preprocessing stage, the raw EEG data measured by electrodes are prepared for analysis and processing. The preprocessing stage includes filtering the signal and only keeping the frequency range of interest. The preprocessing also includes removing artifacts. It also includes normalizing the EEG data to be at the same level of the other signals measured by other equipment or from other patients.

Normalization means that data are converted to a form that is compared to all the other data measured using different measurement equipment or from different patients. For instance, if two different measurement systems are used, the EEG signal of each system would be different. The first system's EEG amplitude may vary from 0 to 15 μV. While the second system's EEG amplitude may vary from -10 to 10 μV. These different EEG signals cannot be directly compared. Hence, all measured EEG data are normalized to the same range from -1 to 1. Then, all EEG signals from different measurement devices and different patients can be compared. The normalization process is done through two steps. First, removing the mean value of the EEG signal. Then, scaling the EEG signal by dividing it by its standard deviation. This normalization techniques should be done again after feature extraction phase.

Artifacts are generated due to different sources. Some artifacts are originated due to movement like eye blinks. Other artifacts are originated due to errors and noise in the measurement devices. Moreover, power line artefacts reside between 50 and 60 Hz depending on the power frequency used in the country. Dealing with the artifacts is performed using several methods. First, some artifacts are ignored as their effect on the features extracted are minor. Second, some artifacts are rejected. The time epoch or frequency domain of this artifacts are excluded from the analysis. Finally, some artifacts are removed from the signal using filters to eliminate specific frequencies using different types of filters: high-pass, low-pass, band-pass and band-stop filters. As many research has proved that most brain EEG power spectral is found between 3 and 30 Hz as shown in Figure 2. Libenson et al. proved that the EEG signals do not exceed 40 Hz [16]. Hence, Blanco et al. proposed using a low pass filter with a cut-off frequency equals to 40 Hz [17]. Preprocessing is the process in which the EEG is prepared for analysis. The signal processing in this area involves the removal of unwanted aspects, such as artifact and high frequency content, and normalizing the EEG data so that it is comparable to all other data (e.g., normalize the amplitude range, sampling frequency, etc).

2.4.3. Feature extraction

The third block is feature extraction and selection. In this stage, different discriminating features are extracted from the EEG signal to differentiate between seizure and non-seizure intervals. Multiple features are used together as an input to the classifier. The appropriate choice of the discriminating features is the key of the classifier performance.

The features are extracted from different domains: time domain, frequency domain and time-frequency domain. The EEG signal is divided in time into several time epochs as shown in Figure 5. In each time epoch, the values of the features used are extracted. If the feature used is a time domain feature, the feature is extracted directly from the EEG signal. If the feature used is a frequency domain feature, FFT is adopted first to get the frequency domain of the EEG signal. Then, the used feature is extracted from the frequency domain of the EEG signal. Finally, if the feature used is a time-frequency domain feature, a Wavelet transform is adopted first on the EEG signal. Then, the feature is extracted from the calculated time-frequency domain.

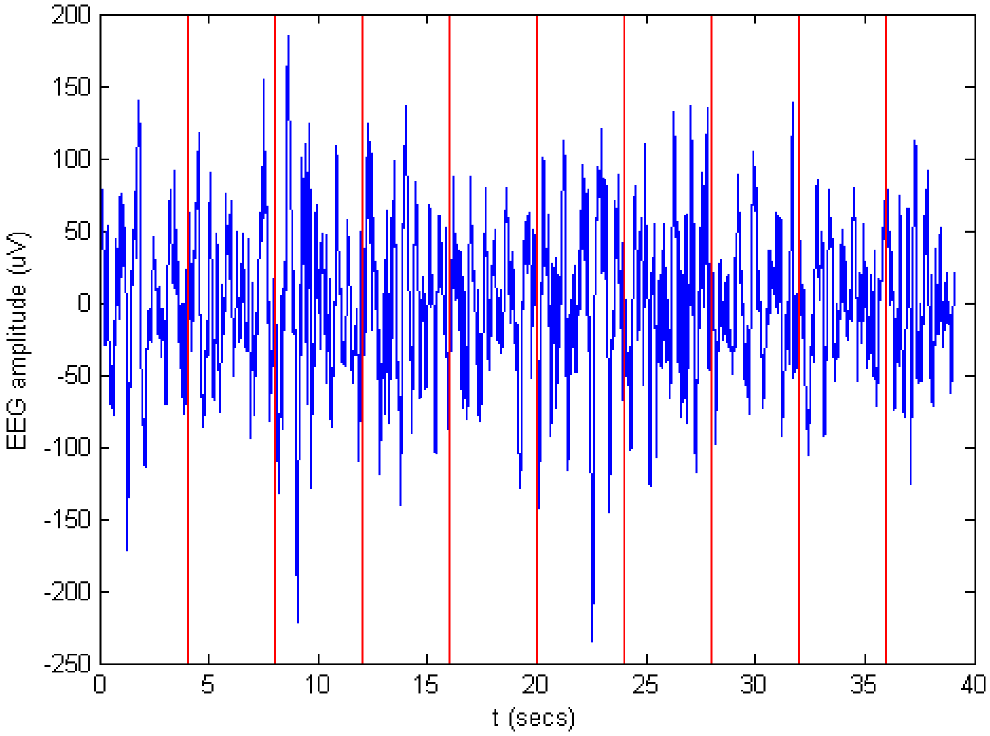


Figure 5 - EEG signal divided into time epochs= 4 secs.

A wavelet transform (WT) is used to represent any signal in multiple wavelets. It helps to represent the signal in time-frequency representation.

2.4.4. Classification

After discriminating features are extracted from the EEG signal, these features need to be judged to detect the existence of seizure. Taking a decision of seizure existence is made based on several methods.

The old-fashioned method is comparing each feature value to a pre-determined threshold. If the value of the feature in a time epoch exceeds the threshold, the system detects a seizure in this time epoch. This method did not achieve an acceptable performance for many reasons. First, choosing the threshold value for each feature is a very challenging task as this value is the main key of the overall performance. Second, the chosen value of the threshold is not constant for all patients and in all conditions. This is due to the fact that the range of normal EEG signal changes from patient to another. Also, the EEG signal range changes with the status of the person. For example, the EEG for the same person varies during sleeping, eye blinking or doing sports.

To overcome this problem, many researches proposed to use machine learning techniques that will be discussed in the next section.

2.5. Machine learning

Machine learning (ML) is the science of making the computers able to learn themselves by their own from observing large number of examples. Machine learning is not a newly invented science. ML has been proposed by Arthur Samuel from 1949 through late 1960s [18]. He explicitly defined ML as it is known today at 1959 [19]. In ML, many statistical studies are performed on a very large amount of data. Recently, machine learning and artificial intelligence become very hot topics for all software and hardware researchers. This is due to the great growth in the computational capabilities. Nowadays, ML is playing a great role in many fields.

ML techniques are classified into different categories as follows:

- **Supervised learning**

  In supervised learning, the task is to find a function to map any new input to the corresponding output based on some training points. Each of the training points is described by their input value and their associated labels or outputs. The input-output relation is deduced from the training example. Then, this relation is used to find the output of any new input test point even if this new point is totally unseen in the training examples. Figure 6 shows an example of the supervised learning problems. In this example, multiple training points from two different groups are given. One group is represented by the red circle while the other group is represented by the blue circle. For each training example, a point is drawn on the x-y plane based on its corresponding label (group). The task of the problem is to find the separable line. After finding the line, any new point is represented on the x-y plane. Then, the type (group) of this point is determined based on its location relative to the line.

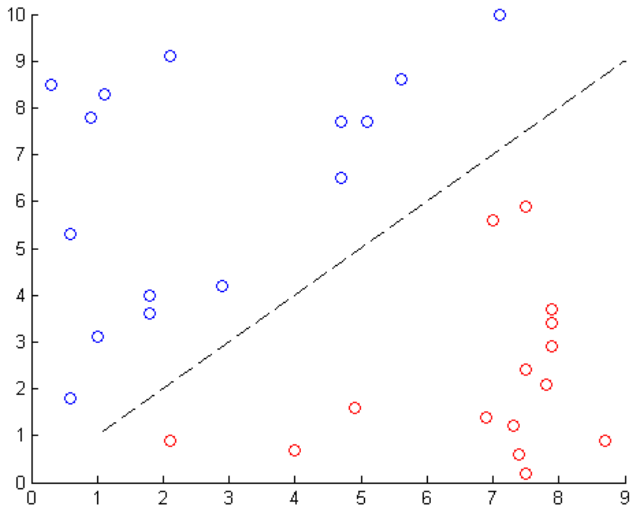


Figure 6 - Supervised learning example.

- **Unsupervised learning**

  In unsupervised learning, the task is to find a function to map any new input to the corresponding group based on some training points. In other words, the task of the unsupervised learning is clustering and categorizing. Each of the training points is described by their input value only and all the points are unlabeled. Hence, in the training phase only the similar training points are clustered in one group. Then, any new testing point is attached to one of these groups. Figure 7 shows an example of the unsupervised learning. In this example, multiple training points are given. All the training points are unlabeled; only their input value are given but their outputs are not. All the points are represented by the same symbol on the x-y plane. The task of the unsupervised learning is to cluster these points into two groups based on their values. After finalizing training and finding the separable line between the two groups, any new test point can be classified into one of the two groups.

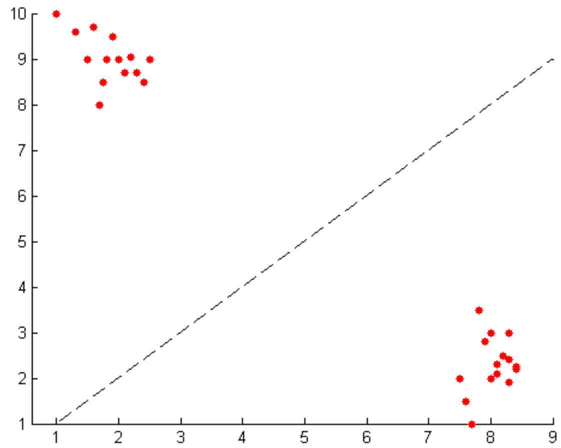


Figure 7 - Unsupervised learning example.

- **Reinforcement learning**

  In the reinforcement learning, the computer interacts with a changing environment, its behavior towards this environment is assessed by some reinforcements. These reinforcements are either rewards or punishments.

For the work proposed in this thesis, supervised learning is the type used as the EEG data is labeled. Different supervised learning techniques are used and compared.

Consider a supervised problem is formulated as follows:
A training data set is given as pairs of input-output points

$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_{n-1}, y_{n-1}), (x_n, y_n)\}$$

The supervised learning's task is to fit a function that maps the inputs x_i to their corresponding outputs y_i. The supervised learning problems are classified into 2 categories based on the range of y_i. If y_i is a real number, the problem is called a regression problem. For example, having a database of prices of different apartments with different areas and predict the price of any apartment of a specific area is a regression problem as the price may take any real number. The second group of supervised learning problems is classification problem where y_i may take only one of discrete set of values. In both groups of problems, the task is the same; finding a function that relates the output to the input. If the performance achieved by a specific function is too low when tested on the training examples, a higher order function should be used. However, the performance may be great on the training data only and is very low for any new testing data point. This problem is a well-known problem in machine learning which is called over-fitting. The problem of over fitting is caused due to:

1- Very complex model: in this case a very complex function is used to fit simple data. The solution in this case is to use a lower order function.
2- Few training examples: the second reason of the over-fitting problem is using a few number of training examples. Hence, adding more training examples to the dataset may solve the problem of over-fitting.

The proposed work is in the field of seizure detection. Hence, supervised learning is the most important machine learning type used. The problem of seizure detection is a classification problem as the output is only one of 2 groups: seizure and non-seizure.

2.6. Dataset

The database used in this work was collected at the Children's Hospital Boston (CHB) by a team of researchers from the Massachusetts Institute of Technology (MIT). The dataset consists of EEG recordings from subjects with intractable seizures. The AEDs doses are stopped for several days. Then, the researchers monitored the patients for multiple days. The signals are recorded from different patients with different age and sex as shown in Table 1. Noting that Chb01 and Chb21 are the same female patient but after 1.5 years.

Each case of the 23 case has 9 up to 42 .edf files. These .edf files are almost continuous with a very limited cuts up to 10 seconds when the EEG signals are not recorded due to some hardware limitations. Moreover, all the protected health information of the patients are preserved and deleted from the .edf files. Even the absolute date of each record has been changed with another one but the relative time and date of the same patient remained constant. Each .edf file contains the data of almost one hour for the patient. Beside the .edf files, a .txt file is available for each patient. This .txt file contains information about the different epileptic seizures of this patient that happened during recording and the specific time of start and end of each seizure.

2.7. Performance Metrics

The performance of the system is measured through different performance metrics that are widely used especially in neural seizure detection. These metrics are accuracy, specificity and sensitivity of the classifier. The sensitivity is the true positive rate or the percentage of seizure that could be detected successfully by the classifier and could be calculated as follows:

$$Sensitvity = \frac{TP}{TP + FN}$$

The specificity is the true negative rate or the number of non-seizure epochs detected successfully by the classifier and could be calculated as follows:

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

Where TP denotes true positives,
TN denotes true negatives,
FP denotes false positives,
FN denotes false negatives.

There is always a trade-off between sensitivity and specificity. As sensitivity increases, specificity decreases and vice versa. Hence, a combining performance metric is defined which is called accuracy. Accuracy means the percentage of the right decisions to the total decisions made by the classifier. Accuracy can be calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Table 1 - CHB MIT patients.

Case	Gender	Age
Chb01	Female	11
Chb02	Male	11
Chb03	Female	14
Chb04	Male	22
Chb05	Female	7
Chb06	Female	1.5
Chb07	Female	14.5
Chb08	Male	3.5
Chb09	Female	10
Chb10	Male	3
Chb11	Female	12
Chb12	Female	2
Chb13	Female	3
Chb14	Female	9
Chb15	Male	16
Chb16	Female	7
Chb17	Female	12
Chb18	Female	18
Chb19	Female	19
Chb20	Female	6
Chb21	Female	13
Chb22	Female	9
Chb23	Female	6

2.8. Previous Work

As explained in the introduction, Epilepsy is a very dangerous disease that affects quality of life of its patients. Due to the large number of epilepsy patients, a great effort is done in treatment of the epilepsy especially using electrical stimulation. The work done to detect seizure using EEG includes many methods: single channel or multi-channel [20]. In single channel based seizure detection systems, it is required to choose the appropriate channel that is the nearest to the seizure focus. This type is mainly used in focal seizures. The process of choosing the channel is performed by measurement of different channels and choose the best performance channel. Another solution is to use all the measured and available signals, and detect seizure based on the EEG signals from multi-channel [21].

After the EEG measurement is done, many research is done on preprocessing. Wackermann et al. used several EEG analysis methods to characterize the sleeping effect of EEG [22]. Another source of artifacts is the eye movement and blinks. The electrical activity accompanied with the eye movement is strong enough to be recorded with EEG. The amplitude of the eye movement artifact is larger than that of the background EEG activity so many research is done in the area of removing eye movement effects [23]. Moreover, many work is done to remove muscles moving artifacts such as that done by Van Boxtel et al. [24].

2.8.1. Feature extraction and selection

Many work is done on the analysis of EEG signals for seizure detection in the literature. Features extracted from EEG along with different machine learning algorithms are used to detect seizure. Yuan Q. et al. used nonlinear feature extraction strategies such as approximate entropy and Hurst exponent and got 93.75% and 79.75% sensitivity respectively [25]. Also, nonlinear feature extraction strategies were used in multiple papers [26], [27], [6]. Li. et al. got a sensitivity ranging from 82.75% to 97% based on the combination used [26]. Panda. et al. got 91.2% classification accuracy [27] and Kolekar et al. got 81.67%, 91.25% and 82.22% accuracy for different classification strategies [28]. Support vector machine (SVM) is used in many of these papers with Radial Basis function (RBF) kernel for classification [25], [26], [27], [28]. Generally, the results obtained through SVM with RBF kernel are usually more accurate, however a hardware implementation for an RBF kernel consumes much more power than linear and polynomial kernels.

2.8.2. Hardware implementation of SVM training accelerators

Many research has been done in implementing hardware implementations and accelerators for SVM training [29]. Keerthiet al. proposed a parallel implementation of multiple CPUs for processing partitioned data sets [30]. The use of multiple CPUs leads to increase the overall performance. One the other hand, it greatly increases the power consumption. Caoet et al. developed a hardware implementation of SVM training circuit using MATLAB HDL coder [31]. The performance degraded due to the lack of optimizations. Chih-Hsiang et al. proposed a re-configurable chip with SMO-based SVM training [32]. The proposed architecture decreased the routing overhead, accelerated kernel function update and used pipelining. However, some hardware usage and training speed problems have appeared. Lazaro et al. proposed a hardware-software architecture to speed up SVM training using SMO. As the dot product takes most of calculation time in SMO, it is chosen to be implemented on hardware [33].

Jhing-Fa et al also proposed a HW/SW co-design solution for multiclass SMO training [34]. A hardware-software co-design system for accelerating the SVM learning phase was presented based on another decomposition algorithm instead of the common SMO algorithm [35]. M. Rabieah et al proposed a complete FPGA-based system for nonlinear SVM learning using ensemble learning [36]. S. Wang et al proposed a FPGA-based reconfiguration framework to speed up the online LS-SVM training [37]. However, the block RAM usage and reconfiguration efficiency are the main challenges. In this paper, more work is done in the area of training the SVM classifier to have better results without the need to have complex transformations or complex kernel functions like those proposed in [38], [39], [40].

Chapter 3 : Design of Feature Extraction and Selection

The feature extraction step is a very important step in automatic seizure detection systems. In feature extraction step the discriminating features are extracted from the EEG signal. These features should differentiate between different phases of the EEG signal. Several features are proposed and used in literature to detect seizure. The extracted features are extracted can be categorized depending on the domain from which they are extracted as follows:

1- Time domain features
2- Frequency domain features
3- Time-frequency domain features (Wavelet)

The features extracted from EEG signals can also be categorized into 2 different groups: linear and non-linear features.

3.1. Linear Features

Different linear features are implemented, extracted and tested. The 11 linear features are as follows:

- Mean Absolute Value (MAV)

$$MAV = \frac{1}{N}\sum_{i=1}^{N}|x_i|$$

- Root Mean Square (RMS)

RMS was used combined with other features for seizure prediction in [41]. RMS is calculated as follows:

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}$$

- Standard Deviation (SD)

Standard Deviation is a measure of the average deviation from the mean. It was used in [42] and achieved high performance. SD can be calculated as follows:

$$SD = \sqrt{\frac{\sum_{i=1}^{N}(x_i - mean(x))}{N-1}}$$

Where $mean(x) = \frac{\sum_{i=1}^{N}x_i}{N}$

- Variance

Variance is the standard deviation raised to the power of two. It is easier to calculate the variance rather than calculate SD. Hence, both SD and variance are tested to check if easier calculation would reflect on the performance or not.

- Maximum Absolute Value

Calculating the maximum absolute value for every epoch of time. It was used in [42] with other features achieving performance more than 98%.

- Minimum Absolute Value

Calculating the minimum absolute value for every epoch of time.

- Average Energy

In epileptic seizures, the amplitude and frequency of the EEG signal increases. This was a motivation to include the average energy of the epoch as a feature. It is defined as follows:

$$E = \sum_{i=1}^{N} x_i^2$$

- Fluctuation Index (Coastline)

Fluctuation Index (FI) measures the fluctuation in the signal. During seizure periods, it is found that EEG exhibits high fluctuations relative to non-seizure periods. FI is defined as follows:

$$FI = \sum_{i=1}^{N} |x_{i+1} - x_i|$$

- Hjorth parameters: Mobility

Mobility is the square root of the variance of the first derivative divided over the variance of the signal.

- Hjorth paramteres: Complexity

Complexity represents the change in frequency with respect to a pure sine wave

- Skew

Skew measures how non symmetric the data is. It was used with other features for classification by Zhang [42]. It is calculated as follows:

$$Skew = \frac{1}{M} \sum_{i=1}^{N} \left(\frac{X(w) - \mu_w}{\sigma_w} \right)^3$$

Where $X(w)$ is the sample value at frequency domain,
μ_w is the mean value of the samples at frequency domain,
σ_w is the standard deviation of the samples at frequency domain.

- Kurtosis

Kurtosis is the same as skew but raised to power 4 as follows:

$$Kurtosis = \frac{1}{M}\sum_{i=1}^{N}\left(\frac{X(w)-\mu_w}{\sigma_w}\right)^4$$

3.2. Nonlinear Features

Non-linear analysis of EEG signal exhibit description of the non-stationary nature of the signals. Different features are used by different researchers in the literature. They used many features from information theory, nonlinear dynamical analysis, and stochastic processes analysis. Non-linear features showed promising results in both detection and prediction for epileptic seizures [43]. In this study, different nonlinear features are examined as follows:

- Approximate Entropy (ApEn)

Approximate entropy is a probabilistic method developed by Steve M. Pincus [44]. It measures how ordered or disordered a given EEG signal is. A small output value indicates regularity in the input EEG signal, and on the contrary, as the EEG gets more irregular, the higher the output value becomes [45]. The dataset is divided into overlapping subsequences.

$$S(i) = [x(i), x(i+1), \dots\dots, x(i+m-1)]$$

Where $i = 1,2, \dots., N-m+1$,
m is the length of each subsequent.

Then, the algorithm searches for matched patterns by calculating the distance between each subsequent and all other subsequences. Finally, it compares this distance with a certain tolerance r. If the distance is less than the tolerance, the patterns are considered matched which supports the decision of having a regular predictable EEG and vice versa. A distance function $d[\,x(\,i\,)\,, x(\,j\,)\,]$ between each subsequent and every other subsequent is calculated first. Then, the correlation $log\ C_i^m\ (\,r\,)$ is calculated by counting the distances that are smaller than a tolerance r and then divided by the number of subsequences $N - m\ +\ 1$. Finally, the logs of these values are summed together and formulating approximate entropy as follows:

$$\varphi^m(r) = \frac{1}{N-m+!}\sum_{i=1}^{N}\log(C(r))$$

Finally the approximate entropy can be calculated as follows:

$$ApEn = \varphi^m(r) - \varphi^{m-1}(r)$$

- Shannon Entropy

Shannon entropy is a measure for information that the system exhibits. It estimates the number of bits required to encode a string of symbols based on their frequencies [46]. Continuous values of EEG signals are quantized. Then, the frequency of each symbol is calculated to get Shannon Entropy as follows:

$$H(x) = -\sum_{i=1}^{N} P(x_i).\log(P(x_i))$$

Where $P(x_i)$ is the probability of the symbol x_i .

- Permutation Entropy

Permutation entropy, as other entropies, measures how disordered the EEG signal is. However, it is computed independent of the values of the samples. First, a mapping function is applied to generate windows of length n. Probability of a given permutation is given as:

$$P(\pi) = \frac{\#\ of\ windows\ permutation\ \pi}{T-n+1}$$

$$H_n^* = -\sum P(\pi).\log\big(P(\pi)\big)$$

- Renyie Entropy

Renyie entropy generalizes Shannon entropy as the parameter α gives an extra degree of freedom for the distributions. It is calculated as follows:

$$H(x) = -\frac{1}{1-\alpha}\log(\sum_{i=1}^{N} P_i^{\alpha})$$

- Hurst Exponent

Hurst Exponent is a measure of whether the data is pure white noise or it contains information. If H is equal to 0.5, then the time series is purely random. However, if it is larger than 0.5, then it contains some trends. It is calculated for a given time series with length t from the rescaled range series (R/S) which is calculated from the standard deviation S and the range series R. Finally, a line fitting is done between $log(R/S)$ and $log(T)$ to get the Hurst exponent value [47].

Where R is the maximum deviation from the mean and the minimum deviation from the mean, S is the standard deviation, $\frac{R}{S}$ is the rescaled value and T is the sample duration.

- Modified Hurst Exponent

The Hurst exponent is the slope of the linear fit of the log-log graph. Another simpler implementation for the Hurst Exponent was using the below equation.

$$H = \frac{\log(\frac{R}{S})}{\log(T)}$$

In this implementation it is assumed that this linear fit will always pass through the origin.

- Fractal Dimension

Fractal Dimension (FD) is based on fractal geometry. Higuchi's algorithm with k= 5 is used to calculate the fractal dimension [48].

3.3. Simulation Setup

A software implementation of all proposed features discussed is done using MATLAB2016a. Different combinations of the 20 proposed features are used and tested along with linear kernel SVM. The performance metrics -sensitivity, specificity and accuracy- are extracted from each combination and compared.

The procedure to get the best performing combination could be built using two methods

1- All-in then backward elimination according to p-value:
This method is done by extracting all the proposed features and testing the performance. Then, a trial to eliminate one of the features is performed. The task is to choose the first features that will be eliminated. The features that will be eliminated is the one that has the minimum effect on the performance metrics. Then, this step is repeated until having the minimum number of features that achieve an acceptable performance.

2- Trying all possible combinations for a fixed number of features:
This method is done by choosing constant number of features in each combination. Then, all the combinations between the proposed features are tested and for each combination the performance metrics are calculated. Then, the best performance combination is chosen.

In this work, the second solution was adopted. The decision was made to use three features in each combination based on many work done in the literature [25], [27], [28]. A total of 1140 combinations are tested and compared.

A MATLAB script is developed to test the combinations between the features one by one. Each combination consists of 3 features. The code chooses one of these combinations and extract the corresponding features from all training and testing data. Then, the code trains a linear kernel SVM using the extracted features. Then, the test data points are tested on the resultant hyperplane. Finally, the performance metrics are calculated and written to the output file. For each combination a line is written to the output file containing the features of this combination, the resulting sensitivity, specificity and accuracy.

The output needed from the simulation is to find a combination of 3 features that make the data points linearly separable. If such combination of feature is found, it will achieve a very high performance using linear kernel SVM. That will save great punch of energy as the linear kernel consumes energy less than any other type of kernel functions such as polynomial and RBF kernel.

3.4. Simulation Results

The visualization of data points with different extracted features can give a good evidence of the great effect of feature selection on the performance of the classifier. Figure 8 shows the training data points when the selected features are Hjorth mobility, Hjorth complexity and maximum absolute value. The figure shows the objection of the data points on the plane of each 2 features where feature 1 is the Hjorth mobility, feature 2 is Hjorth complexity and feature 3 is the maximum absolute value. It is clear from the figure that these features are not linearly separable. When trying these features with linear kernel SVM, the performance achieved is 0 % sensitivity, 100% specificity and 99.7% accuracy which means that the classifier classify all points into non-seizure.

Figure 9 shows the training data points when the selected features are Hurst exponent, average energy and minimum absolute value. In this figure, feature 1 is Hurst exponent, feature 2 is average energy and feature 3 is the minimum absolute value. Some data points can be linearly separable especially in the plane of Hurst exponent and average energy. The performance achieved by these features is: 62.9% sensitivity, 98.8% specificity and 98.7% accuracy.

Figure 10 shows the training data points when the selected features are Fractal Dimension, Hurst Exponent and Coastline features where feature 1 is fractal dimension, feature 2 is Hurst exponent and feature 3 is coastline. It is clear that all the data points are almost separable in all planes. That's why the achieved performance becomes: 96.77% sensitivity, 97.9% specificity and 97.9% accuracy.

After finalizing the simulations of the all 1140 combinations and by analyzing the detailed results shown in Appendix B, it is noticeable that the minimum specificity achieved is 96.4% and the maximum specificity is 100%. Hence, the specificity achieved from all features’ combinations are acceptable. So, the specificity is not the key performance metric to choose the best combination. On the other hand, the sensitivity ranges from 0% to 96.77%. To be able to analysis and visualize these results, the combinations are grouped into multiple groups based on their sensitivity value. The number of features of each group are shown in Figure 11.

The combinations of interest are those which have sensitivity more than 90%. To analyze these combinations, the number of repetition of each feature in these combinations are counted. Then, the features are sorted by their repetition counts from largest to smallest as shown in Figure 12.

It is found that fractual dimension is a very important feature as it exists in all the features’ combinations that have sensitivity more than 90%. Moreover, the best combination –the one that gives the maximum performance- is the combination of fractual dimension, Hurst exponent and coastline. This combination achieves sensitivity equals to 96.77%.

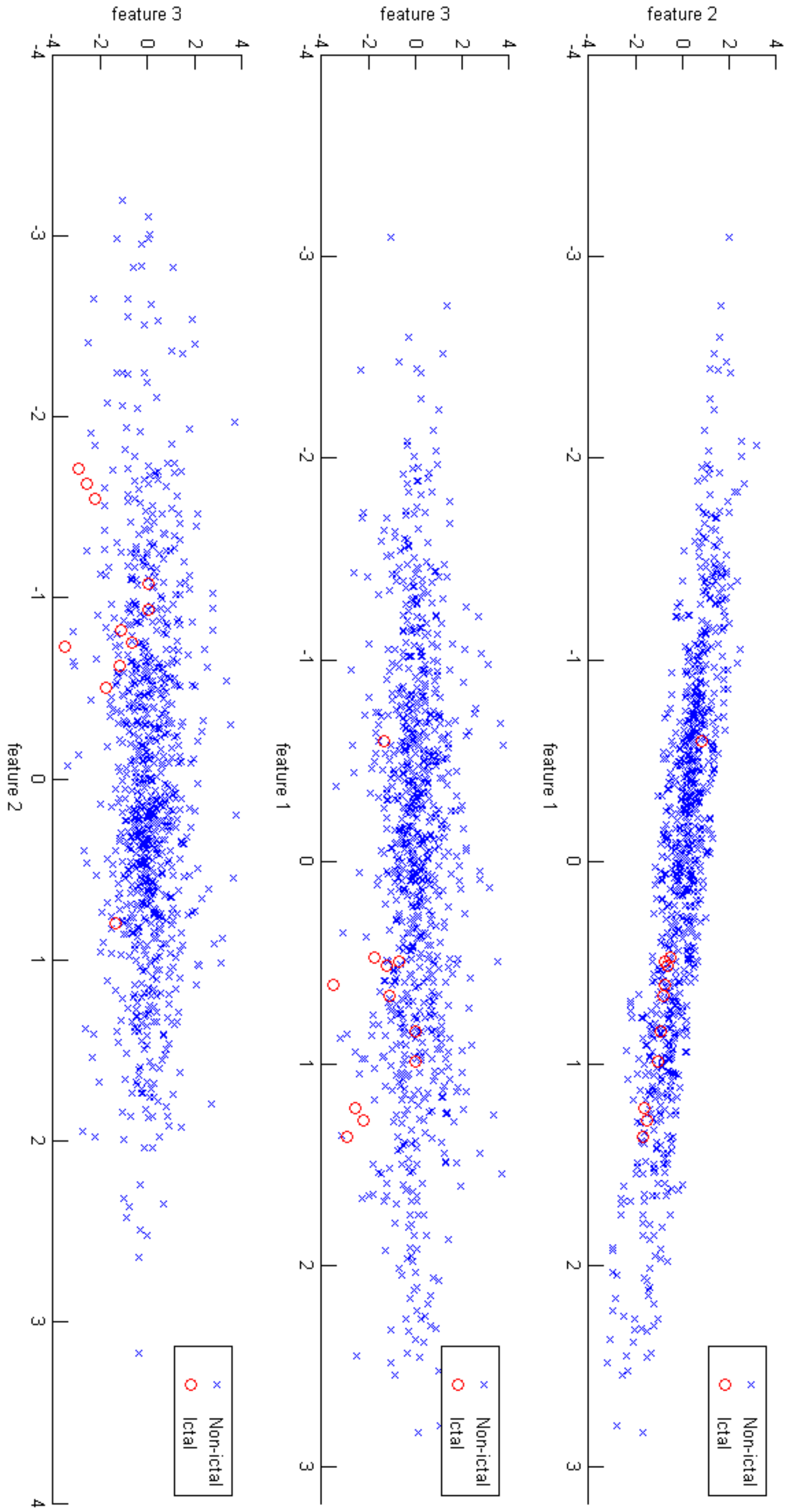


Figure 8 - Training points for Hjorth mobility, Hjorth complexity and Maximum absolute value features.

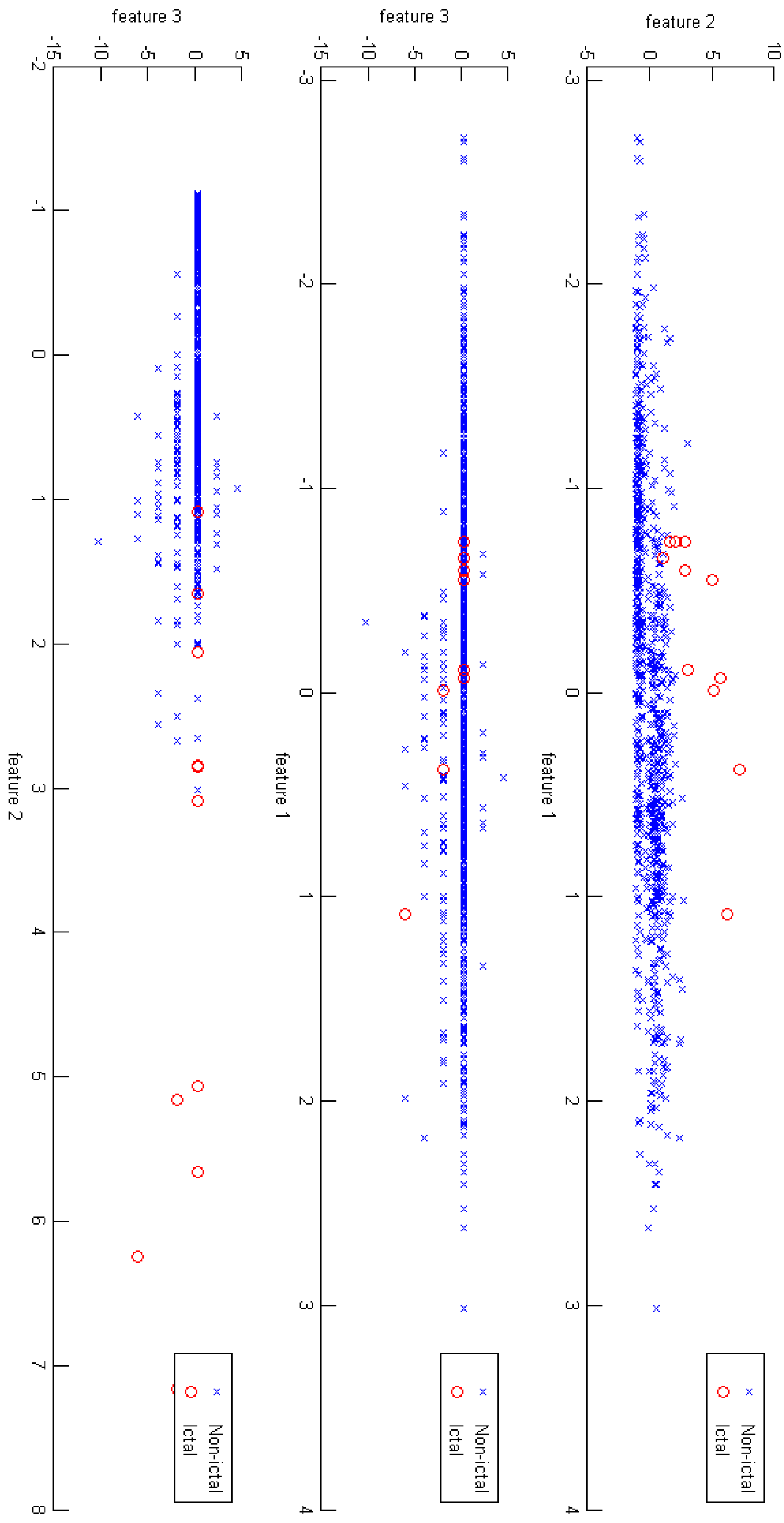


Figure 9 - Training points for Hurst exponent, average energy and minimum absolute value features.

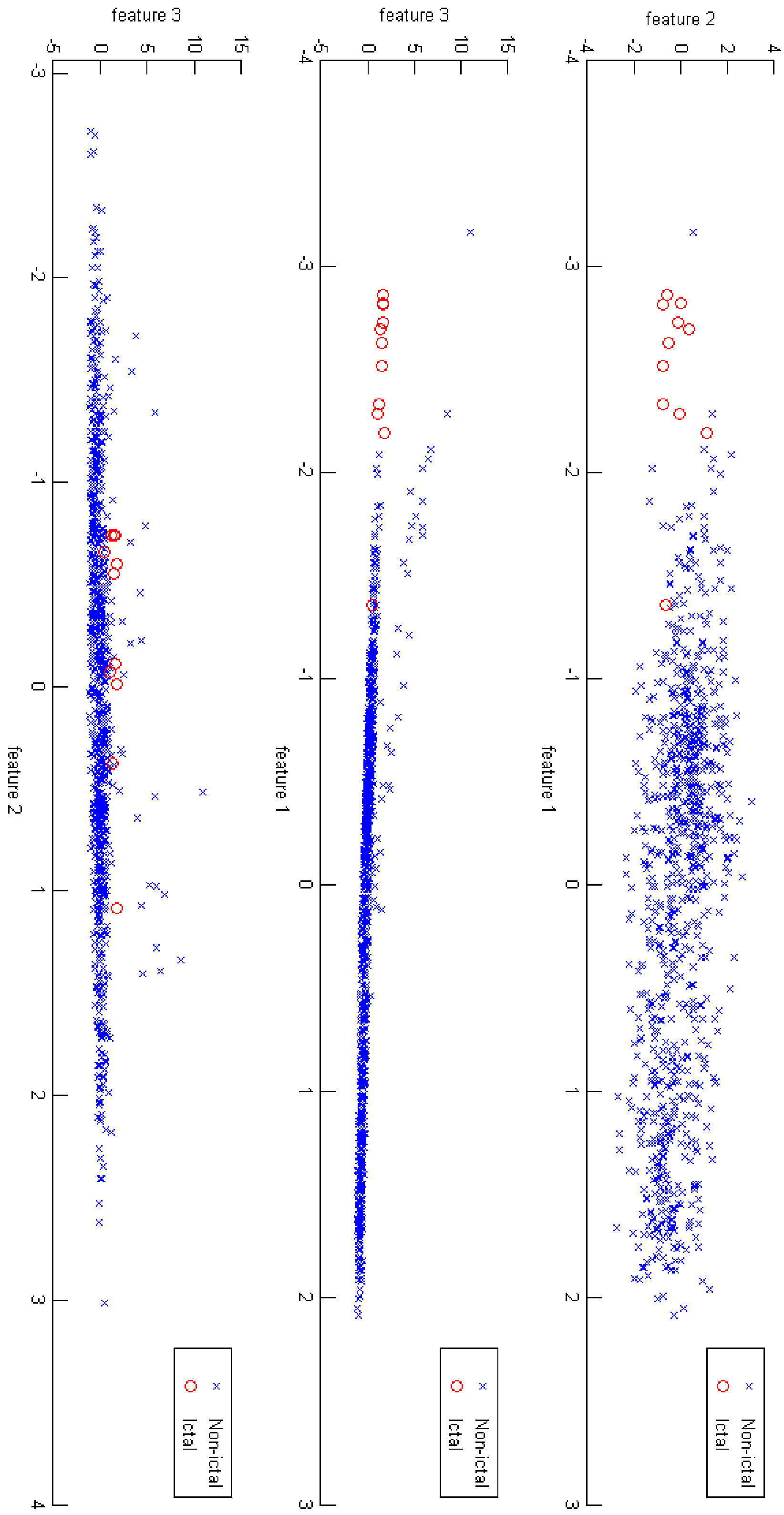


Figure 10 - Training data points of Fractal Dimension, Hurst Exponent and Coastline features.

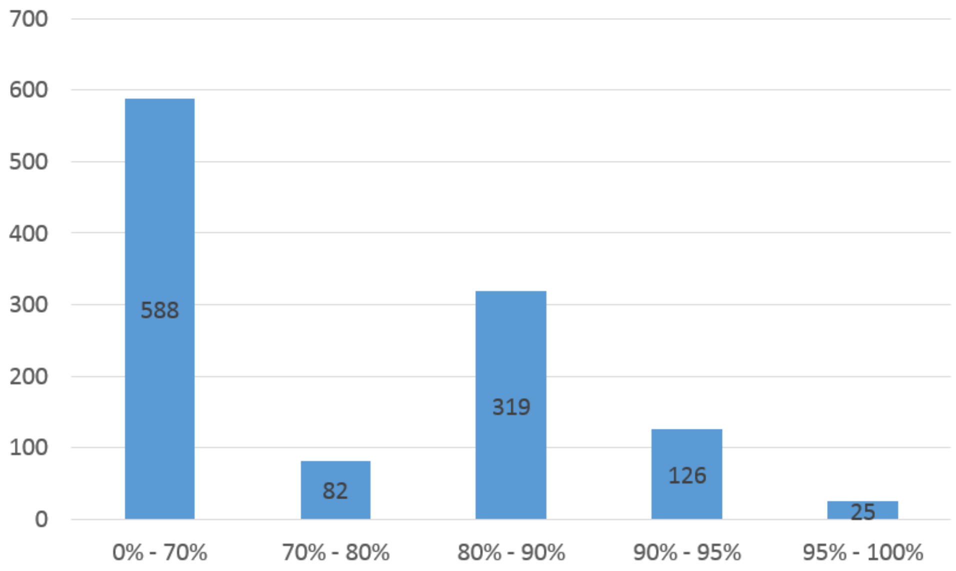


Figure 11 - Number of features' combinations in each range of sensitivity.

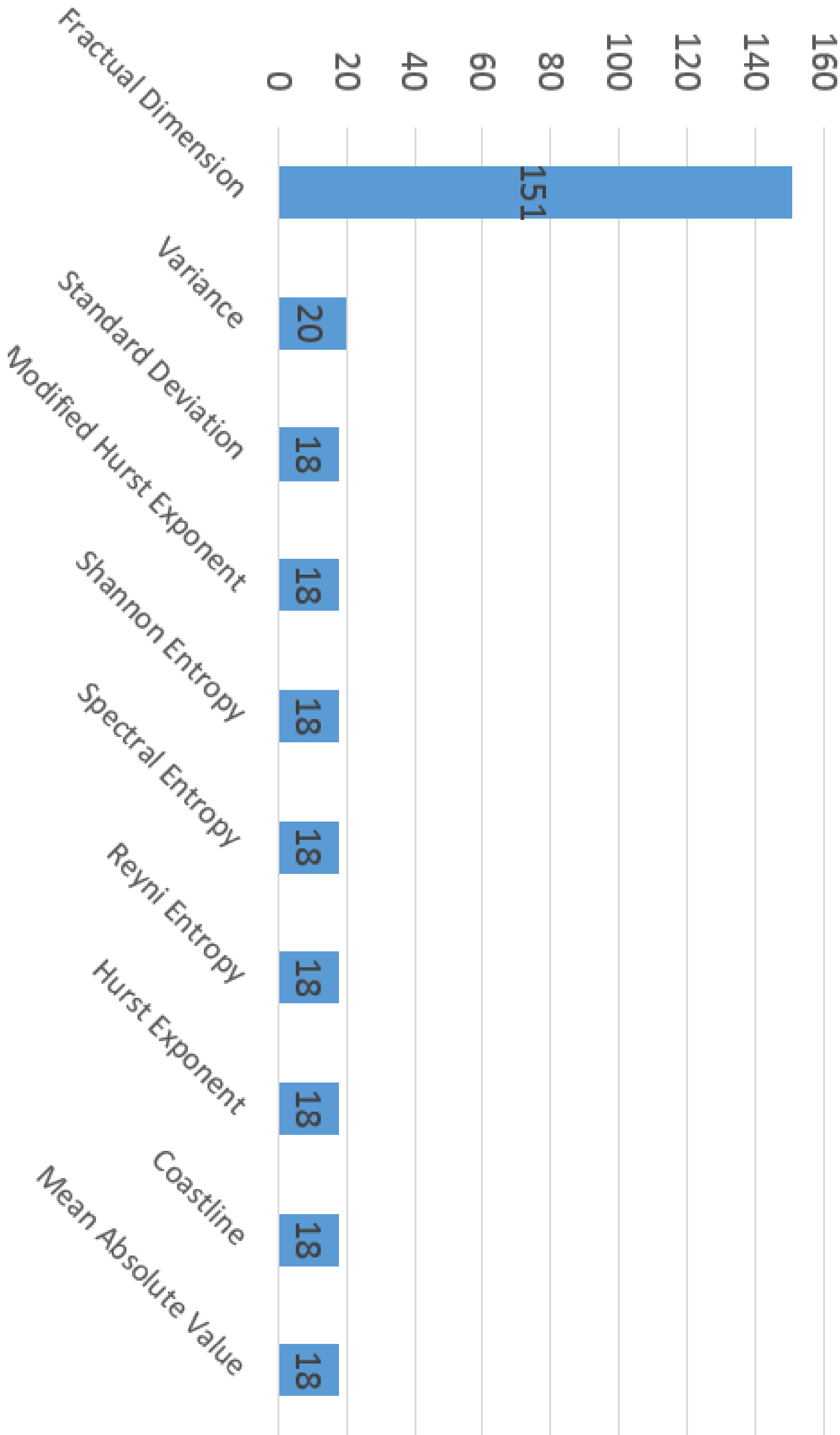


Figure 12 - number of incidence of each feature in the combinations with sensitivity >90%.

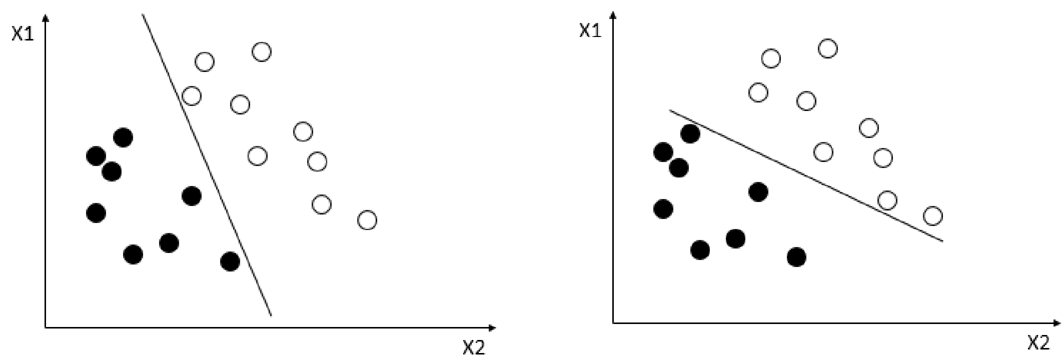
Chapter 4 : Design of Support Vector Machine Training Accelerators

4.1. Support Vector Machine

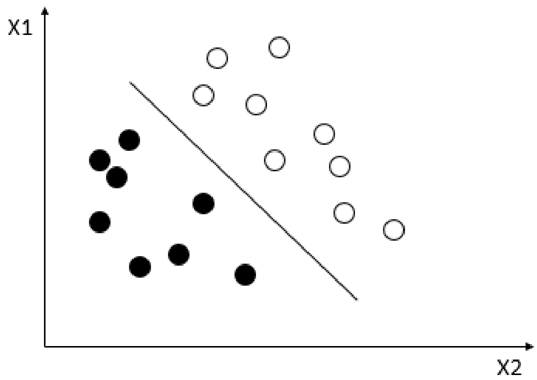
Support Vector Machine (SVM) is a supervised machine learning and classification model that is gaining much attention of researchers in statistical classification and regression analysis problems. SVM is widely used in many applications such as face detection, handwriting detection and bioinformatics [49]. SVM was first introduced by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 [50]. SVM uses a set of training examples categorized into 2 or more groups. SVM works in two main phases: training phase and classification phase.

Training in SVM is a process in which a hyperplane that separates two labeled sets of training examples is determined. SVM searches for the hyperplane that gives the largest margin between the two sets. The subsequent step is to classify unlabeled testing examples into one of two classes. Finding the hyperplane is a problem of solving a quadratic programming (QP) problem subject to constraints [51].

The optimization problem has infinite number of solutions. Hence, different hyperplanes can perfectly separate the two different groups in the case of binary classification as shown in Figure 13. All the three hyperplanes in (A), (B), (C) separates the two groups with zero errors. SVM defines the best hyperplane is the one the hyperplane that gives the largest margin between the two sets. Hence, SVM chooses the hyperplane shown in Figure 13-C.



(A) (B)



(C)

Figure 13- Different classification hyperplanes

As mentioned earlier, SVM learns from a training set of N dimensional vectors x_i and their associated classes (labels) y_i. In case of binary classification, $y_i \in \{0,1\}$, i=1,2,…,n. SVM deals with linearly separable data points directly. For the non-linearly separable data points, the non-linearly separable dataset is mapped into a higher dimensional domain in which the mapped data points are linearly separable. As this mapping may contain heavy computing especially with the large number of data points another approach called Kernel trick is used. Kernel methods uses kernel functions to operate in a high-dimensional feature space without the need of calculating the mapping of each data point. Then, SVM finds the hyperplane that gives the largest margin in the new feature space. This hyper plan is defined as follows:

$$w.\varphi(x) + b = 0 \quad (1)$$

Where w is the normal to the hyperplane, $\varphi(x)$ is the mapping function used to map each input vector to the feature space and b is the bias.

The distance from the nearest points to the hyperplane from each side equals to $\frac{2}{||w||}$. Therefore, to choose the hyperplane that maximize the margin, the optimization problem is formulated as follows:

$$min_{w,b} \frac{||w||^2}{2} \quad (2)$$

Subject to $y_i(w.\varphi(x) + b) \geq 1$

This is denoted by hard margin SVM, where the hyperplane perfectly separates the two sets according to eqn.(1). A modified version of SVM introduces a trade-off between the size of the margin and the number of errors in the classification process is given in eqn.(3). This is performed by defining a penalty parameter C. The optimization problem is formulated as:

$$min_{w,b} \frac{||w||^2}{2} + C \sum_{i=1}^{n} \xi_i \quad (3)$$

Subject to:

$y_i(w.\varphi(x) + b) \geq 1 - \xi_i,$

$\xi_i \geq 0$

Where ξ_i is the slack for the i^{th} training point as shown in Figure 14.

The penalty parameter C should be selected carefully for each data set. If C is selected large, the weight of any wrong classified point is very large so the convergence of the problem takes large number of iterations. If C is selected small, some errors are allowed to maximize the margin and get the solution in fewer number of iterations than the large C scenario.

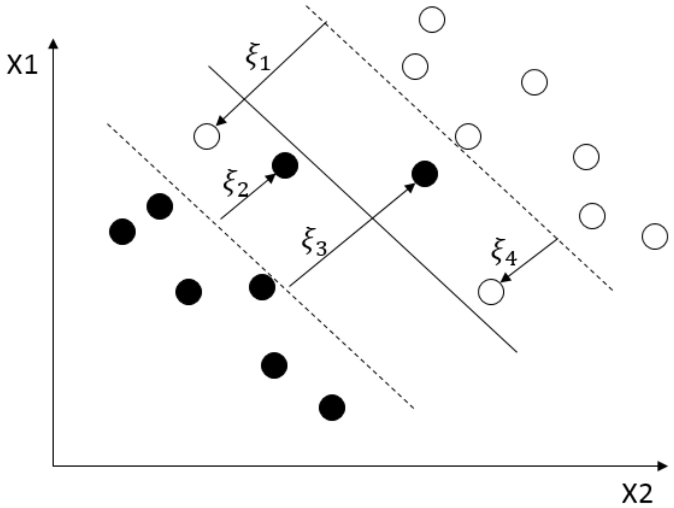


Figure 14- Soft Margin SVM.

The modeled problem is solved using Lagrange multiplier as follows:

$$\min_{\alpha} \psi(\alpha) = \frac{1}{2}\sum_{i=1}^{n} y_i . y_j . K(x_i, x_j). \alpha_i . \alpha_j - \sum_{i=1}^{n} \alpha_i \quad (4)$$

Subject to:

$\sum_{i=1}^{n} y_i . \alpha_i = 0,$

$0 \leq \alpha_i \leq C, \qquad i = 1,2, \dots, n$

Where α is Lagrange multiplier, Kernel functions K. Different Kernel functions are widely used in SVM applications as follows:

Linear Kernel:

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

Polynomial Kernel:

$$K(x_i, x_j) = (x_i . x_j + 1)^d$$

Where d is the polynomial degree

Exponential Kernel:

$$K(x_i, x_j) = e^{-\gamma \left\| x_i - x_j \right\|^2}$$

By solving the problem formulated in eqn.(4), the values of $\alpha_i's$ are obtained. The values of each α is classified into one of the three following classes:

1) $\alpha_i = 0$ represents the correctly classified points outside the margin
2) $0 < \alpha_i < C$ represents the training data points that define the margin

3) $\alpha_i = C$ represents the wrongly classified points and the points that violated the margin (where $\xi_i \neq 0$)

Many techniques are used to solve this QP problem. In this thesis, two training techniques of SVM are tested, hardware implemented and compared. The two techniques are Gradient Ascent (GA) and Sequential Minimal Optimization (SMO). The two techniques' algorithms and hardware implementations are discussed in details in the following sections.

4.2. Gradient Ascent (GA)

4.2.1. Algorithm

Gradient ascent is an iterative optimization algorithm that solves minimization problems. It depends on taking steps towards the minimum point proportional to the slope of the function at the current point. By applying the algorithm of gradient ascent on the SVM optimization problem in eqn.(4), the following formula is used to update α_i in each iteration:

$$\alpha_i^{new} = \alpha_i - step * y_i * \left(\alpha_i . y_i, K\left(x_i, x_j\right) + b\right)$$

Constrained to

$$0 \leq \alpha_i^{new} \leq C$$

Where b is the bias of the training set points.

After calculating all $\alpha's$, the hyper plane is calculated as follows:

$$w = \sum_{i=1}^{n} \alpha_i . x_i . y_i$$

To get the new bias b_{new}, substitute in the following formula by x_i, y_i of any of the support vector points (those with $0 < \alpha_i < C$)

$$b_{new} = y_i - w . K(x_i, x_j)$$

Table I shows the detailed GA algorithm using a pseudo code. First, all Lagrange multipliers $\alpha's$ and bias b are initialized to zero. In each iteration, two loops are performed: the outer loop in which the input vector x_i is read from the memory, and the inner loop in which the Kernel function value is calculated between x_j and all other input vectors. Then, α_i is updated with the new value and passed to the outer loop with the next α till all Lagrange multipliers are updated. Then, the bias is updated and a convergence check is applied. One important note on the training and testing data sets is that they should be normalized to make all data point components mapped to the range (-1; 1). This is conducted easily by subtracting the mean value of the components from each component, then dividing the resultant value by their standard deviation.

Table 2- Psuedo code of Gradient Ascent algorithm.

Initial

$$w = 0, \alpha = 0, b = 0$$

Iterate till convergence

  Loop1

  Read x_i from memory

    Loop2

    Read x_j, α_j, y_j from memory

    Calculate $K(x_i, x_j)$

    Multiply $K(x_i, x_j).\alpha_j.y_j$

    End loop2

  Update α_i^{new}

  Check α_i^{new} satisfies constraint

  End loop1

  Update bias

Check for convergence

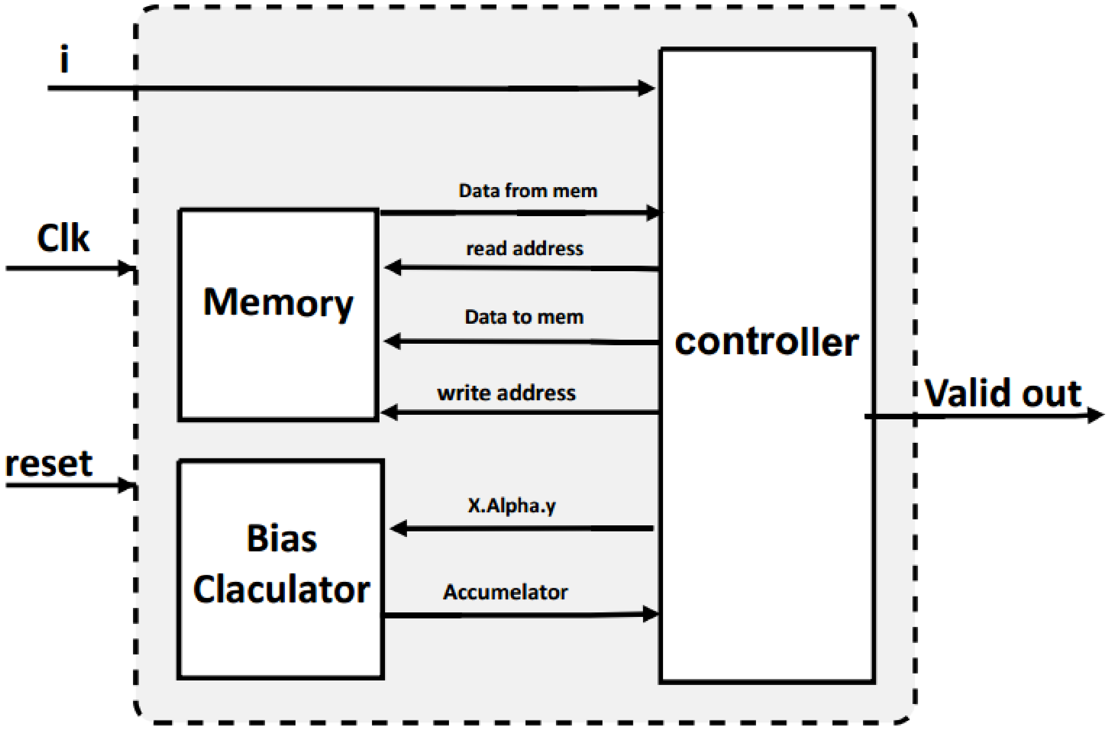


Figure 15 - Gradient Ascent training circuit block diagram.

4.2.2. Hardware Implementation

Figure 15 shows the architecture of the top level design of the Gradient Ascent (GA) algorithm which consists of three main blocks: memory, controller and bias calculator.

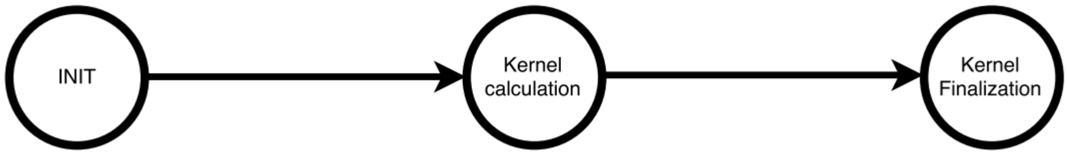


Figure 16 - GA controller finite state machine.

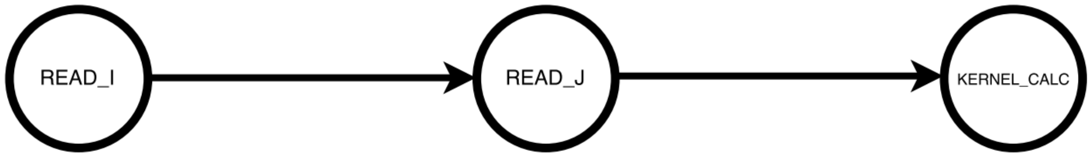


Figure 17 - GA kernel calculation phases finite state machine.

The memory contains the values of y_i , α_i , x_i , b, α_i^{new} and has separate input data, output data, read address and write address ports. All these ports are drived by the controller module. The memory is designed carefully and the data is arranged in it to achieve minimum memory access times.

The controller is the main block in the architecture. It contains the main finite state machine (FSM) that controls the flow of the data and the memory interface. Figure 16 shows the controller FSM noting that some states in the FSM contain other embedded FSMs as will be explained later.

In INIT state, all variables are initialized and the memory read address is set. In the Kernel calculation state, the value of the kernel function is calculated through many phases as depicted in Figure 17.

In READ_I phase, the input vector x_i is read from the memory. In READ_J, the input vector x_j is read form the memory. Then, the kernel function is calculated in KERNEL_CALC phase. After the kernel calculation is conducted, the main controller FSM is moved to the Kernel finalization state.

In the Kernel finalization state, the expression $x_j.\alpha_j$ is calculated and is multiplied by the kernel function value and then the output is sent to be accumulated at the bias calculator. The FSM of different phases of the Kernel finalization state is portrayed in Figure 18. IN ADDRESS_YJ phase, the FSM generates the address of y_j . IN WAIT_FOR_MEM phase, the FSM generates the address of α_j . Then the controller reads the values of y_j and α_j in READ_YJ and READ_ALPHA respectively. IN CALC_ALPHA_Y phase, the value α_j . y_j is calculated using an XOR gate. The value

α_j . y_j .(x_i . x_j) is calculated in CALC_OUT phase using a multiplier. This vale is passed to the top level module to be accumulated for different i's. In this phase, the address of b is generated and sent to memory.

In WAIT_FOR_MEM2, READ_B,READ_YI and SEND_X phases, the FSM reads the values of α_i, y_i, b and passes them to the top level to be used in bias calculation as the controller is the only unit that interfaces with the memory.

Different approximate computing techniques are used in implementing the proposed GA training accelerators to reduce power consumption. First of all, fixed point is used instead of the computationally expensive floating point. Using software simulation results, a 16-bit word length is enough for achieving the same performance (i.e., accuracy). Reducing the word length less than 16 bits achieves more power saving with the cost of performance degradation. At a certain word length, the full dynamic range of the bits should be used in order to achieve the highest accuracy for this configuration. This requires a smart selection of the integer and fraction portions of the fixed point word length.

Second, Computation skipping is used in different steps in the two algorithms (i.e., multiplying by zero is skipped). As $\alpha = 0$ for all non support vector points, many multiplication operations are skipped.

Finally, inaccurate arithmetic techniques are adopted in the hardware accelerator implementations. Using inaccurate arithmetic operations introduces some errors which are acceptable in a specific range. However, using this inaccurate arithmetic operations saves a big chunk of energy. As multiplier are one of the most power hungry blocks, the signed truncated multiplier proposed in [52] is utilized. The signed truncated multiplier consumes less power than accurate multipliers by summing an optimized partial products matrix (PPM). A truncated accumulation is used then accumulating the whole output of the multiplier (i.e., the output of the multiplier is truncated to the specified word length, than the accumulation operation is performed). This also reduces the size/power of the needed accumulator and has a small impact on accuracy. Sign and magnitude representation is used for negative numbers to facilitate the multiplication by -1 which appears in the algorithm several times, therefore an XOR implementation is utilized. Moreover, the step size is chosen to be multiples of 2 to use an add-shift multiplier to reduce the power consumption.

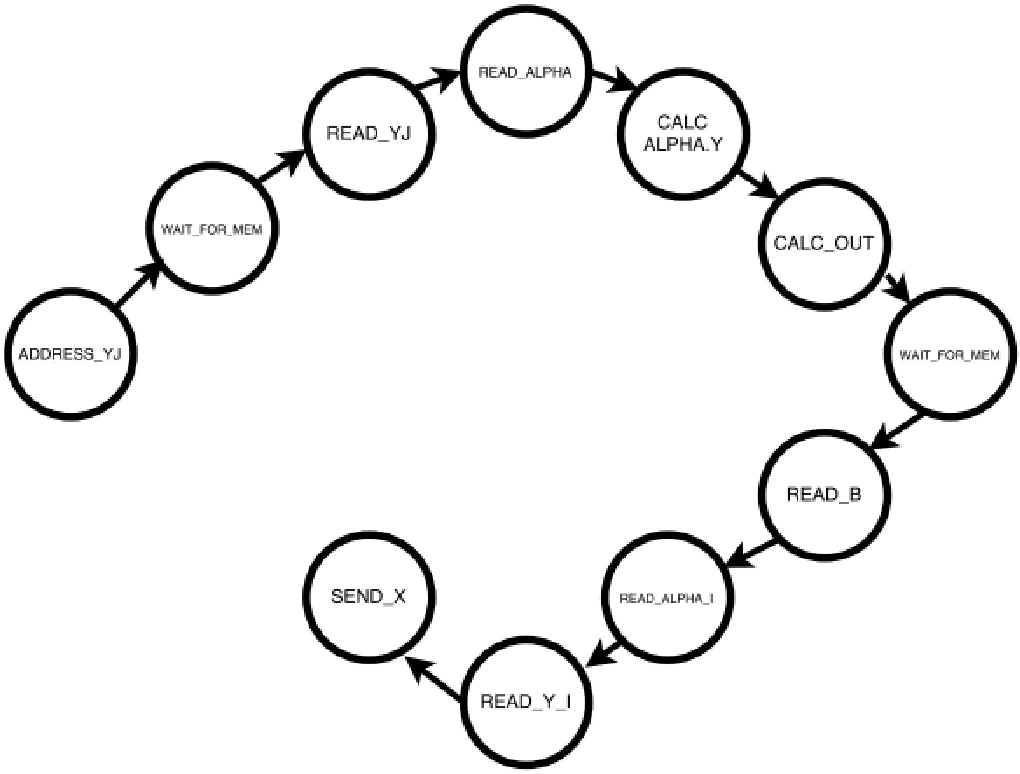


Figure 18 - GA kernel finalization phases finite state machine.

4.3. Sequential Minimal Optimization (SMO)

4.3.1. Algorithm

The SMO algorithm was introduced and comprehensively explained by John Platt [51]. The main idea of the SMO technique is to break any large QP problem into multiple smaller ones. It solves the constrained quadratic programming problem efficiently as it iteratively narrows the optimization problem to just two Lagrange multipliers in each iteration. The selection of the two Lagrange multipliers to optimize the function value in each iteration is performed heuristically. However, depending on the application, the SMO algorithm scales somewhere between linear and quadratic with the number of the data training set.

The SMO algorithm optimizes the objective function by jointly optimizing two Lagrange multipliers. The fact that optimizing two Lagrange multipliers is performed analytically makes the SMO algorithm advantageous. The SMO algorithm is summarized in Table 3.

The SMO algorithm starts by selecting two Lagrange multipliers to optimize the objective function and calculates the bounding values of the two Lagrange multipliers.

The bounding values of only two Lagrange multipliers are depicted in a 2-D square as in Figure 19. On the left, the bounding square when $y_1 \neq y_2$. Hence, $\alpha_1 - \alpha_2 = constant$. On the right, the bounding square when $y_1 = y_2$. Hence, $\alpha_1 + \alpha_2 = constant$. The square sides represent the maximum and the minimum values of the Lagrange multipliers while the diagonal line represents the values the two Lagrange multipliers are allowed to take.

Table 3 - PSUEDO code of Sequential Minimal Optimization algorithm.

$Initial$	
	$w = 0, \alpha = 0, b = 0$
$Iterate\ till\ convergence$	
	$Select\ i\ and\ j$
	$read\ \ x_i and\ x_j from\ memory$
	$cakculate\ kernel\ functions\ K_{ii}, K_{ij}, K_{jj}$
	$calculate\ the\ errors\ E_i, E_j$
	$calculate\ the\ limits\ L\ and\ H$
	$\eta = 2K_{ij} - K_{ii} - K_{ij}$
	$\alpha_j^{new} = \alpha_j^{old} + \frac{y_j(E_j^{old} - E_j^{new})}{\eta}$
	$calculate\ the\ bias\ b$
$check\ for\ convergance$	

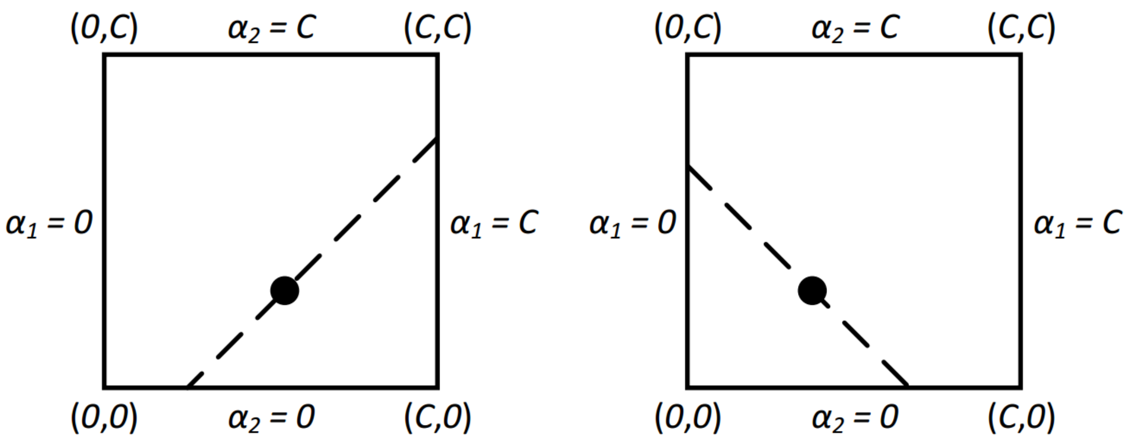


Figure 19 - The bounding values of two Lagrange multipliers.

Denoting the two Lagrange multipliers by: $\alpha_1\ and\ \alpha_2$, it is required to get the new values for the two Lagrange multipliers $\alpha_1^{new}, \alpha_2^{new}$ from the old set of all Lagrange multipliers $\{\alpha_1^{old}, \alpha_2^{old}, \alpha_3, \alpha_4, \dots, \alpha_N\}$, where $\alpha_1^{old}, \alpha_2^{old}$ have the initial value zero.

Given the constraint equation $\sum_{i=1}^{N} \alpha_i . y_i = 0$, the following condition is derived:

$$y_1\alpha_1^{new} + y_2\alpha_2^{new} = y_1\alpha_1^{old} + y_2\alpha_2^{old}$$

Following the derivations in [51], α_j^{new} is obtained by the following equation:

$$\alpha_j^{new} = \alpha_j^{old} + \frac{y_j(E_j^{old} - E_j^{new})}{\eta}$$

Where $K_{ii} = x_i^T . x_i$,

$K_{jj} = x_j^T . x_j$,

$K_{ij} = x_i^T . x_j$,

$\eta = 2K_{ij} - K_{ii} - K_{jj}$,

$E_i = w^T x_i - b - y_i$.

Referring to the constraints depicted in Figure 19, α_j^{new} is clipped to be in the feasible range. Therefore, $\alpha_j^{new,clipped}$ is obtained by:

$$\alpha_j^{new,clipped} = \left\{ \begin{array}{cc} H, & \alpha \geq H \\ \alpha_j^{new}, & L \leq \alpha_j^{new} \leq H \\ L, & \alpha \leq L \end{array} \right\}$$

And therefore, α_i^{new} is calculated as follows:

$$\alpha_i^{new} = \alpha_i^{old} + t\left(\alpha_j^{old} - \alpha_j^{old,clipped}\right)$$

Where $t = y_i . y_j$

4.3.2. Hardware implementation

In order to keep the architecture generalized for any heuristic model of selecting Lagrange multiplier, the SMO training architecture is divided into three main blocks; the SMO processing unit, the SMO controller and the main memory as shown in

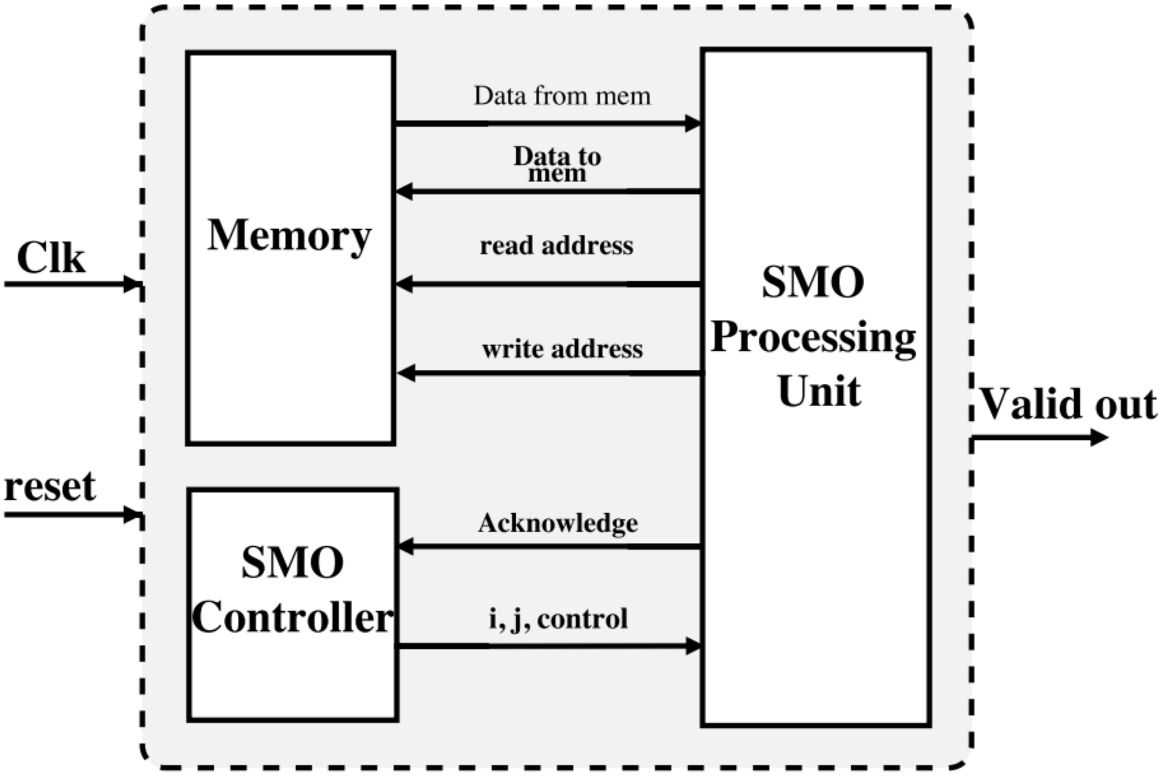


Figure 20 - Sequential Minimal Optimization training circuit block diagram.

4.3.2.1. The SMO Processing Unit

The SMO processing unit is responsible for calculating the new values of the two previously selected Lagrange multiplier. Figure 21 shows the building blocks of the SMO processing unit.

1- Register file

In order to speed up the processing and avoid the repeated memory access, some variables are cached in a register file to be processed later by the other SMO processing unit blocks. The variables chosen to be cached in the register file are $\alpha_i, \alpha_j, y_i, y_j, B, \alpha_i^{new}, \alpha_j^{new}, E_i, E_j$.

2- Kernel function

The calculation of η requires the calculation of the two Lagrange multiplier self and cross kernel. Hence, the kernel function unit calculates the value of the k_{ii} , k_{jj} , k_{ij} simultaneously. After receiving the index of current Lagrange multipliers, the kernel function unit reads from the memory the value of the two Lagrange multipliers and pass them to three multiply-add units as shown in Figure 22. In the case of polynomial kernel instead of the linear one, the design also have an adder to add 1 to each K then use a multiplier to raise the value to the polynomial degree in multiple clocks. The kernel function unit includes an internal controller to manage the iterative process of reading the Lagrange multiplier and updating the kernels value.

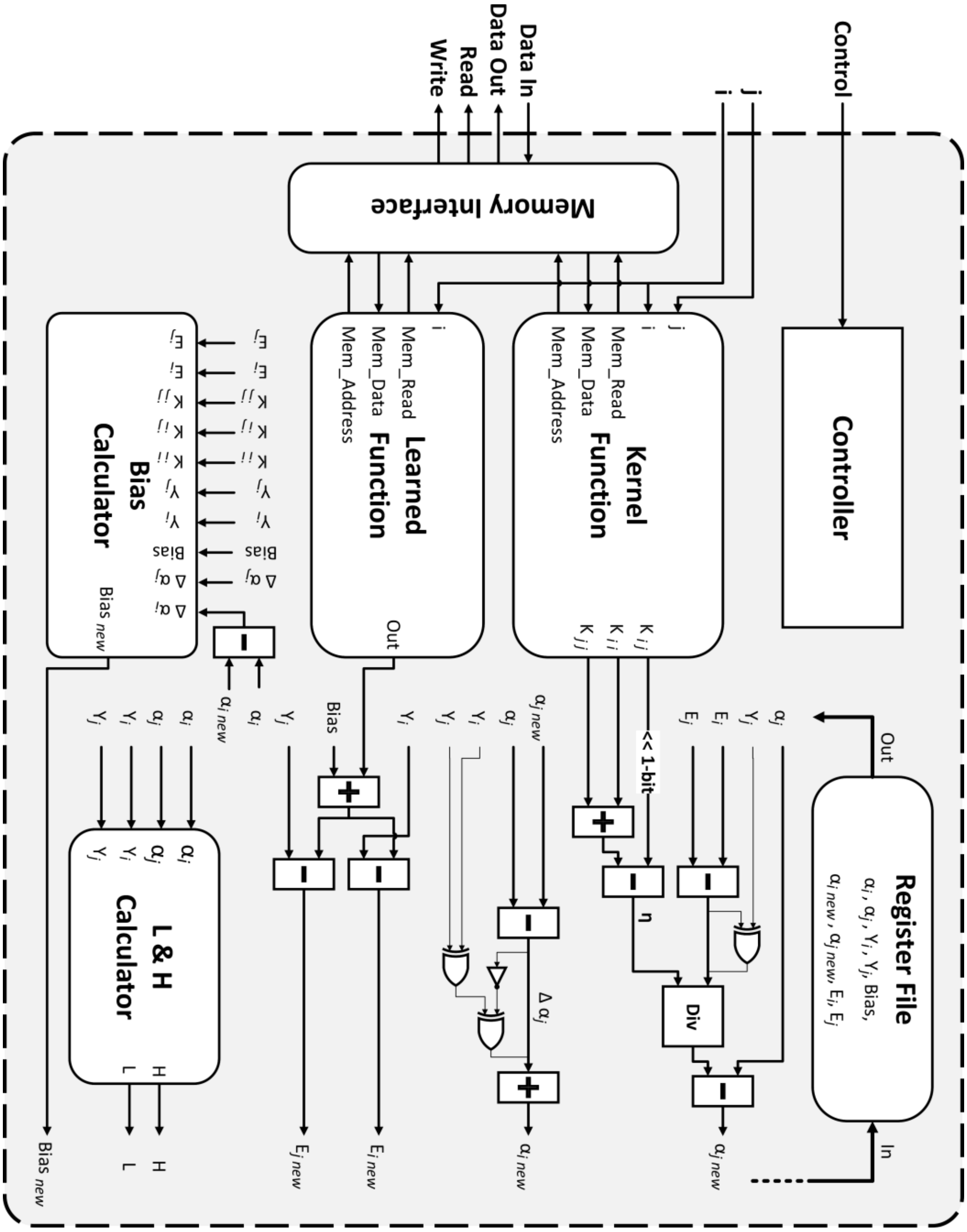


Figure 21 - SMO processing unit block diagram.

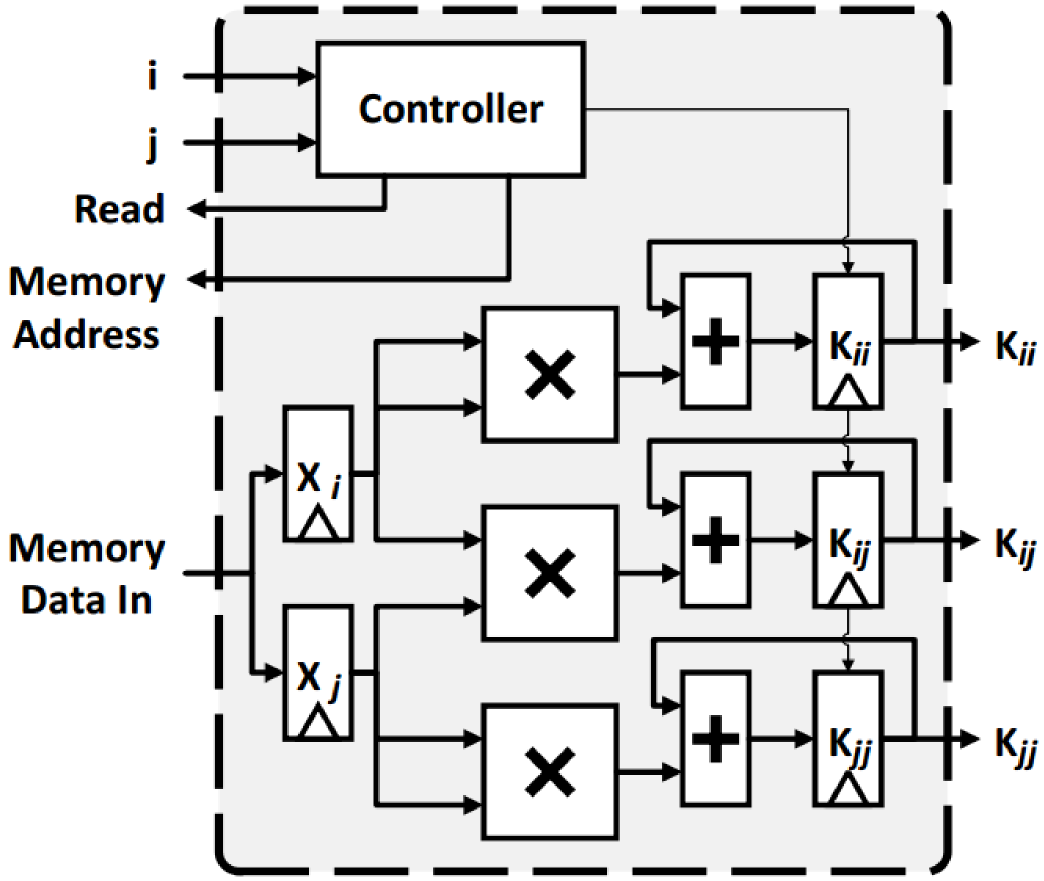


Figure 22 - kernel function block diagram.

3- Learned function

Learned function is used to calculate $w^T x$ or $\sum_{i=1}^{n} \alpha_i . y_i . K(x_i, x_j)$ which is used in calculating the error E. By expanding the equation $\sum_{i=1}^{n} \alpha_i . y_i . K(x_i, x_j)$, the pseudo in Table 2 is obtained.

Table 4 - Learned function PSUEDO code.

$for\ j = 1: N$ $\quad for\ d = 1: dimensions$ $\qquad k = k + x_{i,d} x_{j,d}$ $\quad sum = sum + \alpha_i y_i k$ $func = sum$

The implementation requires two multiply-add units; one to calculate the kernel and the other to update the learned function. However, since the two calculation is dependent, one multiply-add unit is shared to calculate both values.

The FSM of the learned function is shown clearly in Figure 23. In the first state, α_i is read. If $\alpha_i \neq 0$, the FSM is moved to the kernel calculation state. Then, y_j is read to update the learned function value.

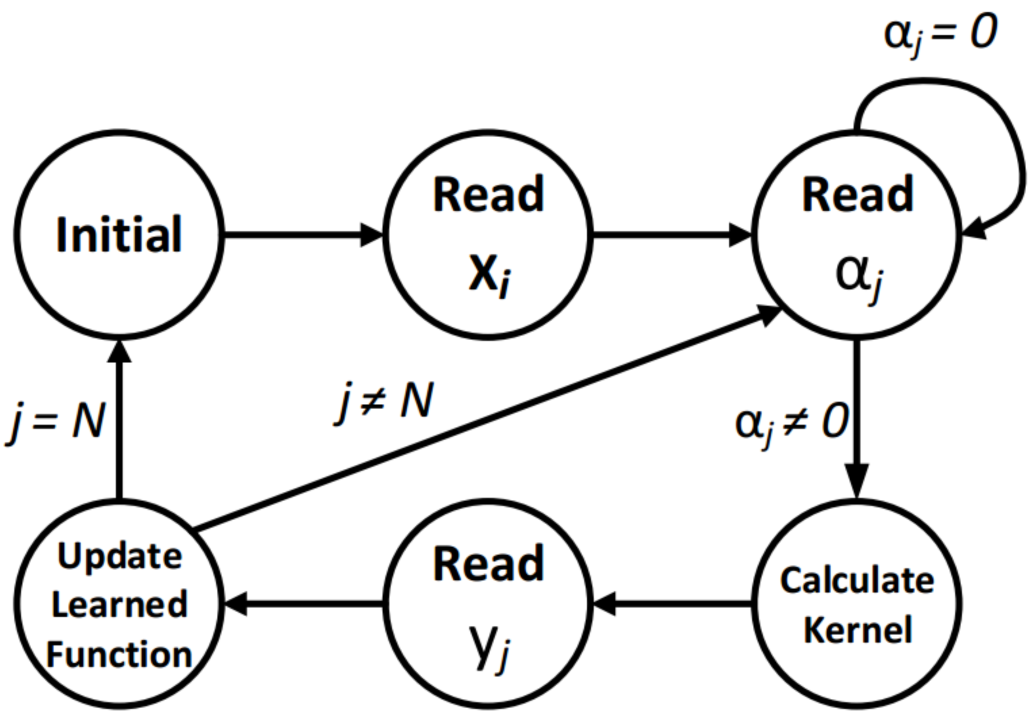


Figure 23 - Learned function FSM.

4- Bias calculator

The change in the threshold is computed by forcing E_i^{new} to be zero if $0 < \alpha_i^{new} < C$ and then

$$b_1 = E_i + y_i.\Delta\alpha_i\,.k_{ii} + y_j.\Delta\alpha_j.k_{ij} + b$$

Where $\Delta\alpha_i = \alpha_i^{new} - \alpha_i$,
$\Delta\alpha_j = \alpha_j^{new} - \alpha_j$

Otherwise, the threshold is computed by forcing E_j^{new} to be zero if $0 < \alpha_i^{new} < C$ and then

$$b_2 = E_j + y_i.\Delta\alpha_i\,.k_{ij} + y_j.\Delta\alpha_j.k_{jj} + b$$

Finally, the new bias is calculated as follows:

$$b = \begin{cases} b_1, & 0 < \alpha_i^{new} < C \\ b_2, & 0 < \alpha_j^{new} < C \\ \dfrac{b_! + b_2}{2}, & otherwise \end{cases}$$

Figure 24 shows clearly the FSM of bias calculator which consists of different states: calculate b1, calculate b2 then choose one of them or their average.

Figure 25 illustrates the implementation of the bias calculator unit. The unit is implemented using only two multipliers, four adders, and three intermediate registers A, B, and b1.

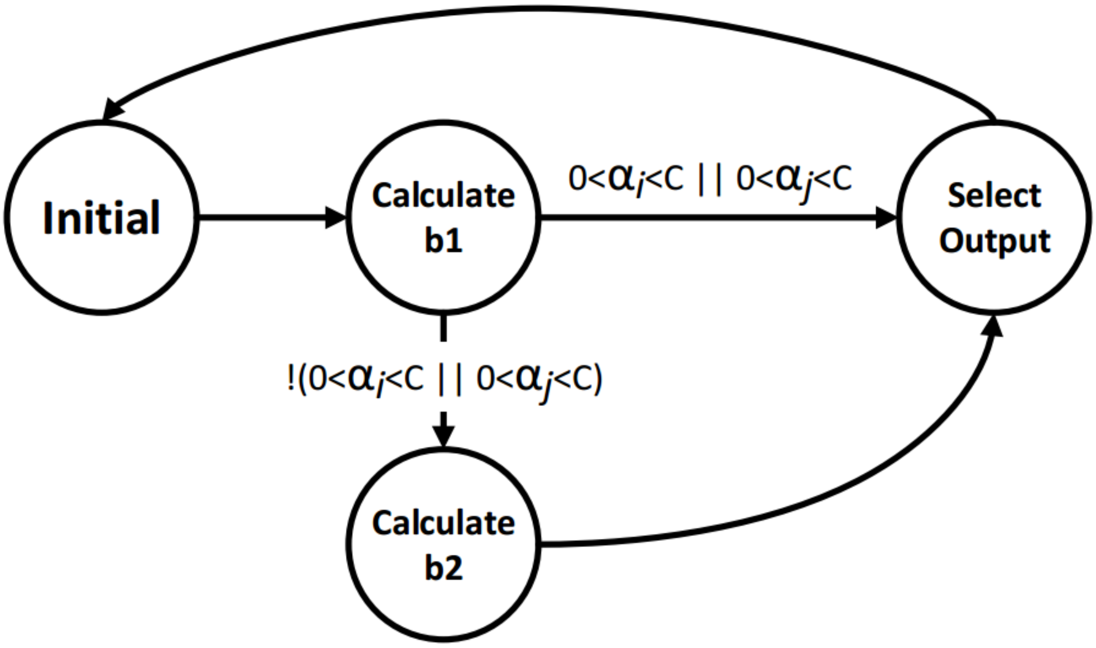


Figure 24 - Bias calculator FSM.

To exploit the similarities between equations of calculating $b_1 and\ b_2$, they can be rewritten as:

$$b_1 = E_1 + T_1 + T_2 + b$$
$$b_2 = E_2 + T_3 + T_4 + b$$

Where $T_1 = y_i.\Delta\alpha_i.K_{ii}$,
$T_2 = y_j.\Delta\alpha_j.K_{ij}$,
$T_3 = y_i.\Delta\alpha_i.K_{ij}$,
and $T_4 = y_j.\Delta\alpha_j.K_{jj}$.

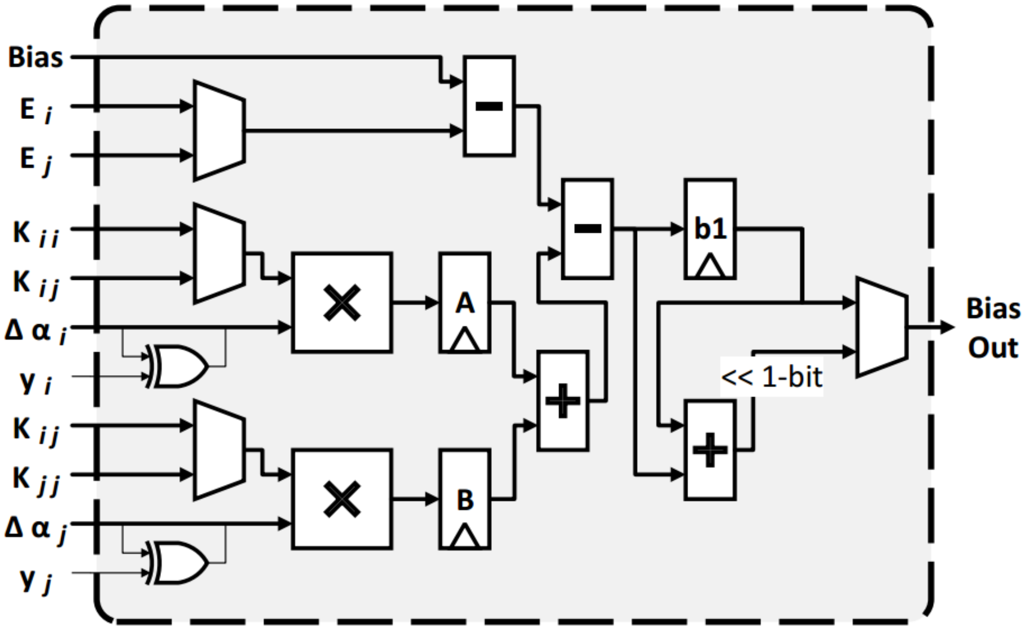


Figure 25 - Bias calculator hardware implementation block diagram.

Noticing the similarity between T1 and T3, only one multiplier is used to calculate $\Delta\alpha_i.k_{ii}$ and $\Delta\alpha_i.k_{ij}$, and therefore the values of T_1 and T_3. Based on the condition $0 < \alpha_i^{new} < C$ and the condition $0 < \alpha_j^{new} < C$, either k_{ii} or k_{ij} is selected to be an input to the multiplier. If both conditions are satisfied, both b_1 and b_2 gives the same value. In the proposed hardware implementation, the priority is given to b_1 to reduce the hardware complexity. Therefore, the value of register A is calculated. The fact that y has a unity value, with positive or negative sign, and adopting the sign and magnitude representation, results in reducing the multiplication of y to a single XOR gate between y sign and the multiplicand sign. Similarly, T_2 and T_4 calculations require only one multiplier and then the value of B register is obtained in parallel with the calculation of the register A. If both conditions are not satisfied, the calculation is carried out to determine the value of b_1 , then the process is repeated to determine the value of b_2 and finally the value of $b_!$ and b_2 are averaged.

5- Limits calculator

The value of the lower band L and the upper band H depends on the slope in Figure 19. Therefore the value of the limits is obtained as follows:

$$if\ y_i \neq y_j \rightarrow L = \max(0, \alpha_j - \alpha_i), H = \min(C, C + \alpha_j - \alpha_i)$$

$$if\ y_i = y_j \rightarrow L = \max(0, \alpha_j + \alpha_i - C), H = \min(C, \alpha_j + \alpha_i)$$

Again, comparing y_i and y_j is done using a single XOR gate. From the previous equations of L and H, L and H take on the values 0, C , $\alpha_j \pm \alpha_i$, or $\alpha_j \pm \alpha_i \pm C$. Therefore, only two adders are required to calculate L and H, while the signs are determined using XOR gates. To further understand the implementation, the limits calculation process is described using the pseudo code in Table 5.

In the first part, the first adder is adjusted to add $\alpha_j - \alpha_i$ and the second adder is adjusted to add C to the output of the first adder, (i.e., $+ \alpha_j - \alpha_i$) . Then a multiplexer is used to select between the values 0 and $\alpha_j - \alpha_i$ for L, and the values C and $C + \alpha_j - \alpha_i$ for H.

In the second part, the first adder is adjusted to add $\alpha_j + \alpha_i$ and the second adder is adjusted to add $-C$ to the output of the first adder, (i.e., $\alpha_j + \alpha_i - C$) . Then a multiplexer is used to select between the values 0 and $\alpha_j + \alpha_i - C$ for L, and the values C and $\alpha_j + \alpha_i$ for H. The sign adjustment of α_i and C is controlled by examining if $y_i \neq y_j$. This examine is performed using an XOR gate. Accordingly, the sign of α_i and C is altered by another two XOR gates. Noting that the cases when α_i is required to be negative is the same cases when C is required to be positive. That is why a NOT gate is added to the sign of C as shown in Figure 26.

Table 5 - Limits calculator PSUEDO code.

$if\ y_i \neq y_j\ then$
    $if\ \alpha_j - \alpha_i is\ positive\ then$
        $L = \alpha_j - \alpha_i$
        $H = C$
    $else\ then$
        $L = 0$
        $H = \alpha_j - \alpha_i + C$
    $end\ if$
$else\ then$
    $if\ \alpha_j + \alpha_i - C\ is\ positive\ then$
        $L = \alpha_j + \alpha_i - C$
        $H = C$
    $else\ then$
        $L = 0$
        $H = \alpha_i + \alpha_j$
    $end\ if$
$end\ if$

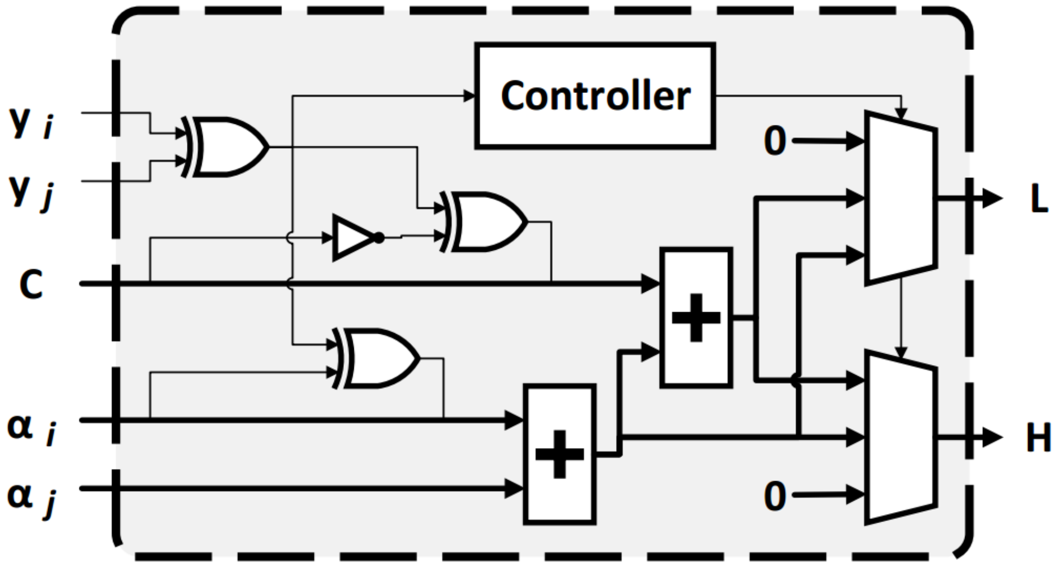


Figure 26 - Limits calculator block diagram.

6- Memory interface

The memory interface is responsible for receiving the requests for the memory read and write operations and handling the memory access separately by different blocks, which increases the memory access parallelism.

7- Controller

This unit controls the other units by initiating a triggering signal for each unit and manages the data flow between them. Figure 27 summarizes the control state machine of the control unit.

4.3.2.2. The SMO controller

The SMO controller is responsible for selecting the two Lagrange multipliers and controls the SMO processing unit. The SMO controller keeps iterating over Lagrange multipliers till conversion happens or the maximum number of iterations is exceeded. Compared to the SMO processing unit, the SMO controller hardware is simpler and consumes less area.

The same approximate computing techniques used in the hardware implementation of the GA accelerator are also adopted in the hardware implementation of the SMO accelerator. Fixed point arithmetic, computation skipping, inaccurate arithmetic and sign/ magnitude implementation is used in the proposed implementation.

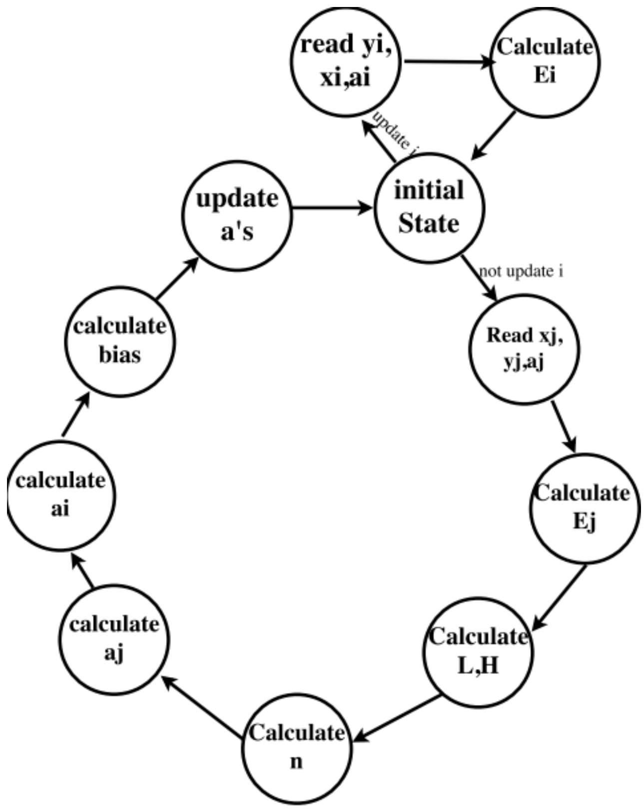


Figure 27 - SMO processing unit FSM.

4.4. Simulation Setup

The SVM training accelerators techniques implemented in this paper are tested first on MATLAB2016a. EEG signals of patients are first processed, then the features that give the best performance are extracted. Then, the training and testing data are used to verify the performance of the training algorithms. The proposed training techniques are software implemented on MATLAB to measure the performance. Xilinx ISE 14.2 is utilized to design and develop the VLSI architecture of the algorithms. The design is synthesized on Xilinx Spartan-6 FPGA. For the implementation on ASIC, Synopsys DesignCompiler (DC) B-2008.09 with UMC 130nm library is adopted.

Results are collected in two main phases. The first phase is evaluating the performance simulation results. The second phase is calculating the hardware implementation metrics such as area, power and maximum frequency for both ASIC and FPGA implementations.

4.5. Simulation Results

After implementing both SVM training algorithms –GA and SMO- on MATLAB 2016a, both algorithms are tested with linear kernel and their results are shown in Table 6. The performance of both algorithms are almost the same. They both achieve sensitivity equals to 96%.

The performance obtained by the proposed architectures is also compared to the performance achieved by prior work as shown in Table 7. It is obvious that the sensitivity obtained by the proposed architectures is equal to and exceeds that achieved by the prior work. This results obtained despite using linear kernel while most of the prior work used Radial Basis Function (RBF) kernel. This saves much energy as the linear function kernel is less complex than the RBF kernel and needs less computations.

Table 6 - Performance measurement for seizure detection using different SVM training techniques.

Algorithm	Sensitivity	Specificity	Accuracy
GA	95.8	92.34	92.35
SMO	96.0	97.9	97.9

Table 7 - Performance comparison to prior work.

Method	Kernel Type	Sensitivity
[25]	RBF	95%
[26]	RBF	97%
[53]	RBF	94.5%
Proposed	Linear	96.7%

4.6. Hardware Implementation Results

The hardware implementations of SVM learning circuit are presented on both FPGA and ASIC platforms. Table 8 shows the ASIC implementation results using UMC 130nm where both techniques use a clock frequency equals to 100 MHz. Table 8 shows area, power and the number of clock cycles that each algorithm takes to finish training. As power consumption is not a good comparison metric, power delay product is calculated as the product of power consumption of each technique and the number of clock cycles needed to finalize training.

Table 9 lists the resources used in Xilinx Spartan-6 FPGA such as LUTs and registers slices. Table 9 also tabulates the dynamic power consumption of each algorithm and the power delay product (PDP). PDP is calculated as the multiplication of dynamic power with the number of clock cycles needed to finish training.

Table 8 - Hardware implementation results of SVM training algorithms on UMC 130nm platform.

Algorithm	Area (nm^2)	Power (μW)	# training cycle	PDP
GA	18143	463	150K	69.45
SMO	43259	910	30K	27.3

Table 9 - Hardware implementation results of SVM training algorithms on Spartan-6 FPGA platform.

Algorithm	Utilization		Power (mW)	PDP
	LUTs	Registers		
GA	661	535	6	900
SMO	3360	566	17.2	516

Table 8 shows the comparison between the implementation of both algorithms on ASIC platform in area and power consumption. It is obvious that the GA implementation consumes less area and instantaneous power than that consumed by the SMO implementation. However, the large number of clock cycles needed for the GA algorithm to finalize training makes the energy consumed by the GA algorithm is more than that consumed by the SMO algorithm. It is so clear that the time required by the GA algorithm to finalize training is 5x the time required by the SMO algorithm.

In Table 9, it is obvious that the GA algorithm has the advantage of less utilization, higher maximum frequency and less power consumption than the SMO algorithm. However, the main disadvantage of the GA algorithm is the large required number of clock cycles for training, which reaches up to 150,000 compared to 30,000 clock cycles only for the SMO algorithm. The utilization used by the SMO accelerator is less than that achieved by [34].

Chapter 5 : Design of Classifiers

As mentioned in the introduction and literature review, many machine learning techniques are used to detect seizure. Two different techniques are proposed and hardware implemented for classification and their performance for neural seizure detection is measured. The two techniques are Support Vector Machine (SVM) and Artificial Neural Networks (ANN). Both algorithms are discussed in details in the following to sections.

5.1. Support Vector Machine (SVM) Classifier

5.1.1. Algorithm

After the completion of training phase, the classification phase starts. For any input vector x_{test} , by substituting in the following formula using the final value of α's and b , the corresponding class y_{test} is calculated as follows:

$$y_{test} = \sum_{j=1}^{n} \alpha_j y_j x_{test} x_j + b$$

5.1.2. Hardware implementation

The training of SVM is done offline or using the hardware accelerator proposed in Chapter 4. Hence, only the SVM classifier needs to be hardware implemented. Figure 28 shows the architecture of the top level design of the SVM classifier which consists of 6 main block: three ROM blocks, classifier block and inner product block.

The first ROM block is used to save the input vectors of the support vector points. The width of this ROM is the same as the data width, while the depth equals to the number of support vectors multiplied by the number of the classification problem dimensions.

The second ROM block is used to save the values of non-zero 's. The width of this ROM is the same as the data width, while the depth equals to the number of support vectors.

The third ROM block is used to save the values of the true labels of the support vector points. The width of this ROM is one bit, while the depth is the number of support vectors.

The finite state machine (FSM) is responsible for generating the addresses of the three ROMs and the enable signal of classifier block.

The classifier block is the main block of the architecture. First, each α is multiplied by its corresponding label y. As the implementation used for negative numbers is sign-magnitude implementation, the multiplication is performed using an XOR gate instead of a multiplier. The value of $\alpha_i . y_i$ is saved in a register. An inner product block of size equal to the number of dimensions is used to multiply the input test vector with the input vector of the i^{th} support vector point. The output of the classifier block is fed to the inner product block to calculate the class.

The inner product block is a multiple-add block with only one adder and one multiplier that multiply two vectors of size equal to the number of non-zero α's. The output of this block is the class and a valid out signal.

In the hardware implementation of SVM classifier, fixed point simulation is used. Using software simulation results, it is found that a 16-bit word length is enough for achieving the same performance (i.e., accuracy). Same as that used in the training accelerators, computation skipping is adopted to save more power/ area.

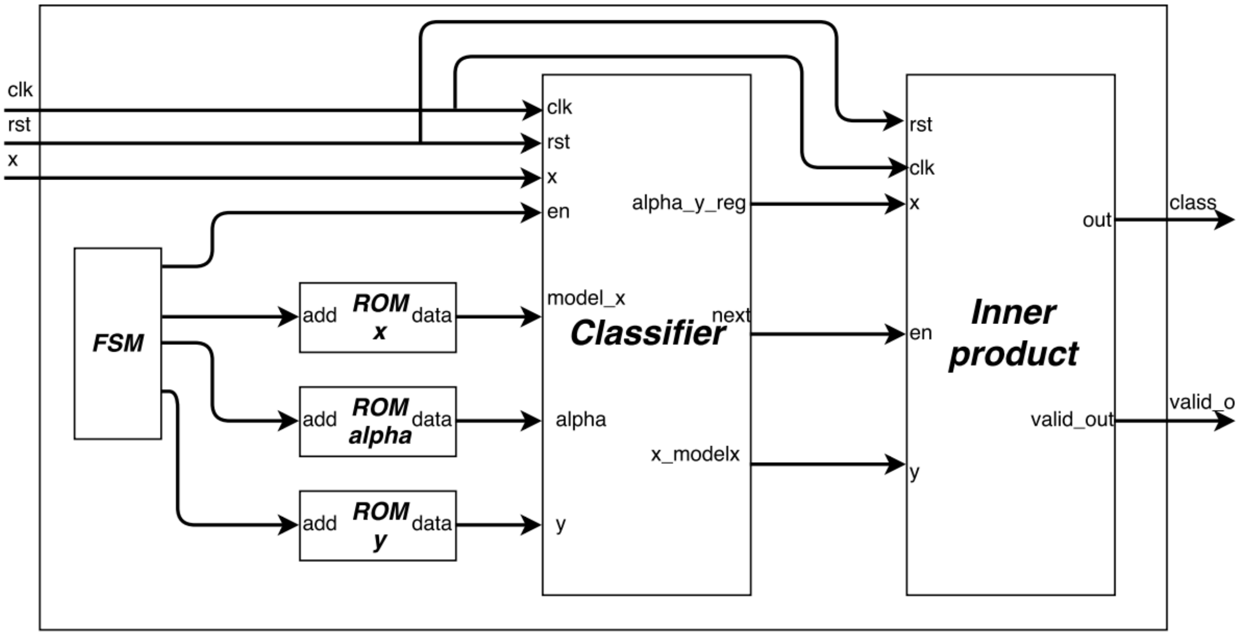


Figure 28 - Top level SVM classifier block diagram.

5.2. Artificial Neural Network (ANN)

5.2.1. Algorithm

Over the past twenty years, many methods inspired by the understanding of the structure and function of the biological neural networks are evolved. One of these methods is the artificial neural network (ANN) [54]. Neural networks are used in various applications such as classification, pattern recognition, and data analysis [55]. ANN mainly consists of an input layer, one or more hidden layers and one output layer as shown in Figure 29. Each layer consists of multiple neurons and different weights are given to the connections among these neurons. Each neuron in the input layer takes in one data source. The output of each input layer neuron is the input for each of the hidden layer neurons [56].

Finding the weight of each neuron is performed in the training phase. After the neural network is trained, any new input vector is fed to the input layer. The value of each node is calculated by multiplying the input node value by the connection weight and adding all the values entering this node. To detect seizure and differentiate between seizure and non-seizure epochs, the architecture of the ANN used is a single hidden layer with 10 neurons. The activation function used is the Sigmoid function.

For any new data point, the data point is submitted to the input layer. The value of each node in the first hidden layer thorough add-multiply operation. This procedure is performed with all nodes in all hidden layer until the value of output layer node is calculated.

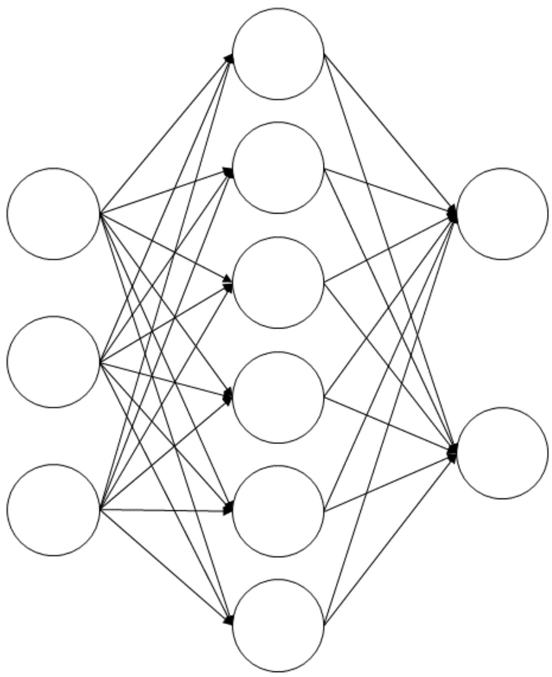


Figure 29 - Three layer feedforward network architecture.

5.3. Hardware implementation

The architecture of the ANN classifier consists of ROM block, two RAM blocks, four counters, neuron block and finite state machine as shown in Figure 30.

A ROM block is used to save the weights of each connection. A single data port RAM is used to save the values of each node (neuron) of the hidden layer. A double data port RAM is used to save the values of each node of the input layer. Four counters are used to generate the addresses of the ROM, single data port RAM and double data port RAM. The neuron block is a multiply-accumulate block that consists of multiplier, adder, register and activation function block. The activation function used is the Sigmoid function and is implemented as a combinational circuit. The FSM is responsible for controlling the overall system.

Different approximate computing techniques are used in implementing the proposed ANN. First of all, fixed point is used instead of the computationally expensive floating point. Using software simulation results, a 16-bit word length is enough for achieving the same performance (i.e., accuracy, in ANNs). Reducing the word length less than 16 bits achieves more power saving with the cost of performance degradation. Another technique for energy saving is the adoption of approximate implementation of the activation functions. For example, instead of implementing the exponential function for calculating the Sigmoid function, a Piece-Wise Linear (PWL) approximation is used to reduce the power consumption.

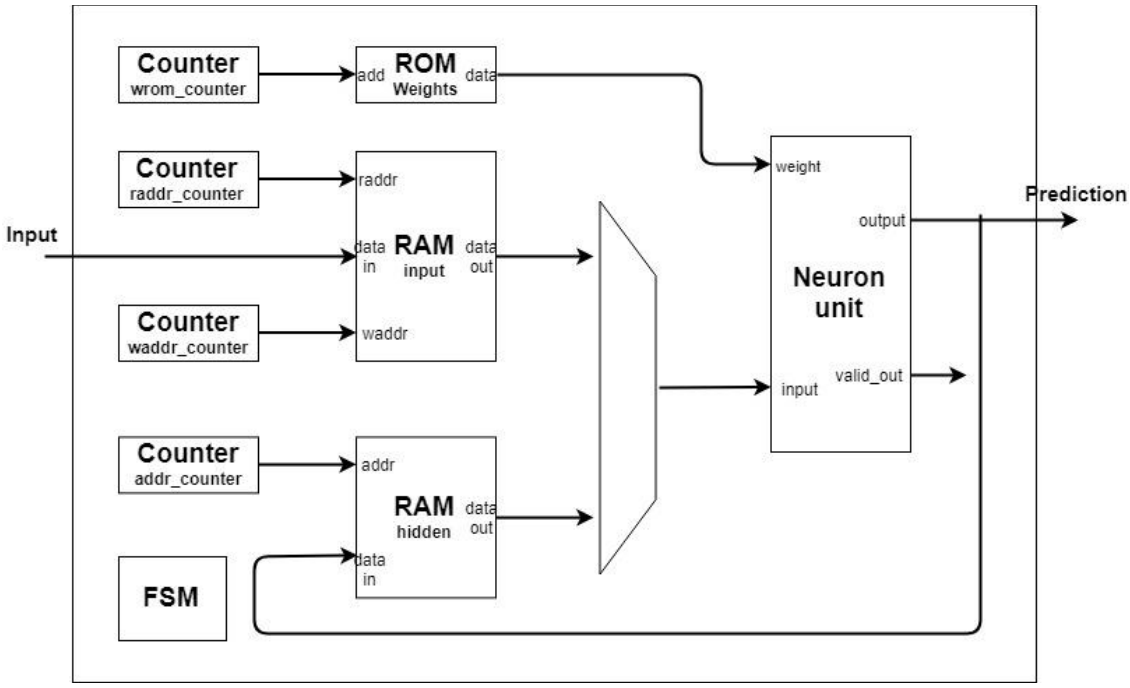


Figure 30 - Top level ANN classifier block diagram.

5.4. Modified ANN

The ANN can achieve a good performance. However, the problem of the ANN is that the decision made in each time epoch is an instantaneous decision. Only the features' values at this time epoch affect the classification output. The task of seizure detection is an accumulative task. The history of the features' values in the previous time epochs can affect the classification. To do so, the single hidden layer used can be a recurrent layer. Recurrent layer has a backward connection. This backward connection means that the output of the nodes in the hidden layer serves as input for the same hidden layer on the next time epoch as shown in Figure 31 and Figure 32. This sort of feedback serves as memory to save the output of the hidden layer in the previous time epochs.

The weight of the backward connection from the hidden layer to the input of the same layer is constant through different time epochs. This weight (W_{hh}) can take one of three different values:

1- $W_{hh} < 1$
2- $W_{hh} \approx 1$
3- $W_{hh} > 1$

In this case, the W_{hh} is chosen less than <1; the memory of the network is limited over time. Hence, the oldest neuron value vanishes over time.

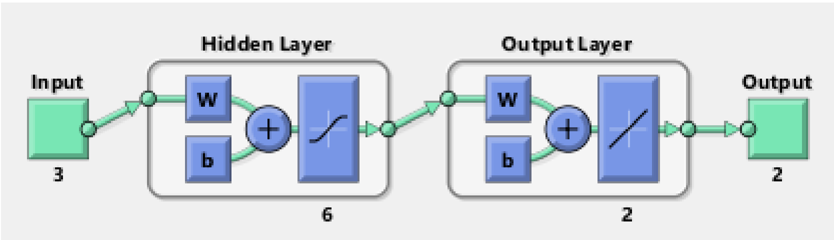


Figure 31 - ANN feadforward architecture.

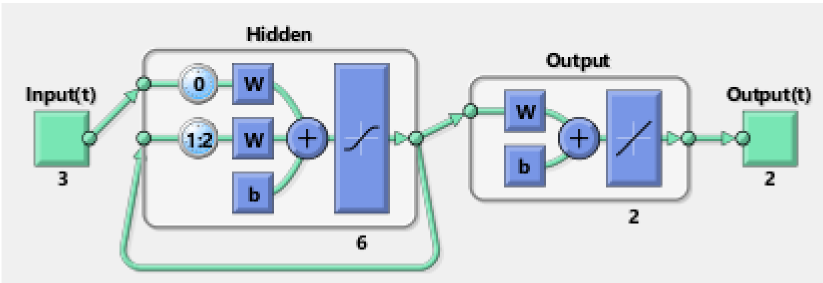


Figure 32 - RNN architecture.

The only difference in hardware implementation is adding a FIFO to the hidden layer to save the output of the hidden layer for the last n outputs to serve as an input in this timestamp.

5.5. Simulation Setup

Both classifiers are software implemented using MATLAB 2016a to measure the performance of each algorithm. The design is synthesized on Xilinx Spartan-6 FPGA. For the implementation on ASIC, Synopsys Design Compiler (DC) B-2008.09 with UMC 130nm library is adopted.

Results are collected in two main phases. The first phase is evaluating the performance simulation results. The second phase is calculating the hardware implementation metrics such as area, power and maximum frequency for both ASIC and FPGA implementations.

5.6. Simulation Results

As shown in Table 10, a comparison between SVM and ANN classifier is performed. The SVM chosen is a linear kernel SVM. The ANN is designed with only one hidden layer with 10 neurons. The two algorithms with the chosen parameters give almost the same performance. This makes the comparison of the power, area and energy as fair as possible.

The appropriate choice of the applied features helps in achieving very high sensitivity using linear kernel in the SVM and using only one hidden layer with only 10 neurons in the hidden layer. This performance exceeds that obtained by Yuan et al. by using SVM with radial basis function (RBF). Yuan et al. got sensitivity ranging from 73.5% to 95% using different features [25].

Table 10 - Performance measurement for seizure detection using different classification techniques.

Algorithm	Sensitivity	Specificity	Accuracy
SVM	96.23	92.90	97.89
ANN	96.5	97.88	97.88

5.7. Hardware Implementation Results

Table 11 shows the hardware implementation results of SVM and ANN classification techniques on ASIC platform. The library UMC 130nm is adopted. In Table 11, it is obvious that the SVM algorithm has the advantage of less utilization, higher maximum frequency and less power consumption than the ANN algorithm. However, the main disadvantage of the SVM algorithm is the large required number of clock cycles to classify every new data point, which reaches up to 1020 clock cycle compared to 30 clock cycle only for the ANN algorithm. This very large number of clock cycle is due to the fact that neural seizure detection problem is a very complex one. Hence, the SVM technique has many support vectors and the inner product occurs for every testing point is very large. However in the case of ANN, only the output of each node is calculated through an add-multiply block. As the throughput of each algorithm is different, power consumption is not a good comparison metric. Hence, power delay product is calculated. Although SVM algorithms consumes less power than the ANN algorithm, the power delay product is much larger.

Table 12 shows the same comparison between the implementation of SVM and ANN classifiers on Spartan-6 FPGA platform. The instantaneous power consumption of the GA algorithm is less than that consumed by the SMO algorithm. However, the energy consumption of the GA is much larger than that consumed by the SMO algorithm due to the large number of clock cycles needed by SVM to finalize classification of each testing point.

Table 11 - Hardware implementation results of different classification techniques on UMC 130nm platform.

Algorithm	Area (nm^2)	Power (μW)	# cycles	PDP
SVM	3963	2.15	1020	2193
ANN	16040	8.08	30	242.4

Table 12 - Hardware implementation results of different classification techniques on Spartan-6 FPGA platform.

Algorithm	Utilization		Power (mW)	PDP
	LUTs	Registers		
SVM	293	137	1	1020
ANN	401	256	3	90

Chapter 6 : Rats Dataset Generation

The PhysioNet data set used in this work has some drawbacks. The first drawback is the limited number of seizures recorded for each patient which makes the training process very difficult. To enhance the overall performance of the seizure detection, more seizure epochs should be recorded for each patient. The second drawback is the unbalanced data. The number of time epochs which have seizure are much less than those which do not have seizure. To solve this problem, a new dataset is measured from rats.

This dataset collected in collaboration with the Faculty of Science, Cairo University and ONE lab. The dataset consists of EEG recordings from rats during ictal and inter-ictal periods. Subjects were injected with drugs that cause temporary seizures. Subjects were monitored for one hour: before, during and after the ictal seizure.

Recordings are measured from 13 different rats. Weights of the rats varies from 90 to 150 gm. Each animal data is exported to an excel sheet that contains the value of the EEG signal in each time sample.

A surgery was performed for each rat to implant 3 electrodes on the cortex lobe. The surgery performed is shown in Figure 33.

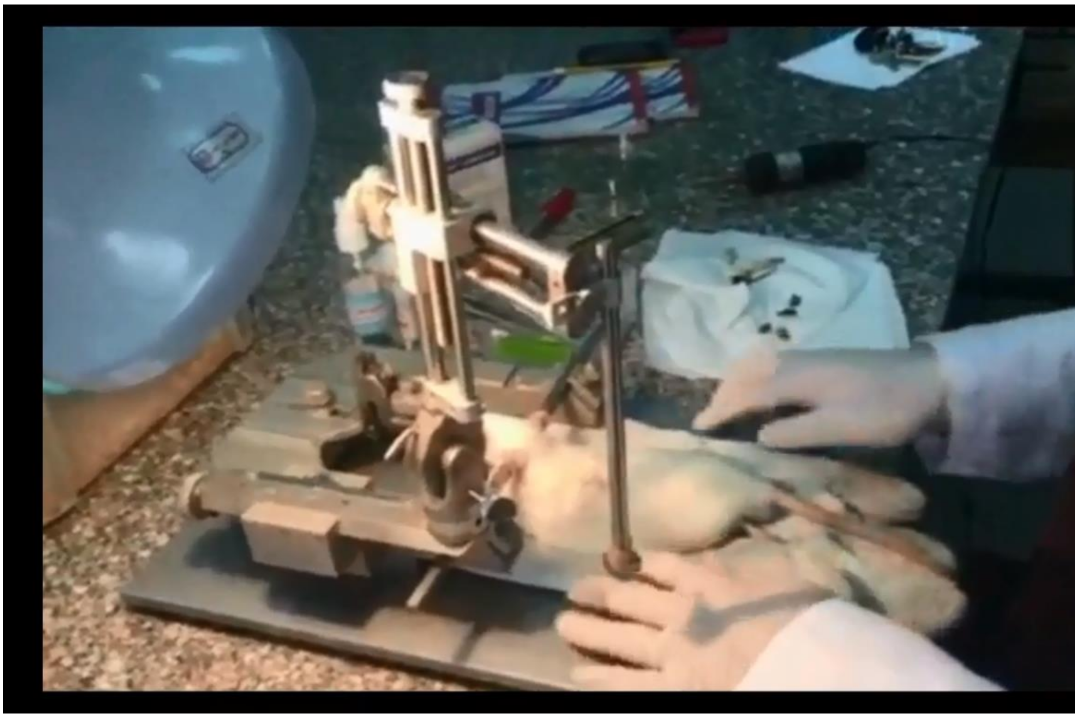
After implanting the electrodes in the rats' cortex as shown in Figure 33-j, the measurement equipment is set up. A commercial EEG instrumentation amplifier is used. The amplifier used is Colbourn instruments' LabLinc V system shown in Figure 34. This system consists of power base, signal acquisition unit, signal processing unit, power amplifier and computer interface module. The system is so modular, as it consists of different modules. Each module has multiple channels and different number of modules can be connected vertically. In this experiment, only one module is used as 2 only channels are adopted. The module used is V75-08 module which consists of 4-channel EEG amplifier. A National Instruments NI 6030E interface card is used to interface the LAbLinc V amplifier with the pc. The card has up to 16 analog input channels, only 2 of them are used. The resolution of the acquisition, measurement, amplification and interfacing modules are 12 bits.

The software used for acquisition of the measured EEG signal, record it and export it in excel sheet is a BioBench based software. The software reads the data from each channel of the NI card and stores them in an excel sheet with the corresponding time frame.

Figure 35 shows a life experiment for EEG signal recording from one of the rats. The recorded EEG signals from the all 13 rats are preprocessed and organized in 13 different excel sheet, a different one for each rat.

(a)

(b)

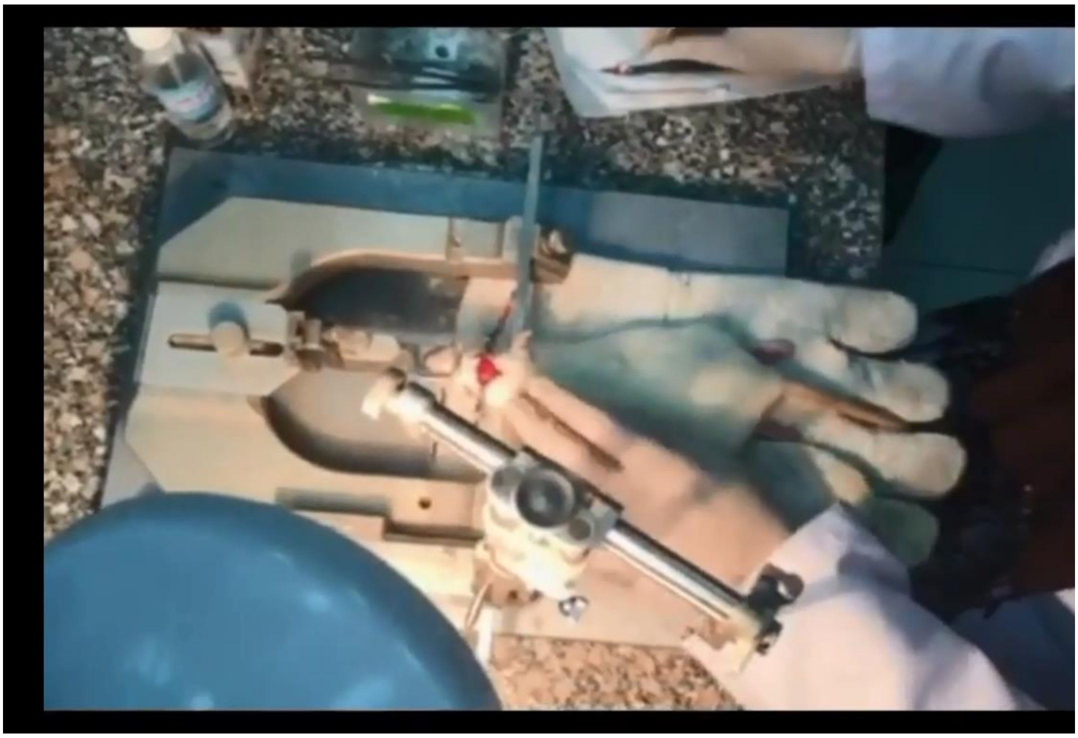


(c)

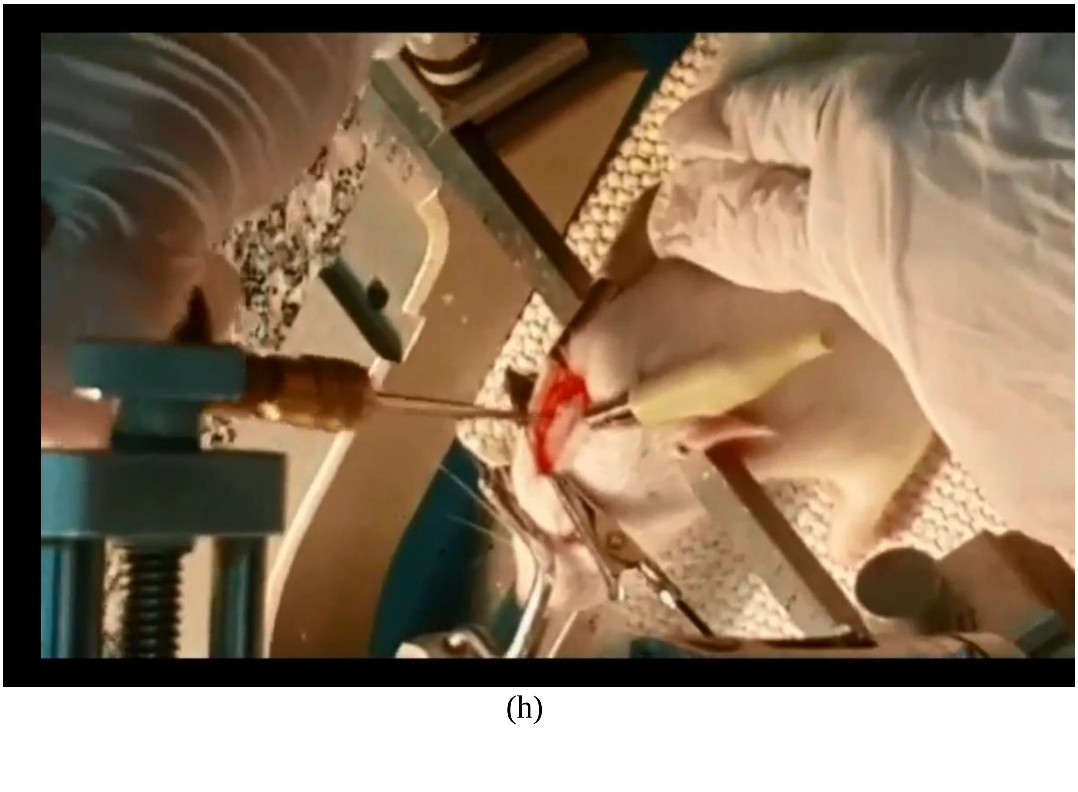
(d)


(e)

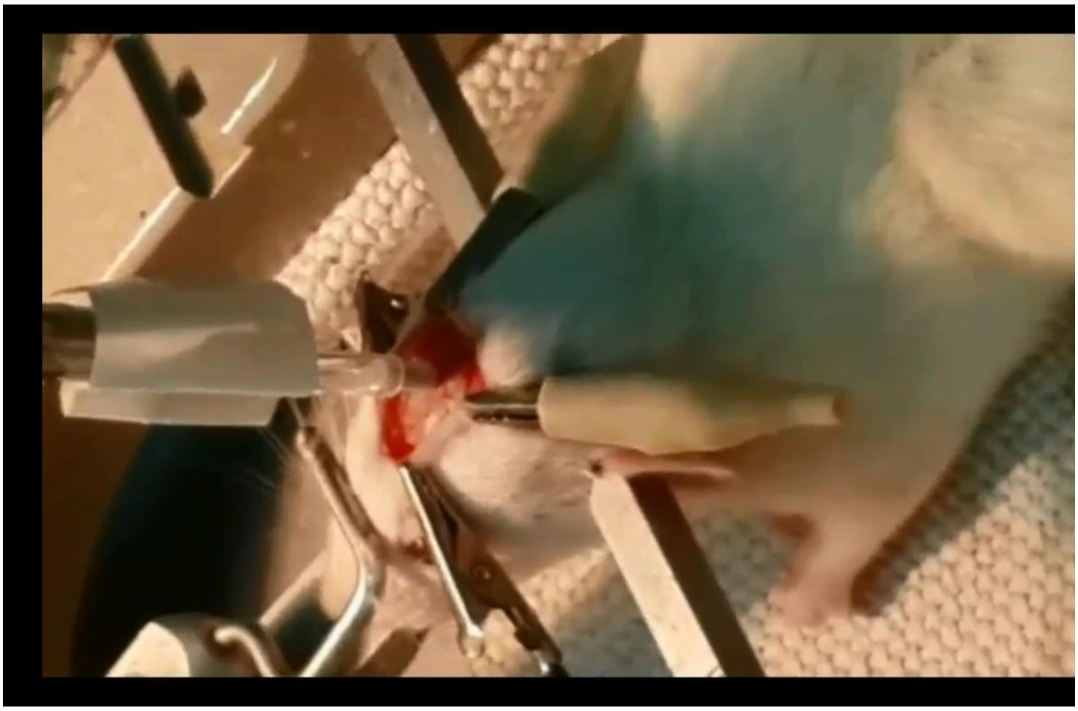

(f)



(g)



(h)



(i)

(j)

Figure 33 - Electrodes implantation surgery on rats.

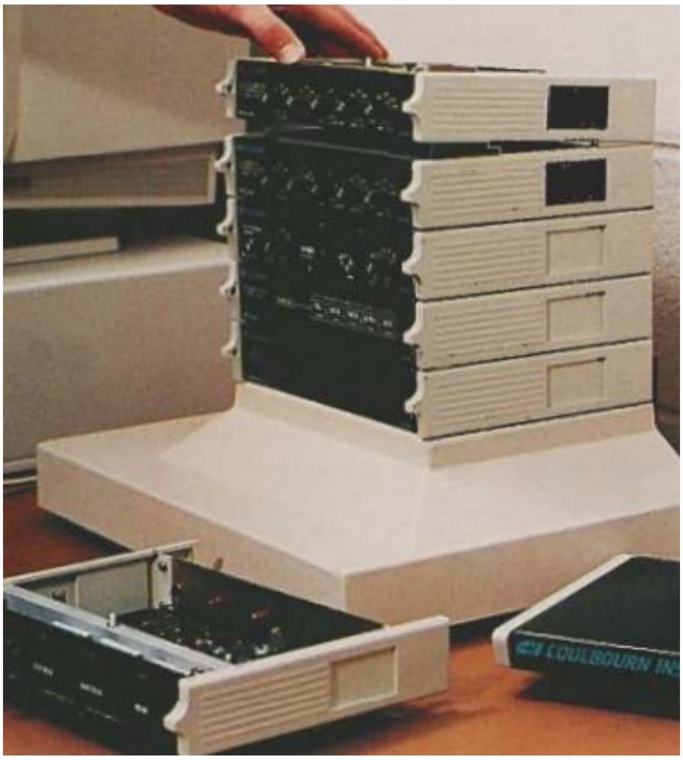
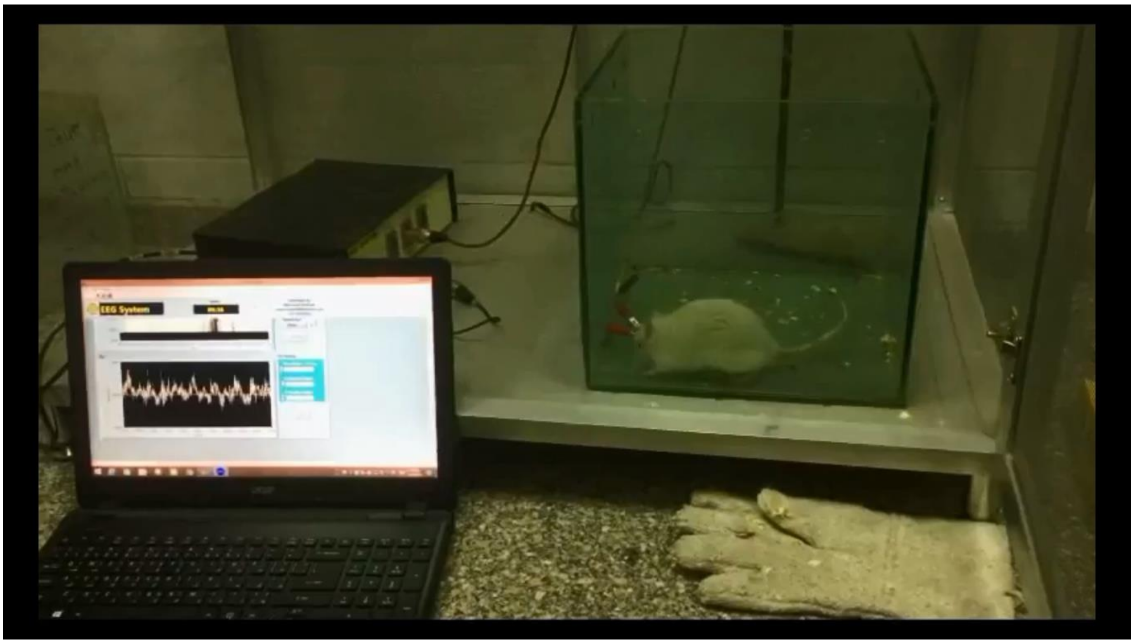
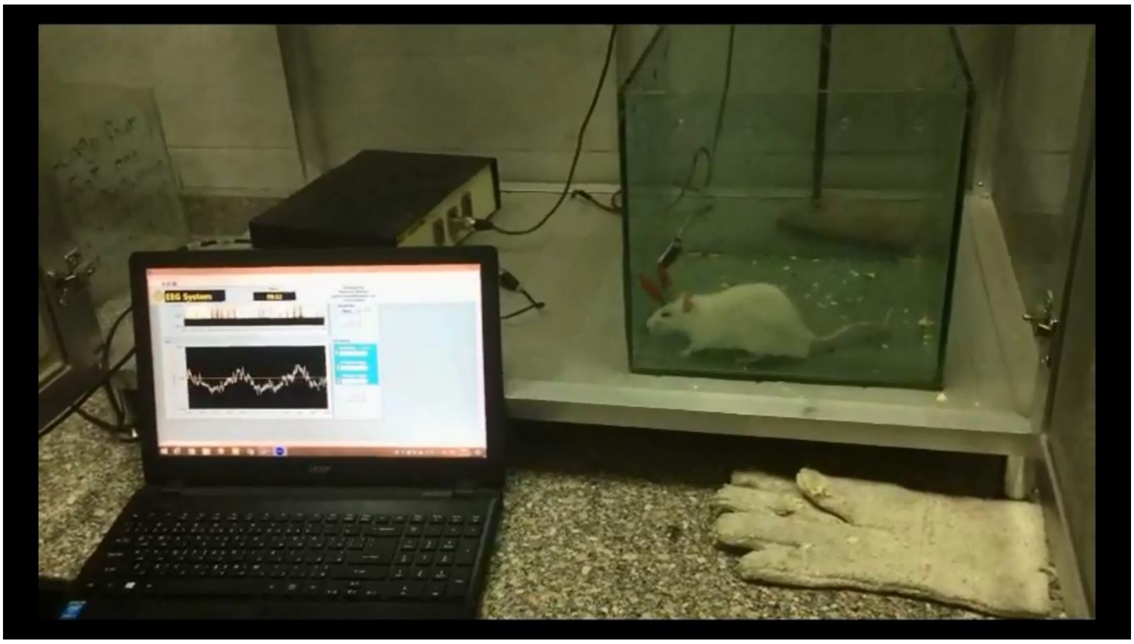


Figure 34 - LabLinc V system



(a)



(b)

Figure 35 - EEG reading experiment.

The EEG is recorded for the 13 rats in both ictal and inter-ictal periods. These EEG signals are the start of the new rats' dataset as shown in Figure 36.

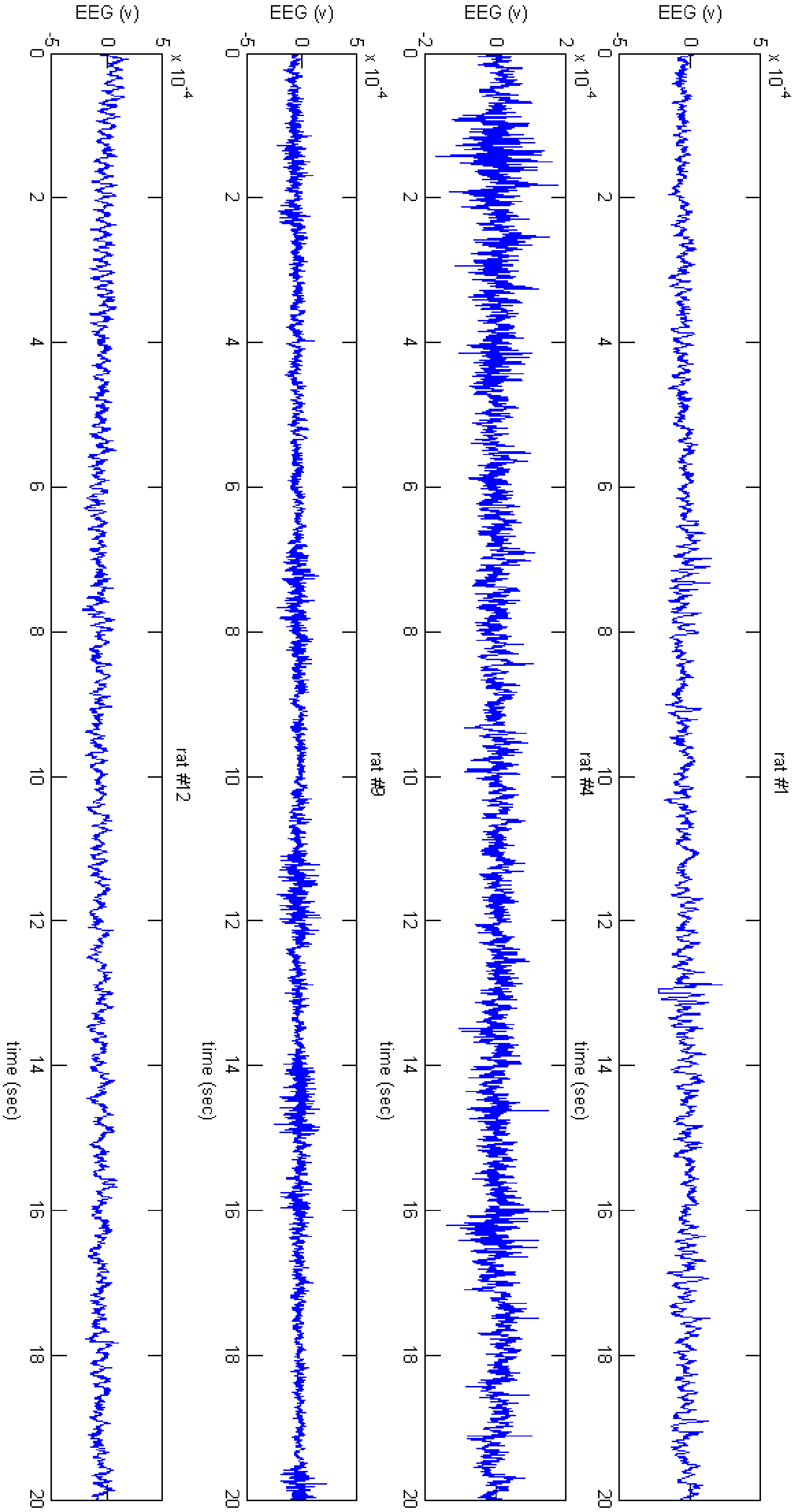


Figure 36 - Sample of the recorded rats EEG.

Conclusions and Future work

In this research, the problem of neural seizure detection problem is addressed. An automatic seizure detection system is proposed with a very-high efficiency.

As Feature extraction and selection is a key metric in enhancing the performance of classifier. More than 1100 combinations are tested with linear kernel SVM. Each combination consists of 3 features. 126 combinations of them give sensitivity between 90 and 95%. 25 combinations of them give sensitivity more than 95%, while the specificity and accuracy are more than 96% for all combinations. This result equals to and exceeds that achieved by prior work however using linear kernel function instead of the RBF kernel used in these prior work [25], [26], [53]. After exhaustive search, it is found that fractal dimension, Hurst exponent and coastline combination is the best combination that achieved sensitivity up to 96.77 % using linear kernel SVM classifier.

As the SVM learning process is a very complex process especially with the large problems like neural seizure detection, a hardware accelerator for SVM training is proposed. The training is accelerator using two different algorithms: Gradient ascent and Sequential Minimal Optimization. The implemented hardware are proposed to be used as accelerators IP especially in the problems with large training examples. The proposed accelerators achieved a sensitivity up to 96% using linear kernel function. It is found that the GA accelerator consumes less power and area than the SMO accelerator. However, the GA accelerator takes 5x clock cycles to finish training more than the SMO accelerator. That makes the GA accelerator more energy hungry than the SMO accelerator.

Then, a hardware implementation of different classifiers techniques are proposed. The proposed techniques are support vector machine (SVM) and artificial neural network (ANN). The proposed SVM is chosen with linear kernel function. On the other hand, the ANN classifier is designed with single hidden layer with 10 neurons in the hidden layer. The ANN and SVM classifiers parameters are chosen to achieve the same performance from both classifiers. For the same performance, the ANN classifier consumes less energy than the SVM classifier for each input vector. However, the instantaneous power consumed in the ANN classifier is more than that of the SVM classifier. This is due to the very large number of clock cycles needed by the SVM classifier to finalize classifying for any input vector compared to the ANN classifier.

Moreover, an effort was done to generate a new EEG dataset for rats that can be used to detect seizures in collaboration with the Faculty of Science, Cairo University and ONE lab. A DBS surgery was performed for 13 rats and depth electrodes were implanted on their cortex. The rats are injected with a specific dose of drugs that cause the rats to have a temporarily epileptic seizure. Some commercial EEG amplifiers were used to measure, amplify and record these EEG signals. The signals measured from the different rats before, during and after the seizure periods are shown in Figure 36.

As extension to this work, the following points are recommended for the future work:

- More optimizations can be done on the proposed hardware implementations to save more energy

- The dataset extracted from rats should be tested against the proposed system.
- Using the DPR capabilities of the FPGA to enhance the utilization and performance of the system.

References

[1] D. Pfaff and N. Volkow, Neuroscience in the 21st century: from basic to clinical, Springer, 2016.

[2] A. Varsavsky, M. Iven and C. Mark, Epileptic seizures and the EEG: measurement, models, detection and prediction, CRC Press, 2016.

[3] T. A. Ketter, R. M. Post and W. H. Theodore., "Positive and negative psychiatric effects of antiepileptic drugs in patients with seizure disorders," Neurology, vol. 53, pp. S53--67, 1999.

[4] M. S. Berger, J. Kincaid, G. A. Ojemann and E. Lettich, "Brain mapping techniques to maximize resection, safety, and seizure control in children with brain tumors," Neurosurgery, vol. 25, pp. 786-792, 1989.

[5] F. L. da Silva, W. Kamphuis, M. Titulaer, M. Vreugdenhil and W. Wadman, "An experimental model of progressive epilepsy: the development of kindling of the hippocampus of the rat," The Italian Journal of Neurological Sciences, vol. 16, pp. 45-57, 1995.

[6] P. Boon, R. Raedt, V. De Herdt, T. Wyckhuys and K. Vonck, "Electrical stimulation for the treatment of epilepsy," Neurotherapeutics, vol. 6, pp. 218-227, 2009.

[7] S. C. Schachter and C. B. Saper, "Vagus nerve stimulation," Epilepsia, vol. 39, pp. 677-686, 1998.

[8] J. V. Murphy and A. A. Patil, "Stimulation of the nervous system for the management of seizures," CNS drugs, vol. 17, pp. 101-115, 2003.

[9] A. Marquez, M. Dunn, J. Ciriaco and F. Farahmand, "iSeiz: A low-cost real-time seizure detection system utilizing cloud computing," in Global Humanitarian Technology Conference (GHTC), 2017.

[10] A. Ghosh, A. Sarkar, T. Das and P. Basak, "Pre-ictal epileptic seizure prediction based on ECG signal analysis," in Convergence in Technology (I2CT), 2017.

[11] J. a. P. R. Malmivuo, Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields, Oxford University Press, USA, 1995.

[12] C. J. Chu, "High density EEG—What do we have to lose?," Clinical neurophysiology: official journal of the International Federation of Clinical Neurophysiology, vol. 126, p. 433, 2015.

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

[14] J. L. Cantero, M. Atienza, R. Stickgold, M. J. Kahana, J. R. Madsen and B. Kocsis, "Sleep-dependent θ oscillations in the human hippocampus and neocortex," Journal of Neuroscience, vol. 23, pp. 10897-10903, 2003.

[15] S. Palva and J. M. Palva, "New vistas for α-frequency band oscillations," Trends in neurosciences, vol. 30, pp. 150-158, 2007.

[16] M. H. Libenson, Practical Approach to Electroencephalography E-Book, 2012.

[17] S. Blanco, S. Kochen, O. Rosso and P. Salgado, "Applying time-frequency analysis to seizure EEG activity," IEEE Engineering in medicine and biology magazine, vol. 16, pp. 64-71, 1997.

[18] J. McCarthy and E. A. Feigenbaum, "In memoriam: Arthur samuel: Pioneer in machine learning," AI Magazine, vol. 11, p. 10, 1990.

[19] A. L. Samuel, "Some studies in machine learning using the game of checkers," IBM Journal of research and development, vol. 3, pp. 210-229, 1959.

[20] U. R. Acharya, S. V. Sree, G. Swapna, R. J. Martis and J. S. Suri, "Automated EEG analysis of epilepsy: a review," Knowledge-Based Systems, vol. 45, pp. 147-165, 2013.

[21] B. Hunyadi, M. Signoretto, W. Van Paesschen, J. A. Suykens, S. Van Huffel and M. De Vos, "Incorporating structural information from the multichannel EEG improves patient-specific seizure detection," Clinical Neurophysiology, vol. 123, pp. 2352-2361, 2012.

[22] J. Wackermann, "Beyond mapping: estimating complexity of multichannel EEG recordings.," Acta neurobiologiae experimentalis, vol. 56, pp. 197-208, 1996.

[23] R. J. Croft and R. J. Barry, "Removal of ocular artifact from the EEG: a review," Neurophysiologie Clinique/Clinical Neurophysiology, vol. 30, pp. 5-19, 2000.

[24] A. Van Boxtel, "Optimal signal bandwidth for the recording of surface EMG activity of facial, jaw, oral, and neck muscles," Psychophysiology, vol. 38, pp. 22-34, 2001.

[25] Q. Yuan, W. Zhou, S. Li and D. Cai, "Epileptic EEG classification based on extreme learning machine and nonlinear features," Epilepsy research, vol. 96, pp. 29-38, 2011.

[26] S. Li, W. Zhou, Q. Yuan, S. Geng and D. Cai, "Feature extraction and recognition of ictal EEG using EMD and SVM," Computers in biology and medicine, vol. 43, pp. 807-816, 2013.

[27] R. Panda, P. Khobragade, P. Jambhule, S. Jengthe, P. Pal and T. Gandhi, "Classification of EEG signal using wavelet transform and support vector machine for epileptic seizure diction," in International Conference on Systems in Medicine and Biology (ICSMB), 2010.

[28] Kolekar, M. H, Dash and D. Prasad, "A nonlinear feature based epileptic seizure detection using least square support vector machine classifier," in TENCON 2015-2015 IEEE Region 10 Conference, 2015.

[29] S. M. Afifi, H. GholamHosseini and S. Poopak, "Hardware implementations of SVM on FPGA: A state-of-the-art review of current practice," International Journal of Innovative Science Engineering and Technology (IJISET), 2015.

[30] L. J. Cao, S. S. Keerthi, C. J. Ong, J. Q. Zhang, U. Periyathamby, X. J. Fu and H. Lee, "Parallel sequential minimal optimization for the training of support vector machines," IEEE Trans. Neural Networks, vol. 17, pp. 1039-1049, 2006.

[31] Cao, Kui-kang, S. Hai-bin and C. Hua-feng, "A parallel and scalable digital architecture for training support vector machines," Journal of Zhejiang University SCIENCE C, vol. 11, pp. 620-628, 2010.

[32] C.-H. Peng, B.-W. Chen, T.-W. Kuan, P.-C. Lin, J.-F. Wang and N.-S. Shih, "REC-STA: Reconfigurable and efficient chip design with SMO-based training accelerator," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, pp. 1791-1802, 2014.

[33] L. Bustio-Mart{\'\i}nez, R. Cumplido, J. Hern{\'a}ndez-Palancar and C. Feregrino-Uribe, "On the Design of a Hardware-Software Architecture for Acceleration of SVM's Training Phase," in Mexican Conference on Pattern Recognition, 2010.

[34] J.-F. Wang, J.-S. Peng, J.-C. Wang, P.-C. Lin and T.-W. Kuan, "Hardware/software co-design for fast-trainable speaker identification system based on SMO," in IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2011.

[35] S. Venkateshan, A. Patel and K. Varghese, "Hybrid working set algorithm for SVM learning with a kernel coprocessor on FPGA," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, pp. 2221-2232, 2015.

[36] M. B. Rabieah and C.-S. Bouganis, "FPGA based nonlinear support vector machine training using an ensemble learning," in 25th International Conference on Field Programmable Logic and Applications (FPL), 2015.

[37] W. Shaojun, P. Yu, Z. Guangquan and P. Xiyuan, "Accelerating on-line training of LS-SVM with run-time reconfiguration," in International Conference on Field-Programmable Technology (FPT), 2011.

[38] A. Bhattacharyya and B. R. Pachori, "A multivariate approach for patient-specific EEG seizure detection using empirical wavelet transform," IEEE Transactions on Biomedical Engineering, vol. 64, pp. 2003-2015, 2017.

[39] M. Sharma, B. Pachori, A. Ram and U. Rajendra, "A new approach to characterize epileptic seizures using analytic time-frequency flexible wavelet transform and fractal dimension," Pattern Recognition Letters, vol. 94, pp. 172-179, 2017.

[40] R. R. Sharma and R. B. Pachori, "Time-frequency representation using IEVDHM-HT with application to classification of epileptic EEG signals," IET Science, Measurement & Technology, vol. 12, pp. 72-82, 2017.

[41] T. Das, A. Ghosh, S. Guha and P. Basak, "Classification of EEG Signals for Prediction of Seizure using Multi-Feature Extraction," 1st International Conference on Electronics, Materials Engineering and Nano-Technology (IEMENTech), pp. 1-4, 2017.

[42] T. Zhang and W. Chen, "LMD based features for the automatic seizure detection of EEG signals using SVM," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 25, pp. 1100-1108, 2017.

[43] V. Sakkalis, Modern Electroencephalographic Assessment Techniques, 2015.

[44] S. Pincus, "Approximate entropy as an irregularity measure for financial data," Econometric Reviews, vol. 27, pp. 4-6, 2008.

[45] S. M. Pincus, "Approximate entropy as a measure of system complexity.," Proceedings of the National Academy of Sciences, vol. 88, pp. 2297-2301, 1991.

[46] P. R. Pal, N. P. Mohanty and T. Gandhi, "ENTROPY BASED DETECTION \& EVALUATION OF EPILEPTIC SEIZURE," International Journal of Applied, vol. 4, pp. 73-77, 2011.

[47] V. Vijith, J. E. Jacob, T. Iype, K. Gopakumar and D. G. Yohannan, "Epileptic seizure detection using non linear analysis of EEG," International Conference on Inventive Computation Technologies (ICICT), pp. 1-6, 2016.

[48] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," Physica D: Nonlinear Phenomena, vol. 31, pp. 277-283, 1988.

[49] P. Bhuvaneswari and J. S. Kumar, "Support vector machine technique for EEG signals," International Journal of Computer Applications, vol. 63, 2013.

[50] A. Gammerman and V. Vovk, "Alexey Chervonenkis's bibliography: introductory comments.," Journal of Machine Learning Research, vol. 16, pp. 2051-2066, 2015.

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

[52] N. Petra, D. De Caro, V. Garofalo, E. Napoli and A. G. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, pp. 1312-1325, 2010.

[53] Y. Liu, W. Zhou, Q. Yuan and S. Chen, "Automatic seizure detection using wavelet transform and SVM in long-term intracranial EEG," IEEE transactions on neural systems and rehabilitation engineering, vol. 20, no. IEEE, pp. 749-755, 2012.

[54] B. Yegnanarayana, Artificial neural networks, PHI Learning Pvt. Ltd., 2009.

[55] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, pp. 1097-1105, 2012.

[56] D. O'Leary and J. Kubby, "Feature Selection and ANN Solar Power Prediction," Journal of Renewable Energy, 2017.

Appendix A : MATLAB Simulation Codes

Main.m

```
Clc
clear
close all

% change the paths to add chb01, functions, helpfunctions in your
machine.
addpath D:\Communications\Research\New_Research\New_tools\chb01
addpath D:\Communications\Research\2018\nonlinear_features\functions
addpath
D:\Communications\Research\2018\nonlinear_features\helpFunctions
warning('off','MATLAB:legend:IgnoringExtraEntries')


%% Code for calculating different combinations of both linear and
non-linear fatures then classify the data according to the
combinations.

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Loading the data of CHB-MIT Scalp EEG Database
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
samplePerSecond = 921600/60/60;
seconds = 4;              %The number of seconds in each window
N = samplePerSecond*seconds;        % window interval

[files_names,seizure_start,seizure_ending,s_starts] = dataLoading();


%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Locating the first patient data for classification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

patient = 1;        % the first patient
file_name=files_names{patient};
start = seizure_start{patient};
ending = seizure_ending{patient};
hour = s_starts{patient};
hour = 15;
%  h=3,4,15,16,18,21,26 => contain seizures for the first patient
clear all_data
all_data=ReadEDF(file_name(hour,:));    % hour that contain seizure
over 23 channels

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Actual Seizure Locations Vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sez_true_train=zeros(floor(length(all_data{1})/N),1);
for j=1:3
    if start(hour,j)~=0
```

```
sez_true_train(floor(start(hour,j)/seconds):floor(ending(hour,j)/sec
onds),1) = ...

ones(length(floor(start(hour,j)/seconds):floor(ending(hour,j)/second
s)),1);
    end
end


% try different cominations (10C3 = 120 combinations)
c = combnk(1:20,3);
%feature1= 2;
%feature2= 12;
%feature3= 14;

 for i = 886:-1:1
 all_data=ReadEDF(file_name(hour,:));    % hour that contain seizure
over 23 channels

 temp = c(i,:);

 feature1=temp(1);
 feature2=temp(2);
 feature3=temp(3);

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Feature extraction and Ploting
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
fprintf('\nTraining ..\n');

[trainingData] = features_detection(all_data,
N,feature1,feature2,feature3);

visualize_trainingdata(trainingData,sez_true_train,'True class of
training examples',patient,hour)



%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            SVM Linear Classification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%svmTrain =
fitcsvm(trainingData,sez_true_train,'KernelFunction','RBF');   %
classes to be 1, 0
%svmTrain =
fitcsvm(trainingData,sez_true_train,'KernelFunction','polynomial','P
olynomialOrder',2);   % classes to be 1, 0
svmTrain =
svmtrain(trainingData,sez_true_train,'kernel_function','linear');
% classes to be 1, 0

fprintf('\nDone.\n');

%%
```

```
%svmClassification = predict(svmTrain,trainingData);

%visualize_trainingdata(trainingData,svmClassification,'Training Set
Classification',patient,hour)

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Testing data generation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SVM_TP=0;
SVM_TN=0;
SVM_FP=0;
SVM_FN=0;

for h = hour+1:size(file_name,1)
    if(h==20|| h==26)
        continue;
    end

    tic
    clear all_data
    all_data=ReadEDF(file_name(h,:));

    % Actual Seizure Locations Vector
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    sez_true_test=zeros(1,floor(length(all_data{1})/N));
    for j=1:3
        if start(h,j)~=0

sez_true_test(1,floor(start(h,j)/seconds):floor(ending(h,j)/seconds)
)= ...

ones(1,length(floor(start(h,j)/seconds):floor(ending(h,j)/seconds)))
;
        end
    end

    fprintf(' \nFor h = %i: \n',h);
    [testingData] = features_detection(all_data,
N,feature1,feature2,feature3);

    svmClassification = svmclassify(svmTrain,testingData);

    % plot ictal hours to see the classification on each hour
%     if(h==4||h==15||h==16||h==18||h==21||h==26)
%
visualize_testingdata(testingData,svmClassification,sez_true_test,'C
lassification of testing examples',patient,hour)
%     end

    % Performance
    %%%%%%%%%%%%%

[TP,TN,FP,FN]=detection_performance(svmClassification,sez_true_test)
;
    SVM_TP=SVM_TP+TP;
    SVM_TN=SVM_TN+TN;
```

```
    SVM_FP=SVM_FP+FP;
    SVM_FN=SVM_FN+FN;
    toc
end

SVM_sensitivity=SVM_TP/(SVM_TP+SVM_FN)*100;
SVM_specificity=SVM_TN/(SVM_TN+SVM_FP)*100;
SVM_accuracy=(SVM_TP+SVM_TN)/(SVM_TP+SVM_TN+SVM_FP+SVM_FN)*100;
results=[patient,hour,SVM_sensitivity,SVM_specificity,SVM_accuracy];
confusion_matrix = [SVM_TP SVM_FP; SVM_FN SVM_TN];
fprintf('--------\nResults:\n--------\n');

% print to the results file each iteration to record the results:

fileID = fopen('results.txt','a');

fprintf(fileID, 'Patient %i trained at hour = %i with Sensitivity =
%f , Specificity = %f and Accuracy = %f with features = [%i,%i,%i]
%s\n',...
                                        patient,hour,
SVM_sensitivity,SVM_specificity,
SVM_accuracy,feature1,feature2,feature3,datestr(now,'HH:MM:SS'));
fclose(fileID);

clearvars -except sez_true_train all_data hour patient c
samplePerSecond N seconds file_name start i ending

end
```

Feature_detection.m

```
function [trainingData] = features_detection(all_data,
N,featureNum1,featureNum2, featureNum3)

numberOfchannels=23;
for channel=1:numberOfchannels        % loop on each channel

    data=cell2mat(all_data(:,channel));

%%%%%%%%%%%%%%%%%%%%%%%% Window is 1024 samples
%%%%%%%%%%%%%%%%%%%%%%%%%%%

new_data = reshape(data,N,floor(length(data)/N));

% %   1.  Standard Deviation
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==1 || featureNum2 ==1 || featureNum3 ==1)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=STD(omar(:,i));
    end
    standarddeviation(channel,:)=oahmed;
end

% %   2.  Fractual Dimension
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==2 || featureNum2 ==2 || featureNum3 ==2)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=FD(omar(:,i));
```

```
    end
    fractualdimension(channel,:)=oahmed;
end

% %   3.  Hurst Exponent
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==3 || featureNum2 ==3 || featureNum3 ==3)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=hurstcomponent(omar(:,i),1/256);
    end
    hurstexp(channel,:)=oahmed;
end


% %   4.  Kurtosis
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==4 || featureNum2 ==4 || featureNum3 ==4)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=Pkurt(omar(:,i));
    end
    Kurtos(channel,:)=oahmed;
end

% %   5.  Skew
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==5 || featureNum2 ==5 || featureNum3 ==5)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=Pskew(omar(:,i));
    end
    skew(channel,:)=oahmed;
end

% %   6.  variance
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==6 || featureNum2 ==6 || featureNum3 ==6)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=VAR(omar(:,i));
    end
    variance(channel,:)=oahmed;
end




% %   7.  Permutation Entropy
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==7 || featureNum2 ==7 || featureNum3 ==7)
    for i=1:length(data)/N
    perEnt(channel,i)  = per_entropy(downsample(new_data(i,:),5),3);
    end
end

%   8.  Approximate Entropy
%   %%%%%%%%%%%%%%%%%%%
```

```
if(featureNum1 ==8 || featureNum2 ==8 || featureNum3 ==8)
    for i=1:length(data)/N
    approxEntropy(channel,i)  =
approxEnt(2,0.5,downsample(new_data(i,:),5));
    end
end

if(featureNum1 ==9 || featureNum2 ==9 || featureNum3 ==9)
%   9.  Shannon Entropy
    %%%%%%%%%%%%%%
    for i=1:length(data)/N
    ShannonEnt(channel,i) =
ShannonEntropy(new_data(i,:),max(new_data(i,:)),4);
    end
end
%   10.  Spectral Entropy
    %%%%%%%%%%%%%%%%%%%%
if(featureNum1 ==10 || featureNum2 ==10 || featureNum3 ==10)
    for i=1:length(data)/N
    SpectralEnt(channel, i) = SpectralEntropy(new_data(i,:),8);
    end
end
%   11.  Renyie Entropy
    %%%%%%%%%%%%%%%%%%
if(featureNum1 ==11 || featureNum2 ==11 || featureNum3 ==11)
    for i=1:length(data)/N
    renyient(channel,i) =
renyientropy(new_data(i,:),2,max(new_data(i,:)),8);
    end
end
%   12.  Hurst Exponent
%   %%%%%%%%%%%%%%%%%%
if(featureNum1 ==12 || featureNum2 ==12 || featureNum3 ==12)
    for i=1:length(data)/N
    hurstExpo(channel,i)  =
estimate_hurst_exponent(new_data(i,:),3);
    end
end
%   13.  Average Energy
% % %%%%%%%%%%%%%%%%%%
if(featureNum1 ==13 || featureNum2 ==13 || featureNum3 ==13)
    E=data.^2;
    E=E(1:floor(length(E)/N)*N,1);
    Eavg(channel,:)=1/N*sum(reshape(E,N,length(E)/N),1);%coastline
vector
end
%   14.  Coastline Feature (Fluctuation Index)
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(featureNum1 ==14 || featureNum2 ==14 || featureNum3 ==14)
    abs_bet_2_succsessive=abs([data(2:length(data));0]-data);%This
vector will have the absolute difference between two successive EEG
data points

abs_bet_2_succsessive=abs_bet_2_succsessive(1:floor(length(abs_bet_2
_succsessive)/N)*N,1);

CL(channel,:)=sum(reshape(abs_bet_2_succsessive,N,length(abs_bet_2_s
uccsessive)/N),1);%coastline vector
end
%   15.  Hjorth Parameters: Mobility
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if(featureNum1 ==15 || featureNum2 ==15 || featureNum3 ==15)
    for i=1:length(data)/N
    [mobility(channel,i),~] = HjorthParameters(new_data(i,:)');
    end
end

%   16.  Hjorth Parameters: Complexity
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(featureNum1 ==16 || featureNum2 ==16 || featureNum3 ==16)
    for i=1:length(data)/N
    [~,complexity(channel,i)] = HjorthParameters(new_data(i,:)');
    end
end

% %   17.  Mean absolute value
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==17 || featureNum2 ==17 || featureNum3 ==17)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=MAV(omar(:,i));
    end
    meanabs(channel,:)=oahmed;
end

% %   18.  Max absolute value
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==18 || featureNum2 ==18 || featureNum3 ==18)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=MAX(omar(:,i));
    end
    maxabs(channel,:)=oahmed;
end

% %   19.  Min absolute value
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==19 || featureNum2 ==19 || featureNum3 ==19)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=MIN(omar(:,i));
    end
    minabs(channel,:)=oahmed;
end

% %   20.  root mean square
%   %%%%%%%%%%%%%%%%%%%
if(featureNum1 ==20 || featureNum2 ==20 || featureNum3 ==20)
      omar=reshape(data,N,(length(data)/N));
    for i=1:(length(data)/N)
        oahmed(1,i)=RMS(omar(:,i));
    end
    rootmeansqua(channel,:)=oahmed;
end


    fprintf('%i ',channel);
end
```

```
%% constructing features:
features = zeros(numberOfchannels,floor(length(data)/N),3);
i=1;
if(featureNum1 ==1 || featureNum2 ==1 || featureNum3 ==1)
    features(:,:,i) = standardeviation;
    i= i+1;
end

if(featureNum1 ==2 || featureNum2 ==2 || featureNum3 ==2)
    features(:,:,i) = fractualdimension;
    i= i+1;
end

if(featureNum1 ==3 || featureNum2 ==3 || featureNum3 ==3)
    features(:,:,i) = hurstexp;
    i= i+1;
end

if(featureNum1 ==4 || featureNum2 ==4 || featureNum3 ==4)
    features(:,:,i) = Kurtos;
    i= i+1;
end

if(featureNum1 ==5 || featureNum2 ==5 || featureNum3 ==5)
    features(:,:,i) = skew;
    i= i+1;
end

if(featureNum1 ==6 || featureNum2 ==6 || featureNum3 ==6)
    features(:,:,i) = variance;
    i= i+1;
end



if(featureNum1 ==7 || featureNum2 ==7 || featureNum3 ==7)
    features(:,:,i) = perEnt;
    i= i+1;
end
if (featureNum1 ==8 || featureNum2 ==8 || featureNum3 ==8)
     features(:,:,i) = approxEntropy;
    i= i+1;
end
if (featureNum1 ==9 || featureNum2 ==9 || featureNum3 ==9)
     features(:,:,i) = ShannonEnt;
     i = i +1;
end
if (featureNum1 ==10 || featureNum2 ==10 || featureNum3 ==10)
     features(:,:,i) = SpectralEnt;
     i = i +1;
end
if (featureNum1 ==11 || featureNum2 ==11 || featureNum3 ==11)
    features(:,:,i) = renyient;
    i = i +1;
end
if (featureNum1 ==12 || featureNum2 ==12 || featureNum3 ==12)
     features(:,:,i) = hurstExpo;
     i = i +1;
end
if (featureNum1 ==13 || featureNum2 ==13 || featureNum3 ==13)
```

```
     features(:,:,i) = Eavg;
     i = i +1;
end
if (featureNum1 ==14 || featureNum2 ==14 || featureNum3 ==14)
     features(:,:,i) = CL;
     i = i +1;
end
if (featureNum1 ==15 || featureNum2 ==15 || featureNum3 ==15)
    features(:,:,i) =  mobility;
    i = i +1;
end
if (featureNum1 ==16 || featureNum2 ==16 || featureNum3 ==16)
     features(:,:,i) = complexity;
     i = i +1;
end
if (featureNum1 ==17 || featureNum2 ==17 || featureNum3 ==17)
     features(:,:,i) = meanabs;
     i = i +1;
end
if (featureNum1 ==18 || featureNum2 ==18 || featureNum3 ==18)
     features(:,:,i) = maxabs;
     i = i +1;
end

if (featureNum1 ==19 || featureNum2 ==19 || featureNum3 ==19)
     features(:,:,i) = minabs;
     i = i +1;
end

if (featureNum1 ==20 || featureNum2 ==20 || featureNum3 ==20)
     features(:,:,i) = rootmeansqua;
     i = i +1;
end

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Combine the channels into an average channel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

feature1 = features(:,:,1);
feature2 = features(:,:,2);
feature3 = features(:,:,3);

feature1_train=sum(feature1,1)/numberOfchannels;
feature2_train=sum(feature2,1)/numberOfchannels;
feature3_train=sum(feature3,1)/numberOfchannels;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Features Normalization & Training
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

trainingData=[feature1_train' feature2_train' feature3_train'];

mean1=nanmean(trainingData(:,1));
mean2=nanmean(trainingData(:,2));
mean3=nanmean(trainingData(:,3));


var1=nanvar(trainingData(:,1));
```

```
var2=nanvar(trainingData(:,2));
var3=nanvar(trainingData(:,3));

trainingData(:,1)=(trainingData(:,1)-mean1)/sqrt(var1);
trainingData(:,2)=(trainingData(:,2)-mean2)/sqrt(var2);
trainingData(:,3)=(trainingData(:,3)-mean3)/sqrt(var3);

end
```

approxEnt.m

```
function [apen] = approxEnt(window_length,r,data)


%% Code for computing approximate entropy for a time series:
Approximate

% To run this function- type: approx_entropy('window
length','similarity measure','data set')
% i.e  approx_entropy(5,0.5,data)
% Author: Avinash Parnandi, parnandi@usc.edu,
http://robotics.usc.edu/~parnandi/

%%


for m=window_length:window_length+1     % to be able to calculate
the phi(r)^m - phi(r)^(m+1)

set = 0;
count = 0;
counter = 0;

for i=1:(length(data))-m+1
    current_window = data(i:i+m-1); % current window stores the
sequence to be compared with other sequences

    for j=1:length(data)-m+1
    sliding_window = data(j:j+m-1); % get a window for comparision
with the current_window

    % compare two windows, element by element
    % can also use some kind of norm measure; that will perform
better
    for k=1:m
        if((abs(current_window(k)-sliding_window(k))>r) && set == 0)
            set = 1; % i.e. the difference between the two sequence
is greater than the given value
        end
    end
    if(set==0)
         count = count+1; % this measures how many sliding_windows
are similar to the current_window
    end
    set = 0; % reseting 'set'

    end
   counter(i)=count/(length(data)-m+1); % need the number of similar
windows for every cuurent_window
```

```
    count=0;

end

correlation(m-window_length+1) = ((sum(counter))/(length(data)-
m+1));


end

apen = log(correlation(1)/correlation(2));
end
```

Estimate_hurst_exponent.m

```
function [hurst] = estimate_hurst_exponent(data,no_iterations)

[~,npoints]=size(data);
yvals = zeros(1,no_iterations);
xvals = zeros(1,no_iterations);

k=1;
for i = 10:(npoints/no_iterations):npoints

original_signal= data(1:i);

signal_mean = sum(original_signal)/npoints;
X = original_signal - signal_mean;
Y = cumsum(X);

Rn = max(Y) - min(Y);
original_std = std(original_signal);

yvals(k) = log(Rn/original_std);
xvals(k) = log(i);
k = k+1;

end

p2=polyfit(xvals,yvals,1);
hurst=p2(1);                         % Hurst exponent is the slope of
the linear fit of log-log plot

end
```

HjorthParameters.m

```
function [mobility,complexity] = HjorthParameters(xV)

n = length(xV);
dxV = diff([0;xV]);
ddxV = diff([0;dxV]);
mx2 = mean(xV.^2);
mdx2 = mean(dxV.^2);
mddx2 = mean(ddxV.^2);

mob = mdx2 / mx2;
complexity = sqrt(mddx2 / mdx2 - mob);
mobility = sqrt(mob);
```

```
end
```

Per_entropy.m

```
function perEnt = per_entropy(data,win)


for i = 1:length(data)-floor(win/2)-1

[~,I(i,:)] = sort(data(i:i+win-1));

end

[~,jj,kk]=unique(I,'rows','stable');
f=histc(kk,1:numel(jj)); % Frequency
P = f/length(data);

perEnt= -sum(P.*log(P));
end
```

Quantizer.m

```
function [quantized_signal] = quantizer(sampled_signal,varargin)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Sample of input for quantizer funtion:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% ts = 0.1;
%%% nLevels = 5;
%%% mp = 5;
%%% m_law=2;
%%% [binary_signal,level_signal,quantized_signal] =
quantizer(sampled_sig,'NLevels', nLevels,
%%%                                                       'SigMax',
mp, 'QuantizerType', 0,'MeuValue',m_law);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Input Oarsing Handeling
quantizationType = 1;
mp = max(sampled_signal);
nLevels = 4;
meu = 1;

p = inputParser();
addOptional(p, 'QuantizerType', quantizationType, @isnumeric);
addOptional(p, 'NLevels', nLevels, @isnumeric);
addOptional(p, 'MeuValue', meu, @isnumeric);
addOptional(p, 'SigMax', mp, @isnumeric);
parse(p, varargin{:});

nLevels = p.Results.NLevels;
mp = p.Results.SigMax;


if (2^(ceil(log2(nLevels))) > nLevels)
    disp('Number of Levels must be multiple of 2');
    nLevels = 2^(ceil(log2(nLevels)));
    fprintf('A %d number of levels was chosen instead \n',nLevels);
```

```
end

%% Uniform mid-rise quantizer

quantized_signal = zeros(size(sampled_signal));
level_signal= zeros(size(sampled_signal));
detla = 2*mp/(nLevels-1);

for n =1:length(sampled_signal)
    current_level = -mp;
    level_number = 0;
    for k= 1:nLevels

        if((sampled_signal(n) <= current_level && sampled_signal(n)
>= current_level - detla/2) || (sampled_signal(n) >= current_level
&& sampled_signal(n) <= current_level + detla/2))
            quantized_signal(n) = current_level;
            level_signal(n) = level_number;
            break;
        end

        level_number = level_number + 1;
        current_level = current_level + detla;

    end
end
end
```

Renyientropy.m

```
function RENYI = renyientropy(X,alpha,sig_Max,levels)

    % Number of levels for quantization and the signal maximum value
    [quantized] = quantizer(X,'NLevels', levels,'SigMax', sig_Max);

    unique_values = unique(quantized);

    Frequency = zeros(size(unique_values));

    % Calculate sample frequencies
    for level = 1:length(unique_values)
        Frequency(level) = sum(quantized == unique_values(level));
    end

    % Calculate sample class probabilities
    P = Frequency / sum(Frequency);

    % Calculate Shannon Entropy
    RENYI=(1/1-alpha).* log2(sum(P .^alpha));
end
```

sampEntropy.m

```
function [ApEn] = sampEntropy(window_length,r,data)


%% Code for computing approximate entropy for a time series: Sample
```

```
% To run this function- type: approx_entropy('window
length','similarity measure','data set')
% i.e  approx_entropy(5,0.5,data)
% Author: Avinash Parnandi, parnandi@usc.edu,
http://robotics.usc.edu/~parnandi/

%%


for m=window_length:window_length+1     % to be able to calculate
the phi(r)^m - phi(r)^(m+1)

set = 0;
count = 0;
counter = 0;

for i=1:(length(data))-m+1
    current_window = data(i:i+m-1); % current window stores the
sequence to be compared with other sequences

    for j=1:length(data)-m+1

        if i==j
            continue;
        end

    sliding_window = data(j:j+m-1); % get a window for comparision
with the current_window

    % compare two windows, element by element
    % can also use some kind of norm measure; that will perform
better
    for k=1:m
        if((abs(current_window(k)-sliding_window(k))>r) && set == 0)
            set = 1; % i.e. the difference between the two sequence
is greater than the given value
        end
    end
    if(set==0)
         count = count+1; % this measures how many sliding_windows
are similar to the current_window
    end
    set = 0; % reseting 'set'

    end
   counter(i)=count/(length(data)-m+1); % need the number of similar
windows for every cuurent_window
   count=0;

end

correlation(m-window_length+1) = ((sum(counter))/(length(data)-
m+1));


end

ApEn = log(correlation(1)/correlation(2));
```

```
end
```

ShannonEntropy.m

```
function H = ShannonEntropy(X,sig_Max,levels)

    % Number of levels for quantization and the signal maximum value
    [quantized] = quantizer(X,'NLevels', levels,'SigMax', sig_Max);

    unique_values = unique(quantized);

    Frequency = zeros(size(unique_values));

    % Calculate sample frequencies
    for level = 1:length(unique_values)
        Frequency(level) = sum(quantized == unique_values(level));
    end

    % Calculate sample class probabilities
    P = Frequency / sum(Frequency);

    % Calculate Shannon Entropy
    H = -sum(P .* log(P));

end
```

SpectralEntropy.m

```
function Entropy = SpectralEntropy(y,levels)

Fs = 100;

Y = fft(y);
Y = Y(1:floor(length(y)/2)+1);
Y = 1/(length(y)*Fs)*(Y.*conj(Y));
df = 1000/length(y);
freq = 0:df:500;

PSD = Y.^2/length(y);
Normalized_PSD = PSD/sum(PSD);

quantized_PSD = quantizer(Normalized_PSD,'NLevels', levels,'SigMax',
max(Normalized_PSD));


% Sampling in Frequency:

Entropy = -sum(Normalized_PSD.*log(Normalized_PSD));

end
```

ACF.m

```
function y=ACF(x,k)
ck=0;
xbar=MAV(x);
for i=1:(length(x)-k)
    ck=ck+((x(i)-xbar)*(x(i+k)-xbar));
end
```

```
ck=ck/length(x);
c0=VAR(x);
y=ck/c0;
end
```

FD.m

```
function p=FD(x)
x1=x(1);
x2=x(2);
x3=x(3);
x4=x(4);
x5=x(5);

for i=1:((length(x)-1)/5) %m?
    x1=[x1 x(1+(5*i))];
end

for i=1:((length(x)-2)/5)
    x2=[x2 x(2+(5*i))];
end

for i=1:((length(x)-3)/5)
    x3=[x3 x(3+(5*i))];
end

for i=1:((length(x)-4)/5)
    x4=[x4 x(4+(5*i))];
end

for i=1:((length(x)-5)/5)
    x5=[x5 x(5+(5*i))];
end
a1=(length(x)-1)/5;
a2=(length(x)-2)/5;
a3=(length(x)-3)/5;
a4=(length(x)-4)/5;
a5=(length(x)-5)/5;


L1=0;
for i=1:a1
   L1=L1+(abs(x(1+(i*5))-x(1+((i-1)*5)))/(length(x)-1));
end
L1=L1/(a1*5);


L2=0;
for i=1:a2
   L2=L2+(abs(x(2+(i*5))-x(2+((i-1)*5)))/(length(x)-1));
end
L2=L2/(a2*5);


L3=0;
for i=1:a3
   L3=L3+(abs(x(3+(i*5))-x(3+((i-1)*5)))/(length(x)-1));
end
L3=L3/(a3*5);
```

```
L4=0;
for i=1:a4
   L4=L4+(abs(x(4+(i*5))-x(4+((i-1)*5))))/(length(x)-1));
end
L4=L4/(a4*5);


L5=0;
for i=1:a5
   L5=L5+(abs(x(5+(i*5))-x(5+((i-1)*5))))/(length(x)-1));
end
L5=L5/(a5*5);

k=(log(L1)/log(1/5));
q=(log(L2)/log(1/5));
r=(log(L3)/log(1/5));
s=(log(L4)/log(1/5));
u=(log(L5)/log(1/5));
p=(k+q+r+s+u)/5;
end
```

Hurstcomponent.m

```
function H=hurstcomponent(x,T)
data=x; %adding input in internal variable
average=MAV(data);
differences=data-average;
maxdevfrommean=MAX(differences);
mindevfrommean=MIN(differences);
R=abs(abs(maxdevfrommean)-abs(mindevfrommean));
S=STD(data);
H=log(R/S)/log(T);
end
```

MAV.m

```
function y=MAV(x)
temp=abs(x);
y=sum(temp)/length(x);
end
```

MAX.m

```
function y=MAX(x)
temp1=x(1);
for i=1:length(x);
    if(abs(x(i))>abs(temp1))
        temp1=x(i);
    elseif(abs(x(i))==abs(temp1))
        if(angle(x(i))>angle(temp1))
            temp1=x(i);
        else
            temp1=temp1;
        end
    else
        temp1=temp1;
    end
end
y=temp1;
end
```

Min.m

```
function y=MIN(x)
temp1=x(1);
for i=1:length(x);
    if(abs(x(i))<abs(temp1))
        temp1=x(i);
    elseif(abs(x(i))==abs(temp1))
        if(angle(x(i))<angle(temp1))
            temp1=x(i);
        else
            temp1=temp1;
        end
    else
        temp1=temp1;
    end
end
y=temp1;
```

Pkurt.m

```
function y=Pkurt(x)
X=x;
averageofX=sum(X)/length(X);
stdofX=STD(x);
y=sum((((X-averageofX)/stdofX).^4))/length(X);
end
```

Pmax.m

```
function y=Pmax(x)
y=MAX(fft(x));
max(x)
end
```

Pskew.m

```
function y=Pskew(x)
X=x;
averageofX=sum(X)/length(X);
stdofX=STD(x);
y=sum((((X-averageofX)/stdofX).^3))/length(X);
end
```

RMS.m

```
function y=RMS(x)
temp=x.*x;
y=sqrt(sum(temp)/length(x));
end
```

STD.m

```
function y=STD(x)
averageofX=sum(x)/length(x);
y=sqrt(sum(((x-averageofX).*(x-averageofX)))/(length(x)-1));
end
```

VAR.m

```
function y=VAR(x)
averageofX=sum(x)/length(x);
y=(sum(((x-averageofX).*(x-averageofX)))/(length(x)-1));
```

```
end
```

dataLoading.m

```
function [files_names,seizure_start,seizure_ending,s_starts] =
dataLoading()

  file_1=['chb01_01.edf'; 'chb01_02.edf'; 'chb01_03.edf';
'chb01_04.edf'; 'chb01_05.edf'; 'chb01_06.edf'; 'chb01_07.edf';
'chb01_08.edf'; 'chb01_09.edf'; 'chb01_10.edf'; 'chb01_11.edf';
'chb01_12.edf'; 'chb01_13.edf'; 'chb01_14.edf'; 'chb01_15.edf';
'chb01_16.edf'; 'chb01_17.edf'; 'chb01_18.edf'; 'chb01_19.edf';
'chb01_20.edf'; 'chb01_21.edf'; 'chb01_22.edf'; 'chb01_23.edf';
'chb01_24.edf'; 'chb01_25.edf'; 'chb01_26.edf'; 'chb01_27.edf';
'chb01_29.edf'; 'chb01_30.edf'; 'chb01_31.edf'; 'chb01_32.edf';
'chb01_33.edf'; 'chb01_34.edf'; 'chb01_36.edf'; 'chb01_37.edf';
'chb01_38.edf'; 'chb01_39.edf'; 'chb01_40.edf'; 'chb01_41.edf';
'chb01_42.edf'; 'chb01_43.edf'; 'chb01_46.edf'];
 start_1=[0    0 0; 0    0 0; 2996 0 0; 1467 0 0; 0    0 0; 0    0
0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 1732 0 0; 1015 0 0; 0    0 0; 1720 0 0; 0    0 0; 0
0 0; 327  0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 1862 0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0];
ending_1=[0    0 0; 0    0 0; 3036 0 0; 1494 0 0; 0    0 0; 0    0
0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 1772 0 0; 1066 0 0; 0    0 0; 1810 0 0; 0    0 0; 0
0 0; 420  0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 1963 0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0];
s_start_1=3;

  file_2=['chb02_01.edf'; 'chb02_02.edf'; 'chb02_03.edf';
'chb02_04.edf'; 'chb02_05.edf'; 'chb02_06.edf'; 'chb02_07.edf';
'chb02_08.edf'; 'chb02_09.edf'; 'chb02_10.edf'; 'chb02_11.edf';
'chb02_12.edf'; 'chb02_13.edf'; 'chb02_14.edf'; 'chb02_15.edf';
'chb02_16.edf'; 'chb02_17.edf'; 'chb02_18.edf'; 'chb02_19.edf';
'chb02_20.edf'; 'chb02_22.edf'; 'chb02_23.edf'; 'chb02_24.edf';
'chb02_25.edf'; 'chb02_26.edf'; 'chb02_27.edf'; 'chb02_28.edf';
'chb02_29.edf'; 'chb02_30.edf'; 'chb02_31.edf'; 'chb02_32.edf';
'chb02_33.edf'; 'chb02_34.edf'; 'chb02_35.edf'];
 start_2=[0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0
0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0
; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0
; 130  0 0;    0    0 0; 0    0 0; 3369 0 0;   0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0];
ending_2=[0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0
0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0
; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0
; 212  0 0;   0    0 0; 0    0 0; 3378 0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0];
s_start_2=16;
  file_3=['chb03_01.edf'; 'chb03_02.edf'; 'chb03_03.edf';
'chb03_04.edf'; 'chb03_05.edf'; 'chb03_06.edf'; 'chb03_07.edf';
'chb03_08.edf'; 'chb03_09.edf'; 'chb03_10.edf'; 'chb03_11.edf';
'chb03_12.edf'; 'chb03_13.edf'; 'chb03_14.edf'; 'chb03_15.edf';
'chb03_16.edf'; 'chb03_17.edf'; 'chb03_18.edf'; 'chb03_19.edf';
```

```
'chb03_20.edf'; 'chb03_21.edf'; 'chb03_22.edf'; 'chb03_23.edf';
'chb03_24.edf'; 'chb03_25.edf'; 'chb03_26.edf'; 'chb03_27.edf';
'chb03_28.edf'; 'chb03_29.edf'; 'chb03_30.edf'; 'chb03_31.edf';
'chb03_32.edf'; 'chb03_33.edf'; 'chb03_34.edf'; 'chb03_35.edf';
'chb03_36.edf'; 'chb03_37.edf'; 'chb03_38.edf'];
 start_3=[362  0 0; 731  0 0; 432  0 0; 2162 0 0;  0    0 0; 0    0
0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0;
1982 0 0;  2592 0 0;  1725 0 0;  0    0 0; 0    0 0];
ending_3=[414  0 0; 796  0 0; 501  0 0; 2214 0 0;  0    0 0; 0    0
0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0;
2029 0 0;  2656 0 0;  1778 0 0;  0    0 0; 0    0 0];
s_start_3=4;
  file_4=['chb04_01.edf'; 'chb04_02.edf'; 'chb04_03.edf';
'chb04_04.edf'; 'chb04_05.edf'; 'chb04_06.edf'; 'chb04_07.edf';
'chb04_08.edf'; 'chb04_09.edf'; 'chb04_10.edf'; 'chb04_11.edf';
'chb04_12.edf'; 'chb04_13.edf'; 'chb04_14.edf'; 'chb04_15.edf';
'chb04_16.edf'; 'chb04_17.edf'; 'chb04_18.edf'; 'chb04_19.edf';
'chb04_21.edf'; 'chb04_22.edf'; 'chb04_23.edf'; 'chb04_24.edf';
'chb04_25.edf'; 'chb04_26.edf'; 'chb04_27.edf'; 'chb04_28.edf';
'chb04_29.edf'; 'chb04_30.edf'; 'chb04_31.edf'; 'chb04_32.edf';
'chb04_33.edf'; 'chb04_34.edf'; 'chb04_35.edf'; 'chb04_36.edf';
'chb04_37.edf'; 'chb04_38.edf'; 'chb04_39.edf'; 'chb04_40.edf';
'chb04_41.edf'; 'chb04_42.edf'; 'chb04_43.edf'];
 start_4=[0    0 0; 0    0 0; 0    0 0; 0    0 0; 7804 0 0; 0    0
0; 0    0 0; 6446 0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0;
1679 3782 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0];
ending_4=[0    0 0; 0    0 0; 0    0 0; 0    0 0; 7853 0 0; 0    0
0; 0    0 0; 6557 0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0;
1781 3898 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0    0 0; 0
0 0; 0    0 0; 0    0 0];
s_start_4=8;
  file_5=['chb05_01.edf'; 'chb05_02.edf'; 'chb05_03.edf';
'chb05_04.edf'; 'chb05_05.edf'; 'chb05_06.edf'; 'chb05_07.edf';
'chb05_08.edf'; 'chb05_09.edf'; 'chb05_10.edf'; 'chb05_11.edf';
'chb05_12.edf'; 'chb05_13.edf'; 'chb05_14.edf'; 'chb05_15.edf';
'chb05_16.edf'; 'chb05_17.edf'; 'chb05_18.edf'; 'chb05_19.edf';
'chb05_20.edf'; 'chb05_21.edf'; 'chb05_22.edf'; 'chb05_23.edf';
'chb05_24.edf'; 'chb05_25.edf'; 'chb05_26.edf'; 'chb05_27.edf';
'chb05_28.edf'; 'chb05_29.edf'; 'chb05_30.edf'; 'chb05_31.edf';
'chb05_32.edf'; 'chb05_33.edf'; 'chb05_34.edf'; 'chb05_35.edf';
'chb05_36.edf'; 'chb05_37.edf'; 'chb05_38.edf'; 'chb05_39.edf'];
 start_5=[0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;417 0 0;0 0 0;0 0 0;0 0 0;0
0 0;0 0 0;0 0 0;1086 0 0;0 0 0;0 0 0;2317 0 0;2451 0 0;0 0 0;0 0 0;0
0 0;0 0 0;2348 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0
0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0];
ending_5=[0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;532 0 0;0 0 0;0 0 0;0 0 0;0
0 0;0 0 0;0 0 0;1196 0 0;0 0 0;0 0 0;2413 0 0;2571 0 0;0 0 0;0 0 0;0
```

```
0 0;0 0 0;2465 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0
0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0;0 0 0];
s_start_5=6;
  file_6=['chb06_01.edf'; 'chb06_02.edf'; 'chb06_03.edf';
'chb06_04.edf'; 'chb06_05.edf'; 'chb06_06.edf'; 'chb06_07.edf';
'chb06_08.edf'; 'chb06_09.edf'; 'chb06_10.edf'; 'chb06_12.edf';
'chb06_13.edf'; 'chb06_14.edf'; 'chb06_15.edf'; 'chb06_16.edf';
'chb06_17.edf'; 'chb06_18.edf'; 'chb06_24.edf'];
 start_6=[ 1724   7461   13525  ;  0   0   0  ;  0   0   0  ;  327
6211   0  ;  0   0   0  ;  0   0   0  ;  0   0   0  ;  0   0   0  ;
12500   0   0  ;  10833   0   0  ;  0   0   0  ;  506   0   0  ;  0
0   0  ;  0   0   0  ;  0   0   0  ;  0   0   0  ;  7799   0   0  ;
9387   0   0 ];
ending_6=[ 1738  7476  13540 ; 0  0  0 ; 0  0  0 ; 347  6231  0 ; 0
0  0 ; 0  0  0 ; 0  0  0 ; 0  0  0 ; 12516  0  0 ; 10845  0  0 ; 0
0  0 ; 519  0  0 ; 0  0  0 ; 0  0  0 ; 0  0  0 ; 0  0  0 ; 7811  0
0 ; 9403  0  0 ];
s_start_6=10;
  file_7=['chb07_01.edf'; 'chb07_02.edf'; 'chb07_03.edf';
'chb07_04.edf'; 'chb07_05.edf'; 'chb07_06.edf'; 'chb07_07.edf';
'chb07_08.edf'; 'chb07_09.edf'; 'chb07_10.edf'; 'chb07_11.edf';
'chb07_12.edf'; 'chb07_13.edf'; 'chb07_14.edf'; 'chb07_15.edf';
'chb07_16.edf'; 'chb07_17.edf'; 'chb07_18.edf'; 'chb07_19.edf'];
 start_7=[ 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0
0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 4920 0 0 ; 3285 0 0 ; 0 0 0 ; 0 0 0 ;
0 0 0 ; 0 0 0 ; 0 0 0 ; 13688 0 0 ];
ending_7=[ 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0
0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 5006 0 0 ; 3381 0 0 ; 0 0 0 ; 0 0 0 ;
0 0 0 ; 0 0 0 ; 0 0 0 ; 13831 0 0 ];
s_start_7=12;
  file_8=['chb08_02.edf'; 'chb08_03.edf'; 'chb08_04.edf';
'chb08_05.edf'; 'chb08_10.edf'; 'chb08_11.edf'; 'chb08_12.edf';
'chb08_13.edf'; 'chb08_14.edf'; 'chb08_15.edf'; 'chb08_16.edf';
'chb08_17.edf'; 'chb08_18.edf'; 'chb08_19.edf'; 'chb08_20.edf';
'chb08_21.edf'; 'chb08_22.edf'; 'chb08_23.edf'; 'chb08_24.edf';
'chb08_29.edf'];
 start_8=[ 2670 0 0 ; 0 0 0 ; 0 0 0 ; 2856 0 0 ; 0 0 0 ; 2988 0 0 ;
0 0 0 ; 2417 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0
0 0 ; 2083 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ];
ending_8=[  2841 0 0 ; 0 0 0 ; 0 0 0 ; 3046 0 0 ; 0 0 0 ; 3122 0 0 ;
0 0 0 ; 2577 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0
0 0 ; 2347 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ; 0 0 0 ];
s_start_8=4;
  file_9=['chb09_01.edf'; 'chb09_02.edf'; 'chb09_03.edf';
'chb09_04.edf'; 'chb09_05.edf'; 'chb09_06.edf'; 'chb09_07.edf';
'chb09_08.edf'; 'chb09_09.edf'; 'chb09_10.edf'; 'chb09_11.edf';
'chb09_12.edf'; 'chb09_13.edf'; 'chb09_14.edf'; 'chb09_15.edf';
'chb09_16.edf'; 'chb09_17.edf'; 'chb09_18.edf'; 'chb09_19.edf'];
 start_9=[0   0   0 ; 0   0   0 ; 0   0   0 ; 0   0   0 ; 0   0   0 ; 12231   0
0 ; 0   0   0 ; 2951   9196   0 ; 0   0   0 ; 0   0   0 ;
0   0   0 ; 0   0   0 ; 0   0   0 ; 0   0   0 ; 0   0   0 ; 0   0   0 ; 0   0   0
; 0   0   0 ; 5299   0 0];
ending_9=[0 0   0 ;0 0   0 ;0 0   0 ;0 0   0 ;0 0   0 ;12295 0   0 ;0 0   0
;3030 9267   0 ;0 0   0 ;0 0   0 ;0 0   0 ;0 0   0 ;0 0   0 ;0 0   0 ;0 0
0 ;0 0   0 ;0 0   0 ;0 0   0 ;5361 0 0];
s_start_9=6;
  file_10=['chb10_01.edf'; 'chb10_02.edf'; 'chb10_03.edf';
'chb10_04.edf'; 'chb10_05.edf'; 'chb10_06.edf'; 'chb10_07.edf';
'chb10_08.edf'; 'chb10_12.edf'; 'chb10_13.edf'; 'chb10_14.edf';
'chb10_15.edf'; 'chb10_16.edf'; 'chb10_17.edf'; 'chb10_18.edf';
'chb10_19.edf'; 'chb10_20.edf'; 'chb10_21.edf'; 'chb10_22.edf';
```

```
'chb10_27.edf'; 'chb10_28.edf'; 'chb10_30.edf'; 'chb10_31.edf';
'chb10_38.edf'; 'chb10_89.edf'];
 start_10=[ 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0
0 0 ; 0  0 0 ; 6313  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0
0 ; 0  0 0 ; 0  0 0 ; 6888  0 0 ; 0  0 0 ; 0  0 0 ; 2382  0 0 ; 0  0
0 ; 3021  0 0 ; 3801  0 0 ; 4618  0 0 ; 1383  0 0 ];
ending_10=[ 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0
0 0 ; 0  0 0 ; 6348  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0 0 ; 0  0
0 ; 0  0 0 ; 0  0 0 ; 6958  0 0 ; 0  0 0 ; 0  0 0 ; 2447  0 0 ; 0  0
0 ; 3079  0 0 ; 3877  0 0 ; 4707  0 0 ; 1437  0 0];
s_start_10=23;

files_names = {file_1, file_2, file_3, file_4, file_5, file_6,
file_7, file_8, file_9, file_10};
seizure_start = {start_1, start_2, start_3, start_4, start_5,
start_6, start_7, start_8, start_9, start_10};
seizure_ending = {ending_1, ending_2, ending_3, ending_4, ending_5,
ending_6, ending_7, ending_8, ending_9, ending_10};
s_starts = {s_start_1, s_start_2, s_start_3, s_start_4, s_start_5,
s_start_6, s_start_7, s_start_8, s_start_9, s_start_10};
```

Detection_Performance.m

```
function
[TP,TN,FP,FN]=detection_performance(Classification,seizure_true)
TP=0;TN=0;FP=0;FN=0;

for i=1:length(Classification)
    if(Classification(i)==1)&&(seizure_true(1,i)==1)
        TP=TP+1;
    elseif(Classification(i)==0)&&(seizure_true(1,i)==0)
        TN=TN+1;
    elseif(Classification(i)==1)&&(seizure_true(1,i)==0)
        FP=FP+1;
    elseif(Classification(i)==0)&&(seizure_true(1,i)==1)
        FN=FN+1;
    end
end
```

ReadEDF.m

```
function [data, header] = ReadEDF(filename)

% Author:  Shapkin Andrey,
% 15-OCT-2012

% filename - File name
% data - Contains a signals in structure of cells
% header  - Contains header

fid = fopen(filename,'r','ieee-le');

%%% HEADER LOAD
% PART1: (GENERAL)
hdr = char(fread(fid,256,'uchar')');
header.ver=str2num(hdr(1:8));            % 8 ascii : version of this
data format (0)
header.patientID  = char(hdr(9:88));     % 80 ascii : local patient
identification
header.recordID  = char(hdr(89:168));    % 80 ascii : local
recording identification
```

```
header.startdate=char(hdr(169:176));     % 8 ascii : startdate of
recording (dd.mm.yy)
header.starttime  = char(hdr(177:184));  % 8 ascii : starttime of
recording (hh.mm.ss)
header.length = str2num (hdr(185:192));  % 8 ascii : number of bytes
in header record
reserved = hdr(193:236); % [EDF+C       ] % 44 ascii : reserved
header.records = str2num (hdr(237:244)); % 8 ascii : number of data
records (-1 if unknown)
header.duration = str2num (hdr(245:252)); % 8 ascii : duration of a
data record, in seconds
header.channels = str2num (hdr(253:256));% 4 ascii : number of
signals (ns) in data record

%%%% PART2 (DEPENDS ON QUANTITY OF CHANNELS)

header.labels=cellstr(char(fread(fid,[16,header.channels],'char')'))
; % ns * 16 ascii : ns * label (e.g. EEG FpzCz or Body temp)
header.transducer
=cellstr(char(fread(fid,[80,header.channels],'char')')); % ns * 80
ascii : ns * transducer type (e.g. AgAgCl electrode)
header.units =
cellstr(char(fread(fid,[8,header.channels],'char')')); % ns * 8
ascii : ns * physical dimension (e.g. uV or degreeC)
header.physmin =
str2num(char(fread(fid,[8,header.channels],'char')')); % ns * 8
ascii : ns * physical minimum (e.g. -500 or 34)
header.physmax =
str2num(char(fread(fid,[8,header.channels],'char')')); % ns * 8
ascii : ns * physical maximum (e.g. 500 or 40)
header.digmin =
str2num(char(fread(fid,[8,header.channels],'char')')); % ns * 8
ascii : ns * digital minimum (e.g. -2048)
header.digmax =
str2num(char(fread(fid,[8,header.channels],'char')')); % ns * 8
ascii : ns * digital maximum (e.g. 2047)
header.prefilt
=cellstr(char(fread(fid,[80,header.channels],'char')')); % ns * 80
ascii : ns * prefiltering (e.g. HP:0.1Hz LP:75Hz)
header.samplerate =
str2num(char(fread(fid,[8,header.channels],'char')')); % ns * 8
ascii : ns * nr of samples in each data record
reserved = char(fread(fid,[32,header.channels],'char')'); % ns * 32
ascii : ns * reserved


f1=find(cellfun('isempty', regexp(header.labels, 'EDF Annotations',
'once'))==0); % Channels number with the EDF Annotations
f2=find(cellfun('isempty', regexp(header.labels, 'Status',
'once'))==0); % Channels number with the EDF Annotations
f=[f1(:); f2(:)];
%%%%%% PART 3: Loading of signals

%Structure of the data in format EDF:

%[block1 block2 .. , block N], where N=header.records
% Block structure:
% [(d seconds of 1 channel) (d seconds of 2 channel) ... (d seconds
of ï¿½h channel)], Where ï¿½h - quantity of channels, d - duration
of the block
```

```
% Ch = header.channels
% d = header.duration

Ch_data = fread(fid,'int16'); % Loading of signals


fclose(fid); % close a file

%%%%% PART 4: Transformation of the data
if header.records<0, % If the quantity of blocks is not known
R=sum(header.duration*header.samplerate); % Length of one block
header.records=fix(length(Ch_data)./R); % Quantity of written down
blocks
end

% Separating a read signal into blocks
Ch_data=reshape(Ch_data, [], header.records);

% establishing calibration parametres




sf = (header.physmax - header.physmin)./(header.digmax -
header.digmin);
dc = header.physmax - sf.* header.digmax;

data=cell(1, header.channels);
Rs=cumsum([1; header.duration*header.samplerate]); %
ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½
ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ï¿½ Rs(k):Rs(k+1)-1


% separating of signals of everyone the channel from blocks
% and recording of signals in structure of cells

for k=1:header.channels

data{k}=reshape(Ch_data(Rs(k):Rs(k+1)-1, :), [], 1);
if sum(k==f)==0 % non ï¿½nnotation
% Calibration of the data
data{k}=data{k}.*sf(k)+dc(k);
end
end

% PART 5: ANNOTATION READ
    header.annotation.event={};
    header.annotation.starttime=[];
    header.annotation.duration=[];
    header.annotation.data={};

if sum(f)>0

try

for p1=1:length(f)
Annt=char(typecast(int16(data{f(p1)}), 'uint8'))';
```

```
% separate of annotation on blocks
Annt=buffer(Annt, header.samplerate(f(p1)).*2, 0)';
ANsize=size(Annt);
    for p2=1:ANsize(1)
   % search TALs starttime
    Annt1=Annt(p2, :);
    Tstart=regexp(Annt1, '+');
    Tstart=[Tstart(2:end) ANsize(2)];

    for p3=1:length(Tstart)-1
   A=Annt1(Tstart(p3):Tstart(p3+1)-1); % TALs block
   header.annotation.data={header.annotation.data{:} A};

       % duration and starttime TALs
        Tds=find(A==20 | A==21);
         if length(Tds)>2
             td=str2num(A(Tds(1)+1:Tds(2)-1));
             if isempty(td), td=0; end

header.annotation.duration=[header.annotation.duration(:); td];

header.annotation.starttime=[header.annotation.starttime(:);
str2num(A(2:Tds(1)-1))];
            header.annotation.event={header.annotation.event{:}
A(Tds(2)+1:Tds(end)-1)};
          else

header.annotation.duration=[header.annotation.duration(:); 0];

header.annotation.starttime=[header.annotation.starttime(:);
str2num(A(2:Tds(1)-1))];
            header.annotation.event={header.annotation.event{:}
A(Tds(1)+1:Tds(end)-1)};
        end
    end
    end
end

% delete annotation
a=find(cell2mat(cellfun(@length, header.annotation.event,
'UniformOutput', false))==0);
header.annotation.event(a)=[];
header.annotation.starttime(a)=[];
header.annotation.duration(a)=[];

end

end

header.samplerate(f)=[];
header.channels=header.channels-length(f);
header.labels(f)=[];
header.transducer(f)=[];
header.units(f)=[];
header.physmin(f)=[];
header.physmax(f)=[];
header.digmin(f)=[];
header.digmax(f)=[];
header.prefilt(f)=[];
data(f)=[];
```

```
end
```

Visualize_testingData.m

```
function
visualize_testingdata(testingData,svmClassification,sez_true_test,te
xt,patient,h)

figure
subplot(3,1,1)

gscatter(testingData(:,1),testingData(:,2),
svmClassification,'br','x+')
hold on
gscatter(testingData(:,1),testingData(:,2), sez_true_test,'kb','oo')
legend('Predicted Non-ictal','Predicted Ictal','Actual Non-
ictal','Actual Ictal')
% title(text + ' for h = '+ string(h) + ' from patient #'+
string(patient))
xlabel('feature 1');
ylabel('feature 2');
hold off


subplot(3,1,2)

gscatter(testingData(:,1),testingData(:,3),
svmClassification,'br','x+')
hold on
gscatter(testingData(:,1),testingData(:,3), sez_true_test,'kb','oo')
legend('Predicted Non-ictal','Predicted Ictal','Actual Non-
ictal','Actual Ictal')
% title(text + ' for h = '+string(h) + ' from patient
#'+string(patient))
xlabel('feature 1');
ylabel('feature 3');
hold off


subplot(3,1,3)

gscatter(testingData(:,2),testingData(:,3),
svmClassification,'br','x+')
hold on
gscatter(testingData(:,2),testingData(:,3), sez_true_test,'kb','oo')
legend('Predicted Non-ictal','Predicted Ictal','Actual Non-
ictal','Actual Ictal')
% title(text + ' for h = '+string(h) + ' from patient
#'+string(patient))
xlabel('feature 2');
ylabel('feature 3');
hold off


end
```

visualize_trainingdata.m

```
function
visualize_trainingdata(trainingData,sez_true_train,text,patient,hour
)
```

```
figure
gscatter((trainingData(:,1)),(trainingData(:,2)),
sez_true_train,'br','xo')

hold on
legend('Non-ictal','Ictal')
%title(string(text) + ' for h = '+string(hour) + ' from patient
#'+string(patient))
%xlabel('Mean Absolute Value');
%ylabel('RMS');

hold off
figure;
figure
subplot(3,1,1)
gscatter((trainingData(:,1)),(trainingData(:,2)),
sez_true_train,'br','xo')

hold on
legend('Non-ictal','Ictal')
%title(string(text) + ' for h = '+string(hour) + ' from patient
#'+string(patient))
xlabel('feature 1');
ylabel('feature 2');

hold off

subplot(3,1,2)
gscatter((trainingData(:,1)),(trainingData(:,3)),
sez_true_train,'br','xo')
hold on
legend('Non-ictal','Ictal')
%title(string(text) + ' for h = '+string(hour) + ' from patient
#'+string(patient))
xlabel('feature 1');
ylabel('feature 3');

hold off

subplot(3,1,3)
gscatter((trainingData(:,2)),(trainingData(:,3)),
sez_true_train,'br','xo')
hold on
legend('Non-ictal','Ictal')
%title(string(text) + ' for h = '+string(hour) + ' from patient
#'+string(patient))
xlabel('feature 2');
ylabel('feature 3');

hold off
end
```

Linear_grad_svm.m

```
function [model] = linear_grad_svm(xt,y,Q)


N=length(xt);
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
alpha=zeros(N,1);
b=0;
alpha_new=zeros(N,1);
skip=zeros(N,1);
C=1;
margin=1.5*1e-7;
%step=1e-10;
step=1e-7;
%step=0.0016;
keep_search=1;
alpha_hist=zeros(100000,15);
k=1;

while(keep_search && k<1000)
%for k=1:100000
        %acc_w=zeros(1,size(xt,2));
        acc_w=0;
    for i=1:N
        acc=0;
        for j=1:N
acc=acc+alpha(j,:)*y(j,:)*((xt(i,:)*xt(j,:)'+1).^Q);
                 %acc=acc+alpha(j,:)*y(j,:)*((xt(i,:)*xt(j,:)'));
        end
        alpha_new(i,1)= alpha(i,1)-(step*((y(i,:)*(acc+b))-1));
        %             alpha_new(i,1)= 1-step*(y(i,:)*acc);
        if alpha_new(i,1)>C
            alpha_new(i,1) = C;
            skip(i,1)=1;
        elseif alpha_new(i,1) < 0
            alpha_new(i,1) = 0;
            skip(i,1)=1;
        end
        %acc_w=acc_w+alpha(i)*y(i)*xt(i,:);
        %acc_w=acc_w+alpha(i)*y(i);
        %acc_w=acc_w + ((xt(i,:)*xt(2,:)' +1)^Q);
%          alpha_new(i,1)=min(C,max(0, alpha(i,1)-
step*(y(i,:)*(acc+b)-1)));
    end
    %b_new=b-step*(alpha'*y);
    W=(alpha_new.*y)';
    SV=1;
    for l=1:N
        if(alpha_new(l)~=0)
            SV=l;
            break;
        end
    end
    b_new=y(SV) - (alpha_new.*y)'*((xt*xt(SV,:)'+1).^Q);
    %b_new=y(3) - W*xt*xt(3,:)';
%MA
     %b_new=y(3)-acc_w*xt(3,:)';
     %b_new = y(2) -acc_w
     %b_new=y(1)- (alpha.*y)'*((xt*(xt(1,:)') +1).^Q)
%MA_end


    comp=sum(abs([alpha;b]-[alpha_new;b_new]))>margin;
    alpha=alpha_new;
```

```
    b=b_new;
    %alpha_hist(k,:)=alpha;
    %keep_search=sum(comp);
    keep_search=comp;

    k=k+1

    %plot_svm(x1,x2,W,b);
    %pause;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
W=(alpha.*y)'*xt;
model.w=W;
model.b=b;
model.alpha=alpha(alpha~=0);
model.xt=xt(alpha~=0,:);
model.y=y(alpha~=0);
sum(model.y)
size(model.y)
end
```

Smo_training_fn.m

```
function [model]=smo_train_fn(X,Y,Q)

tol = 1e-23;
max_passes = 100;
% Data parameters
m = size(X, 1);
n = size(X, 2);
% Map 0 to -1
Y(Y==0) = -1;
% Variables
alphas = zeros(m, 1);
b = 0;
E = zeros(m, 1);
passes = 0;
eta = 0;
L = 0;
H = 0;
C = 50;
K = (X*X'+1).^Q;
% K = X*X';
% Train
dots = 12;
while passes < max_passes,
    num_changed_alphas = 0;
    for i = 1:m,
        % Calculate Ei = f(x(i)) - y(i) using (2).
        % E(i) = b + sum (X(i, :) * (repmat(alphas.*Y,1,n).*X)') -
Y(i);
        E(i) = b + sum (alphas.*Y.*K(:,i)) - Y(i);
        if ((Y(i)*E(i) < -tol && alphas(i) < C) || (Y(i)*E(i) > tol
&& alphas(i) > 0)),
            % In practice, there are many heuristics one can use to
select
            % the i and j. In this simplified code, select them
randomly.
```

```
%               j = ceil(m * rand());
%               while j == i,  % Make sure i \neq j
%                   j = ceil(m * rand());
%               end
          for j=[1:i-1,i+1:m]
             % Calculate Ej = f(x(j)) - y(j) using (2).
             E(j) = b + sum (alphas.*Y.*K(:,j)) - Y(j);
             % Save old alphas
             alpha_i_old = alphas(i);
             alpha_j_old = alphas(j);
             % Compute L and H by (10) or (11).
             if (Y(i) == Y(j)),
                 L = max(0, alphas(j) + alphas(i) - C);
                 H = min(C, alphas(j) + alphas(i));
             else
                 L = max(0, alphas(j) - alphas(i));
                 H = min(C, C + alphas(j) - alphas(i));
             end
             if (L == H),
                 % continue to next i.
                 continue;
             end
             % Compute eta by (14).
             eta = 2 * K(i,j) - K(i,i) - K(j,j);
             if (eta >= 0),
                 % continue to next i.
                 continue;
             end
             % Compute and clip new value for alpha j using (12) and
(15).
             alphas(j) = alphas(j) - (Y(j) * (E(i) - E(j))) / eta;
             % Clip
             alphas(j) = min (H, alphas(j));
             alphas(j) = max (L, alphas(j));
             % Check if change in alpha is significant
             if (abs(alphas(j) - alpha_j_old) < tol),
                 % continue to next i.
                 % replace anyway
                 alphas(j) = alpha_j_old;
                 continue;
             end
             % Determine value for alpha i using (16).
             alphas(i) = alphas(i) + Y(i)*Y(j)*(alpha_j_old -
alphas(j));
             % Compute b1 and b2 using (17) and (18) respectively.
             b1 = b - E(i) ...
                  - Y(i) * (alphas(i) - alpha_i_old) *  K(i,i)' ...
                  - Y(j) * (alphas(j) - alpha_j_old) *  K(i,j)';
             b2 = b - E(j) ...
                  - Y(i) * (alphas(i) - alpha_i_old) *  K(i,j)' ...
                  - Y(j) * (alphas(j) - alpha_j_old) *  K(j,j)';
             % Compute b by (19).
             if (0 < alphas(i) && alphas(i) < C),
                 b = b1;
             elseif (0 < alphas(j) && alphas(j) < C),
                 b = b2;
             else
                 b = (b1+b2)/2;
             end
             num_changed_alphas = num_changed_alphas + 1;
          end
```

```
        end
    end
%     if (num_changed_alphas == 0),
        passes = passes + 1;
%     else
%         passes = 0;
%     end

%     X=X((find(alphas~=0)),:);
%     Y=Y((find(alphas~=0)),:);
%     alphas=alphas((find(alphas~=0)),:);
%     K = (X*X'+1).^Q;
%     m = size(X, 1);

    fprintf('.');
    dots = dots + 1;
    if dots > 78
        dots = 0;
        fprintf('\n');
    end
end
fprintf(' Done! \n\n');
% Save the model
idx = alphas > 0;
model.X= X(idx,:);
model.Y= Y(idx);
model.b= b;
model.alphas= alphas(idx);
model.w = ((alphas.*Y)'*X)';
end
```

Appendix B - Detailed feature selection results

Feature1	***Feature2***	***Feature3***	***Sensitivity***	***Specificity***	***Accuracy***
Max Absolute Value	Min Absolute Value	Root Mean Square	82.25807	98.22944	98.18391
Mean Absolute Value	Min Absolute Value	Root Mean Square	87.09677	97.86057	97.82989
Mean Absolute Value	Max Absolute Value	Root Mean Square	83.87097	98.2156	98.17471
Mean Absolute Value	Max Absolute Value	Min Absolute Value	87.09677	97.91129	97.88046
Hjorth Complexity	Mean Absolute Value	Max Absolute Value	83.87097	98.3862	98.34483
Hjorth Complexity	Mean Absolute Value	Min Absolute Value	85.48387	98.12339	98.08736
Hjorth Complexity	Mean Absolute Value	Root Mean Square	83.87097	98.28938	98.24828
Hjorth Complexity	Max Absolute Value	Min Absolute Value	0	100	99.71494
Hjorth Complexity	Max Absolute Value	Root Mean Square	80.64516	98.50148	98.45058
Hjorth Complexity	Min Absolute Value	Root Mean Square	83.87097	98.20177	98.16092
Hjorth Mobility	Hjorth Complexity	Mean Absolute Value	85.48387	98.17872	98.14253
Hjorth Mobility	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Hjorth Mobility	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Hjorth Mobility	Hjorth Complexity	Root Mean Square	82.25807	98.2986	98.25287
Hjorth Mobility	Mean Absolute Value	Max Absolute Value	83.87097	98.48303	98.44138
Hjorth Mobility	Mean Absolute Value	Min Absolute Value	85.48387	98.07728	98.04138
Hjorth Mobility	Mean Absolute Value	Root Mean Square	82.25807	98.31243	98.26667
Hjorth Mobility	Max Absolute Value	Min Absolute Value	0	100	99.71494
Hjorth Mobility	Max Absolute Value	Root Mean Square	80.64516	98.48764	98.43678
Hjorth Mobility	Min Absolute Value	Root Mean Square	83.87097	98.22944	98.18851
Coastline	Min Absolute Value	Root Mean Square	85.48387	97.98967	97.95402
Coastline	Max Absolute Value	Root Mean Square	80.64516	98.33087	98.28046

Coastline	Max Absolute Value	Min Absolute Value	0	100	99.71494
Coastline	Mean Absolute Value	Root Mean Square	83.87097	98.06806	98.02759
Coastline	Mean Absolute Value	Min Absolute Value	87.09677	97.81907	97.78851
Coastline	Mean Absolute Value	Max Absolute Value	83.87097	98.23405	98.1931
Coastline	Hjorth Complexity	Root Mean Square	80.64516	98.3862	98.33563
Coastline	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Coastline	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Coastline	Hjorth Complexity	Mean Absolute Value	85.48387	98.22021	98.18391
Coastline	Hjorth Mobility	Root Mean Square	83.87097	98.34471	98.30345
Coastline	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Coastline	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Coastline	Hjorth Mobility	Mean Absolute Value	85.48387	98.25249	98.21609
Coastline	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Average Energy	Min Absolute Value	Root Mean Square	72.58065	98.61214	98.53793
Average Energy	Max Absolute Value	Root Mean Square	67.74194	98.99023	98.90115
Average Energy	Max Absolute Value	Min Absolute Value	56.45161	98.80579	98.68506
Average Energy	Mean Absolute Value	Root Mean Square	74.19355	98.72741	98.65747
Average Energy	Mean Absolute Value	Min Absolute Value	75.80645	98.51531	98.45058
Average Energy	Mean Absolute Value	Max Absolute Value	75.80645	98.79657	98.73103
Average Energy	Hjorth Complexity	Root Mean Square	67.74194	98.981	98.89195
Average Energy	Hjorth Complexity	Min Absolute Value	61.29032	99.02711	98.91954
Average Energy	Hjorth Complexity	Max Absolute Value	62.90323	99.18388	99.08046
Average Energy	Hjorth Complexity	Mean Absolute Value	69.35484	98.94412	98.85977
Average Energy	Hjorth Mobility	Root Mean Square	67.74194	98.95334	98.86437
Average Energy	Hjorth Mobility	Min Absolute Value	61.29032	98.98561	98.87816

Average Energy	Hjorth Mobility	Max Absolute Value	61.29032	99.15161	99.04368
Average Energy	Hjorth Mobility	Mean Absolute Value	67.74194	98.9349	98.84598
Average Energy	Hjorth Mobility	Hjorth Complexity	62.90323	99.07322	98.97012
Average Energy	Coastline	Root Mean Square	61.29032	98.8519	98.74483
Average Energy	Coastline	Min Absolute Value	61.29032	98.79196	98.68506
Average Energy	Coastline	Max Absolute Value	58.06452	99.064	98.94713
Average Energy	Coastline	Mean Absolute Value	70.96774	98.7689	98.68966
Average Energy	Coastline	Hjorth Complexity	62.90323	99.06861	98.96552
Average Energy	Coastline	Hjorth Mobility	62.90323	99.0225	98.91954
Hurst Exponent	Min Absolute Value	Root Mean Square	83.87097	97.90668	97.86667
Hurst Exponent	Max Absolute Value	Root Mean Square	83.87097	98.17872	98.13793
Hurst Exponent	Max Absolute Value	Min Absolute Value	0	100	99.71494
Hurst Exponent	Mean Absolute Value	Root Mean Square	85.48387	97.80524	97.77012
Hurst Exponent	Mean Absolute Value	Min Absolute Value	87.09677	97.74069	97.71035
Hurst Exponent	Mean Absolute Value	Max Absolute Value	87.09677	98.07728	98.04598
Hurst Exponent	Hjorth Complexity	Root Mean Square	83.87097	98.17872	98.13793
Hurst Exponent	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Hurst Exponent	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Hurst Exponent	Hjorth Complexity	Mean Absolute Value	85.48387	97.92051	97.88506
Hurst Exponent	Hjorth Mobility	Root Mean Square	85.48387	97.63003	97.5954
Hurst Exponent	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Hurst Exponent	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Hurst Exponent	Hjorth Mobility	Mean Absolute Value	85.48387	97.47787	97.44368
Hurst Exponent	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Hurst Exponent	Coastline	Root Mean Square	85.48387	97.87901	97.84368

Hurst Exponent	Coastline	Min Absolute Value	0	100	99.71494
Hurst Exponent	Coastline	Max Absolute Value	0	100	99.71494
Hurst Exponent	Coastline	Mean Absolute Value	87.09677	97.73146	97.70115
Hurst Exponent	Coastline	Hjorth Complexity	0	100	99.71494
Hurst Exponent	Coastline	Hjorth Mobility	0	100	99.71494
Hurst Exponent	Average Energy	Root Mean Square	72.58065	98.69974	98.62529
Hurst Exponent	Average Energy	Min Absolute Value	62.90323	98.82424	98.72184
Hurst Exponent	Average Energy	Max Absolute Value	59.67742	99.17466	99.06207
Hurst Exponent	Average Energy	Mean Absolute Value	74.19355	98.59369	98.52414
Hurst Exponent	Average Energy	Hjorth Complexity	72.58065	98.47842	98.4046
Hurst Exponent	Average Energy	Hjorth Mobility	82.25807	97.81907	97.77471
Hurst Exponent	Average Energy	Coastline	62.90323	98.9349	98.83218
Renyie Entropy	Min Absolute Value	Root Mean Square	82.25807	97.7868	97.74253
Renyie Entropy	Max Absolute Value	Root Mean Square	82.25807	98.5107	98.46437
Renyie Entropy	Max Absolute Value	Min Absolute Value	0	100	99.71494
Renyie Entropy	Mean Absolute Value	Root Mean Square	82.25807	98.16488	98.11954
Renyie Entropy	Mean Absolute Value	Min Absolute Value	87.09677	97.74991	97.71954
Renyie Entropy	Mean Absolute Value	Max Absolute Value	83.87097	98.34932	98.30805
Renyie Entropy	Hjorth Complexity	Root Mean Square	80.64516	98.33549	98.28506
Renyie Entropy	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Renyie Entropy	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Renyie Entropy	Hjorth Complexity	Mean Absolute Value	85.48387	98.22944	98.1931
Renyie Entropy	Hjorth Mobility	Root Mean Square	82.25807	98.3401	98.29425
Renyie Entropy	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Renyie Entropy	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Renyie Entropy	Hjorth Mobility	Mean Absolute Value	85.48387	98.19716	98.16092

Renyie Entropy	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Renyie Entropy	Coastline	Root Mean Square	77.41936	98.12339	98.06437
Renyie Entropy	Coastline	Min Absolute Value	0	100	99.71494
Renyie Entropy	Coastline	Max Absolute Value	0	100	99.71494
Renyie Entropy	Coastline	Mean Absolute Value	79.03226	98.11417	98.05977
Renyie Entropy	Coastline	Hjorth Complexity	0	100	99.71494
Renyie Entropy	Coastline	Hjorth Mobility	0	100	99.71494
Renyie Entropy	Average Energy	Root Mean Square	72.58065	98.62597	98.55172
Renyie Entropy	Average Energy	Min Absolute Value	56.45161	98.86573	98.74483
Renyie Entropy	Average Energy	Max Absolute Value	64.51613	99.0225	98.92414
Renyie Entropy	Average Energy	Mean Absolute Value	75.80645	98.52914	98.46437
Renyie Entropy	Average Energy	Hjorth Complexity	62.90323	99.064	98.96092
Renyie Entropy	Average Energy	Hjorth Mobility	62.90323	99.07783	98.97471
Renyie Entropy	Average Energy	Coastline	58.06452	98.8104	98.69425
Renyie Entropy	Hurst Exponent	Root Mean Square	85.48387	97.80985	97.77471
Renyie Entropy	Hurst Exponent	Min Absolute Value	0	100	99.71494
Renyie Entropy	Hurst Exponent	Max Absolute Value	0	100	99.71494
Renyie Entropy	Hurst Exponent	Mean Absolute Value	87.09677	97.72685	97.69655
Renyie Entropy	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Renyie Entropy	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Renyie Entropy	Hurst Exponent	Coastline	0	100	99.71494
Renyie Entropy	Hurst Exponent	Average Energy	62.90323	98.90262	98.8
Spectral Entropy	Min Absolute Value	Root Mean Square	85.48387	97.81446	97.77931
Spectral Entropy	Max Absolute Value	Root Mean Square	83.87097	98.40926	98.36782
Spectral Entropy	Max Absolute Value	Min Absolute Value	0	100	99.71494
Spectral Entropy	Mean Absolute Value	Root Mean Square	87.09677	98.06806	98.03678
Spectral Entropy	Mean Absolute Value	Min Absolute Value	87.09677	97.80524	97.77471
Spectral Entropy	Mean Absolute Value	Max Absolute Value	83.87097	98.24327	98.2023

Spectral Entropy	Hjorth Complexity	Root Mean Square	80.64516	98.22021	98.17012
Spectral Entropy	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Spectral Entropy	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Spectral Entropy	Hjorth Complexity	Mean Absolute Value	85.48387	98.13261	98.09655
Spectral Entropy	Hjorth Mobility	Root Mean Square	83.87097	98.1695	98.12874
Spectral Entropy	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Spectral Entropy	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Spectral Entropy	Hjorth Mobility	Mean Absolute Value	83.87097	98.20177	98.16092
Spectral Entropy	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Spectral Entropy	Coastline	Root Mean Square	82.25807	98.0035	97.95862
Spectral Entropy	Coastline	Min Absolute Value	0	100	99.71494
Spectral Entropy	Coastline	Max Absolute Value	0	100	99.71494
Spectral Entropy	Coastline	Mean Absolute Value	85.48387	98.04961	98.01379
Spectral Entropy	Coastline	Hjorth Complexity	0	100	99.71494
Spectral Entropy	Coastline	Hjorth Mobility	0	100	99.71494
Spectral Entropy	Average Energy	Root Mean Square	67.74194	98.86573	98.77701
Spectral Entropy	Average Energy	Min Absolute Value	62.90323	98.80579	98.70345
Spectral Entropy	Average Energy	Max Absolute Value	62.90323	99.08244	98.97931
Spectral Entropy	Average Energy	Mean Absolute Value	72.58065	98.75046	98.67586
Spectral Entropy	Average Energy	Hjorth Complexity	62.90323	99.08244	98.97931
Spectral Entropy	Average Energy	Hjorth Mobility	62.90323	99.00867	98.90575
Spectral Entropy	Average Energy	Coastline	67.74194	98.86112	98.77241
Spectral Entropy	Hurst Exponent	Root Mean Square	83.87097	97.80524	97.76552
Spectral Entropy	Hurst Exponent	Min Absolute Value	0	100	99.71494
Spectral Entropy	Hurst Exponent	Max Absolute Value	0	100	99.71494
Spectral Entropy	Hurst Exponent	Mean Absolute Value	87.09677	97.68536	97.65517

Spectral Entropy	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Spectral Entropy	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Spectral Entropy	Hurst Exponent	Coastline	0	100	99.71494
Spectral Entropy	Hurst Exponent	Average Energy	61.29032	98.93951	98.83218
Spectral Entropy	Renyie Entropy	Root Mean Square	82.25807	98.20638	98.16092
Spectral Entropy	Renyie Entropy	Min Absolute Value	0	100	99.71494
Spectral Entropy	Renyie Entropy	Max Absolute Value	0	100	99.71494
Spectral Entropy	Renyie Entropy	Mean Absolute Value	82.25807	98.1695	98.12414
Spectral Entropy	Renyie Entropy	Hjorth Complexity	0	100	99.71494
Spectral Entropy	Renyie Entropy	Hjorth Mobility	0	100	99.71494
Spectral Entropy	Renyie Entropy	Coastline	0	100	99.71494
Spectral Entropy	Renyie Entropy	Average Energy	66.12903	98.8934	98.8
Spectral Entropy	Renyie Entropy	Hurst Exponent	0	100	99.71494
Shannon Entropy	Min Absolute Value	Root Mean Square	83.87097	97.66691	97.62759
Shannon Entropy	Max Absolute Value	Root Mean Square	82.25807	98.3862	98.34023
Shannon Entropy	Max Absolute Value	Min Absolute Value	0	100	99.71494
Shannon Entropy	Mean Absolute Value	Root Mean Square	83.87097	97.98967	97.94943
Shannon Entropy	Mean Absolute Value	Min Absolute Value	87.09677	97.67613	97.64598
Shannon Entropy	Mean Absolute Value	Max Absolute Value	85.48387	98.22483	98.18851
Shannon Entropy	Hjorth Complexity	Root Mean Square	82.25807	98.31243	98.26667
Shannon Entropy	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Shannon Entropy	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Shannon Entropy	Hjorth Complexity	Mean Absolute Value	85.48387	98.20638	98.17012
Shannon Entropy	Hjorth Mobility	Root Mean Square	82.25807	98.31704	98.27126
Shannon Entropy	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Shannon Entropy	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Shannon Entropy	Hjorth Mobility	Mean Absolute Value	85.48387	98.22483	98.18851
Shannon Entropy	Hjorth Mobility	Hjorth Complexity	0	100	99.71494

Shannon Entropy	Coastline	Root Mean Square	82.25807	98.04039	97.9954
Shannon Entropy	Coastline	Min Absolute Value	0	100	99.71494
Shannon Entropy	Coastline	Max Absolute Value	0	100	99.71494
Shannon Entropy	Coastline	Mean Absolute Value	82.25807	98.0865	98.04138
Shannon Entropy	Coastline	Hjorth Complexity	0	100	99.71494
Shannon Entropy	Coastline	Hjorth Mobility	0	100	99.71494
Shannon Entropy	Average Energy	Root Mean Square	72.58065	98.57525	98.50115
Shannon Entropy	Average Energy	Min Absolute Value	58.06452	98.78274	98.66667
Shannon Entropy	Average Energy	Max Absolute Value	64.51613	99.00867	98.91035
Shannon Entropy	Average Energy	Mean Absolute Value	79.03226	98.43231	98.37701
Shannon Entropy	Average Energy	Hjorth Complexity	62.90323	99.064	98.96092
Shannon Entropy	Average Energy	Hjorth Mobility	62.90323	99.06861	98.96552
Shannon Entropy	Average Energy	Coastline	61.29032	98.78274	98.67586
Shannon Entropy	Hurst Exponent	Root Mean Square	85.48387	97.86518	97.82989
Shannon Entropy	Hurst Exponent	Min Absolute Value	0	100	99.71494
Shannon Entropy	Hurst Exponent	Max Absolute Value	0	100	99.71494
Shannon Entropy	Hurst Exponent	Mean Absolute Value	87.09677	97.69919	97.66897
Shannon Entropy	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Shannon Entropy	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Shannon Entropy	Hurst Exponent	Coastline	0	100	99.71494
Shannon Entropy	Hurst Exponent	Average Energy	62.90323	98.90262	98.8
Shannon Entropy	Renyie Entropy	Root Mean Square	83.87097	98.04039	98
Shannon Entropy	Renyie Entropy	Min Absolute Value	0	100	99.71494
Shannon Entropy	Renyie Entropy	Max Absolute Value	0	100	99.71494
Shannon Entropy	Renyie Entropy	Mean Absolute Value	83.87097	98.0035	97.96322
Shannon Entropy	Renyie Entropy	Hjorth Complexity	0	100	99.71494
Shannon Entropy	Renyie Entropy	Hjorth Mobility	0	100	99.71494
Shannon Entropy	Renyie Entropy	Coastline	0	100	99.71494
Shannon Entropy	Renyie Entropy	Average Energy	67.74194	98.8104	98.72184

Shannon Entropy	Renyie Entropy	Hurst Exponent	0	100	99.71494
Shannon Entropy	Spectral Entropy	Root Mean Square	83.87097	98.03578	97.9954
Shannon Entropy	Spectral Entropy	Min Absolute Value	0	100	99.71494
Shannon Entropy	Spectral Entropy	Max Absolute Value	0	100	99.71494
Shannon Entropy	Spectral Entropy	Mean Absolute Value	83.87097	97.98506	97.94483
Shannon Entropy	Spectral Entropy	Hjorth Complexity	0	100	99.71494
Shannon Entropy	Spectral Entropy	Hjorth Mobility	0	100	99.71494
Shannon Entropy	Spectral Entropy	Coastline	0	100	99.71494
Shannon Entropy	Spectral Entropy	Average Energy	66.12903	98.83346	98.74023
Shannon Entropy	Spectral Entropy	Hurst Exponent	0	100	99.71494
Shannon Entropy	Spectral Entropy	Renyie Entropy	0	100	99.71494
Approximate Entropy	Min Absolute Value	Root Mean Square	85.48387	97.82368	97.78851
Approximate Entropy	Max Absolute Value	Root Mean Square	83.87097	98.29399	98.25287
Approximate Entropy	Max Absolute Value	Min Absolute Value	0	100	99.71494
Approximate Entropy	Mean Absolute Value	Root Mean Square	87.09677	98.05883	98.02759
Approximate Entropy	Mean Absolute Value	Min Absolute Value	87.09677	97.80063	97.77012
Approximate Entropy	Mean Absolute Value	Max Absolute Value	85.48387	98.16488	98.12874
Approximate Entropy	Hjorth Complexity	Root Mean Square	82.25807	98.15566	98.11035
Approximate Entropy	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Approximate Entropy	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Approximate Entropy	Hjorth Complexity	Mean Absolute Value	85.48387	98.11417	98.07816
Approximate Entropy	Hjorth Mobility	Root Mean Square	83.87097	98.13722	98.09655
Approximate Entropy	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Approximate Entropy	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Approximate Entropy	Hjorth Mobility	Mean Absolute Value	85.48387	98.0865	98.05058
Approximate Entropy	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Approximate Entropy	Coastline	Root Mean Square	82.25807	98.01273	97.96782
Approximate Entropy	Coastline	Min Absolute Value	0	100	99.71494

Approximate Entropy	Coastline	Max Absolute Value	0	100	99.71494
Approximate Entropy	Coastline	Mean Absolute Value	85.48387	98.00812	97.97241
Approximate Entropy	Coastline	Hjorth Complexity	0	100	99.71494
Approximate Entropy	Coastline	Hjorth Mobility	0	100	99.71494
Approximate Entropy	Average Energy	Root Mean Square	70.96774	98.73202	98.65287
Approximate Entropy	Average Energy	Min Absolute Value	64.51613	98.75507	98.65747
Approximate Entropy	Average Energy	Max Absolute Value	64.51613	99.04556	98.94713
Approximate Entropy	Average Energy	Mean Absolute Value	75.80645	98.60291	98.53793
Approximate Entropy	Average Energy	Hjorth Complexity	64.51613	99.02711	98.92874
Approximate Entropy	Average Energy	Hjorth Mobility	62.90323	98.97178	98.86897
Approximate Entropy	Average Energy	Coastline	64.51613	98.86112	98.76322
Approximate Entropy	Hurst Exponent	Root Mean Square	85.48387	97.82368	97.78851
Approximate Entropy	Hurst Exponent	Min Absolute Value	0	100	99.71494
Approximate Entropy	Hurst Exponent	Max Absolute Value	0	100	99.71494
Approximate Entropy	Hurst Exponent	Mean Absolute Value	85.48387	97.78218	97.74713
Approximate Entropy	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Approximate Entropy	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Approximate Entropy	Hurst Exponent	Coastline	0	100	99.71494
Approximate Entropy	Hurst Exponent	Average Energy	62.90323	98.91645	98.81379
Approximate Entropy	Renyie Entropy	Root Mean Square	82.25807	98.13261	98.08736
Approximate Entropy	Renyie Entropy	Min Absolute Value	0	100	99.71494
Approximate Entropy	Renyie Entropy	Max Absolute Value	0	100	99.71494
Approximate Entropy	Renyie Entropy	Mean Absolute Value	85.48387	98.15105	98.11494
Approximate Entropy	Renyie Entropy	Hjorth Complexity	0	100	99.71494
Approximate Entropy	Renyie Entropy	Hjorth Mobility	0	100	99.71494

Approximate Entropy	Renyie Entropy	Coastline	0	100	99.71494
Approximate Entropy	Renyie Entropy	Average Energy	69.35484	98.84729	98.76322
Approximate Entropy	Renyie Entropy	Hurst Exponent	0	100	99.71494
Approximate Entropy	Spectral Entropy	Root Mean Square	85.48387	98.06345	98.02759
Approximate Entropy	Spectral Entropy	Min Absolute Value	0	100	99.71494
Approximate Entropy	Spectral Entropy	Max Absolute Value	0	100	99.71494
Approximate Entropy	Spectral Entropy	Mean Absolute Value	87.09677	98.04961	98.01839
Approximate Entropy	Spectral Entropy	Hjorth Complexity	0	100	99.71494
Approximate Entropy	Spectral Entropy	Hjorth Mobility	0	100	99.71494
Approximate Entropy	Spectral Entropy	Coastline	0	100	99.71494
Approximate Entropy	Spectral Entropy	Average Energy	67.74194	98.88418	98.7954
Approximate Entropy	Spectral Entropy	Hurst Exponent	0	100	99.71494
Approximate Entropy	Spectral Entropy	Renyie Entropy	0	100	99.71494
Approximate Entropy	Shannon Entropy	Root Mean Square	83.87097	98.04961	98.0092
Approximate Entropy	Shannon Entropy	Min Absolute Value	0	100	99.71494
Approximate Entropy	Shannon Entropy	Max Absolute Value	0	100	99.71494
Approximate Entropy	Shannon Entropy	Mean Absolute Value	87.09677	98.04039	98.0092
Approximate Entropy	Shannon Entropy	Hjorth Complexity	0	100	99.71494
Approximate Entropy	Shannon Entropy	Hjorth Mobility	0	100	99.71494
Approximate Entropy	Shannon Entropy	Coastline	0	100	99.71494
Approximate Entropy	Shannon Entropy	Average Energy	67.74194	98.75046	98.66207
Approximate Entropy	Shannon Entropy	Hurst Exponent	0	100	99.71494
Approximate Entropy	Shannon Entropy	Renyie Entropy	0	100	99.71494
Approximate Entropy	Shannon Entropy	Spectral Entropy	0	100	99.71494
Permutation Entropy	Min Absolute Value	Root Mean Square	83.87097	97.90207	97.86207

Permutation Entropy	Max Absolute Value	Root Mean Square	82.25807	98.23866	98.1931
Permutation Entropy	Max Absolute Value	Min Absolute Value	0	100	99.71494
Permutation Entropy	Mean Absolute Value	Root Mean Square	82.25807	98.02195	97.97701
Permutation Entropy	Mean Absolute Value	Min Absolute Value	87.09677	97.89746	97.86667
Permutation Entropy	Mean Absolute Value	Max Absolute Value	83.87097	98.2156	98.17471
Permutation Entropy	Hjorth Complexity	Root Mean Square	82.25807	98.34471	98.29885
Permutation Entropy	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Permutation Entropy	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Permutation Entropy	Hjorth Complexity	Mean Absolute Value	85.48387	98.16488	98.12874
Permutation Entropy	Hjorth Mobility	Root Mean Square	80.64516	98.3401	98.28966
Permutation Entropy	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Permutation Entropy	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Permutation Entropy	Hjorth Mobility	Mean Absolute Value	83.87097	98.14644	98.10575
Permutation Entropy	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Permutation Entropy	Coastline	Root Mean Square	82.25807	98.06806	98.02299
Permutation Entropy	Coastline	Min Absolute Value	0	100	99.71494
Permutation Entropy	Coastline	Max Absolute Value	0	100	99.71494
Permutation Entropy	Coastline	Mean Absolute Value	83.87097	98.01273	97.97241
Permutation Entropy	Coastline	Hjorth Complexity	0	100	99.71494
Permutation Entropy	Coastline	Hjorth Mobility	0	100	99.71494
Permutation Entropy	Average Energy	Root Mean Square	67.74194	98.83807	98.74943
Permutation Entropy	Average Energy	Min Absolute Value	62.90323	98.85651	98.75402
Permutation Entropy	Average Energy	Max Absolute Value	61.29032	99.21155	99.10345
Permutation Entropy	Average Energy	Mean Absolute Value	70.96774	98.7689	98.68966
Permutation Entropy	Average Energy	Hjorth Complexity	64.51613	99.05939	98.96092

Permutation Entropy	Average Energy	Hjorth Mobility	62.90323	99.02711	98.92414
Permutation Entropy	Average Energy	Coastline	62.90323	98.91645	98.81379
Permutation Entropy	Hurst Exponent	Root Mean Square	83.87097	97.93434	97.89425
Permutation Entropy	Hurst Exponent	Min Absolute Value	0	100	99.71494
Permutation Entropy	Hurst Exponent	Max Absolute Value	0	100	99.71494
Permutation Entropy	Hurst Exponent	Mean Absolute Value	85.48387	97.82368	97.78851
Permutation Entropy	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Permutation Entropy	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Permutation Entropy	Hurst Exponent	Coastline	0	100	99.71494
Permutation Entropy	Hurst Exponent	Average Energy	62.90323	98.92106	98.81839
Permutation Entropy	Renyie Entropy	Root Mean Square	80.64516	98.13722	98.08736
Permutation Entropy	Renyie Entropy	Min Absolute Value	0	100	99.71494
Permutation Entropy	Renyie Entropy	Max Absolute Value	0	100	99.71494
Permutation Entropy	Renyie Entropy	Mean Absolute Value	82.25807	98.08189	98.03678
Permutation Entropy	Renyie Entropy	Hjorth Complexity	0	100	99.71494
Permutation Entropy	Renyie Entropy	Hjorth Mobility	0	100	99.71494
Permutation Entropy	Renyie Entropy	Coastline	0	100	99.71494
Permutation Entropy	Renyie Entropy	Average Energy	64.51613	98.90262	98.8046
Permutation Entropy	Renyie Entropy	Hurst Exponent	0	100	99.71494
Permutation Entropy	Spectral Entropy	Root Mean Square	83.87097	97.96662	97.92644
Permutation Entropy	Spectral Entropy	Min Absolute Value	0	100	99.71494
Permutation Entropy	Spectral Entropy	Max Absolute Value	0	100	99.71494
Permutation Entropy	Spectral Entropy	Mean Absolute Value	87.09677	97.99428	97.96322
Permutation Entropy	Spectral Entropy	Hjorth Complexity	0	100	99.71494
Permutation Entropy	Spectral Entropy	Hjorth Mobility	0	100	99.71494

Permutation Entropy	Spectral Entropy	Coastline	0	100	99.71494
Permutation Entropy	Spectral Entropy	Average Energy	64.51613	98.91645	98.81839
Permutation Entropy	Spectral Entropy	Hurst Exponent	0	100	99.71494
Permutation Entropy	Spectral Entropy	Renyie Entropy	0	100	99.71494
Permutation Entropy	Shannon Entropy	Root Mean Square	83.87097	98.05422	98.01379
Permutation Entropy	Shannon Entropy	Min Absolute Value	0	100	99.71494
Permutation Entropy	Shannon Entropy	Max Absolute Value	0	100	99.71494
Permutation Entropy	Shannon Entropy	Mean Absolute Value	83.87097	98.01273	97.97241
Permutation Entropy	Shannon Entropy	Hjorth Complexity	0	100	99.71494
Permutation Entropy	Shannon Entropy	Hjorth Mobility	0	100	99.71494
Permutation Entropy	Shannon Entropy	Coastline	0	100	99.71494
Permutation Entropy	Shannon Entropy	Average Energy	64.51613	98.92106	98.82299
Permutation Entropy	Shannon Entropy	Hurst Exponent	0	100	99.71494
Permutation Entropy	Shannon Entropy	Renyie Entropy	0	100	99.71494
Permutation Entropy	Shannon Entropy	Spectral Entropy	0	100	99.71494
Permutation Entropy	Approximate Entropy	Root Mean Square	82.25807	98.05422	98.0092
Permutation Entropy	Approximate Entropy	Min Absolute Value	0	100	99.71494
Permutation Entropy	Approximate Entropy	Max Absolute Value	0	100	99.71494
Permutation Entropy	Approximate Entropy	Mean Absolute Value	83.87097	98.08189	98.04138
Permutation Entropy	Approximate Entropy	Hjorth Complexity	0	100	99.71494
Permutation Entropy	Approximate Entropy	Hjorth Mobility	0	100	99.71494
Permutation Entropy	Approximate Entropy	Coastline	0	100	99.71494
Permutation Entropy	Approximate Entropy	Average Energy	64.51613	98.88879	98.79081
Permutation Entropy	Approximate Entropy	Hurst Exponent	0	100	99.71494
Permutation Entropy	Approximate Entropy	Renyie Entropy	0	100	99.71494

Permutation Entropy	Approximate Entropy	Spectral Entropy	0	100	99.71494
Permutation Entropy	Approximate Entropy	Shannon Entropy	0	100	99.71494
Variance	Min Absolute Value	Root Mean Square	70.96774	98.61214	98.53333
Variance	Max Absolute Value	Root Mean Square	66.12903	99.00867	98.91494
Variance	Max Absolute Value	Min Absolute Value	59.67742	98.86112	98.74943
Variance	Mean Absolute Value	Root Mean Square	74.19355	98.73663	98.66667
Variance	Mean Absolute Value	Min Absolute Value	74.19355	98.5522	98.48276
Variance	Mean Absolute Value	Max Absolute Value	75.80645	98.8104	98.74483
Variance	Hjorth Complexity	Root Mean Square	67.74194	98.97178	98.88276
Variance	Hjorth Complexity	Min Absolute Value	61.29032	99.04556	98.93793
Variance	Hjorth Complexity	Max Absolute Value	62.90323	99.19771	99.09425
Variance	Hjorth Complexity	Mean Absolute Value	69.35484	98.95334	98.86897
Variance	Hjorth Mobility	Root Mean Square	66.12903	99.01789	98.92414
Variance	Hjorth Mobility	Min Absolute Value	61.29032	99.00406	98.89655
Variance	Hjorth Mobility	Max Absolute Value	61.29032	99.14699	99.03908
Variance	Hjorth Mobility	Mean Absolute Value	67.74194	98.95795	98.86897
Variance	Hjorth Mobility	Hjorth Complexity	62.90323	99.07783	98.97471
Variance	Coastline	Root Mean Square	62.90323	98.84729	98.74483
Variance	Coastline	Min Absolute Value	61.29032	98.79657	98.68966
Variance	Coastline	Max Absolute Value	58.06452	99.07783	98.96092
Variance	Coastline	Mean Absolute Value	70.96774	98.75968	98.68046
Variance	Coastline	Hjorth Complexity	62.90323	99.09166	98.98851
Variance	Coastline	Hjorth Mobility	64.51613	99.04556	98.94713
Variance	Average Energy	Root Mean Square	69.35484	98.90262	98.81839
Variance	Average Energy	Min Absolute Value	64.51613	98.78274	98.68506

Variance	Average Energy	Max Absolute Value	64.51613	99.14238	99.04368
Variance	Average Energy	Mean Absolute Value	72.58065	98.82424	98.74943
Variance	Average Energy	Hjorth Complexity	62.90323	99.08244	98.97931
Variance	Average Energy	Hjorth Mobility	62.90323	99.02711	98.92414
Variance	Average Energy	Coastline	62.90323	98.8519	98.74943
Variance	Hurst Exponent	Root Mean Square	72.58065	98.71819	98.64368
Variance	Hurst Exponent	Min Absolute Value	62.90323	98.87034	98.76782
Variance	Hurst Exponent	Max Absolute Value	61.29032	99.16544	99.05747
Variance	Hurst Exponent	Mean Absolute Value	74.19355	98.58447	98.51494
Variance	Hurst Exponent	Hjorth Complexity	74.19355	98.47842	98.4092
Variance	Hurst Exponent	Hjorth Mobility	82.25807	97.86518	97.82069
Variance	Hurst Exponent	Coastline	62.90323	98.95334	98.85058
Variance	Hurst Exponent	Average Energy	62.90323	98.91645	98.81379
Variance	Renyie Entropy	Root Mean Square	72.58065	98.6398	98.56552
Variance	Renyie Entropy	Min Absolute Value	56.45161	98.87495	98.75402
Variance	Renyie Entropy	Max Absolute Value	62.90323	99.05939	98.95632
Variance	Renyie Entropy	Mean Absolute Value	75.80645	98.55681	98.49195
Variance	Renyie Entropy	Hjorth Complexity	62.90323	99.09166	98.98851
Variance	Renyie Entropy	Hjorth Mobility	62.90323	99.08244	98.97931
Variance	Renyie Entropy	Coastline	58.06452	98.84268	98.72644
Variance	Renyie Entropy	Average Energy	66.12903	98.8104	98.71724
Variance	Renyie Entropy	Hurst Exponent	62.90323	98.93028	98.82759
Variance	Spectral Entropy	Root Mean Square	69.35484	98.87957	98.7954
Variance	Spectral Entropy	Min Absolute Value	62.90323	98.8104	98.70805
Variance	Spectral Entropy	Max Absolute Value	62.90323	99.07322	98.97012
Variance	Spectral Entropy	Mean Absolute Value	72.58065	98.78735	98.71264
Variance	Spectral Entropy	Hjorth Complexity	62.90323	99.08705	98.98391
Variance	Spectral Entropy	Hjorth Mobility	64.51613	99.05017	98.95172
Variance	Spectral Entropy	Coastline	69.35484	98.87495	98.79081
Variance	Spectral Entropy	Average Energy	69.35484	98.86112	98.77701
Variance	Spectral Entropy	Hurst Exponent	61.29032	98.95795	98.85058

Variance	Spectral Entropy	Renyie Entropy	66.12903	98.88879	98.7954
Variance	Shannon Entropy	Root Mean Square	72.58065	98.59369	98.51954
Variance	Shannon Entropy	Min Absolute Value	58.06452	98.80118	98.68506
Variance	Shannon Entropy	Max Absolute Value	61.29032	99.00406	98.89655
Variance	Shannon Entropy	Mean Absolute Value	77.41936	98.4692	98.4092
Variance	Shannon Entropy	Hjorth Complexity	62.90323	99.08705	98.98391
Variance	Shannon Entropy	Hjorth Mobility	62.90323	99.09627	98.9931
Variance	Shannon Entropy	Coastline	61.29032	98.79657	98.68966
Variance	Shannon Entropy	Average Energy	67.74194	98.82424	98.73563
Variance	Shannon Entropy	Hurst Exponent	62.90323	98.93028	98.82759
Variance	Shannon Entropy	Renyie Entropy	67.74194	98.84268	98.75402
Variance	Shannon Entropy	Spectral Entropy	66.12903	98.84729	98.75402
Variance	Approximate Entropy	Root Mean Square	70.96774	98.74124	98.66207
Variance	Approximate Entropy	Min Absolute Value	64.51613	98.74585	98.64828
Variance	Approximate Entropy	Max Absolute Value	64.51613	99.08244	98.98391
Variance	Approximate Entropy	Mean Absolute Value	75.80645	98.60753	98.54253
Variance	Approximate Entropy	Hjorth Complexity	64.51613	99.05939	98.96092
Variance	Approximate Entropy	Hjorth Mobility	62.90323	99.0225	98.91954
Variance	Approximate Entropy	Coastline	62.90323	98.84729	98.74483
Variance	Approximate Entropy	Average Energy	69.35484	98.84268	98.75862
Variance	Approximate Entropy	Hurst Exponent	62.90323	98.93951	98.83678
Variance	Approximate Entropy	Renyie Entropy	69.35484	98.84729	98.76322
Variance	Approximate Entropy	Spectral Entropy	67.74194	98.88879	98.8
Variance	Approximate Entropy	Shannon Entropy	67.74194	98.75507	98.66667
Variance	Permutation Entropy	Root Mean Square	69.35484	98.84729	98.76322
Variance	Permutation Entropy	Min Absolute Value	62.90323	98.87495	98.77241
Variance	Permutation Entropy	Max Absolute Value	61.29032	99.20232	99.09425
Variance	Permutation Entropy	Mean Absolute Value	72.58065	98.77352	98.69885

Variance	Permutation Entropy	Hjorth Complexity	64.51613	99.07322	98.97471
Variance	Permutation Entropy	Hjorth Mobility	62.90323	99.05017	98.94713
Variance	Permutation Entropy	Coastline	62.90323	98.90262	98.8
Variance	Permutation Entropy	Average Energy	64.51613	98.90723	98.8092
Variance	Permutation Entropy	Hurst Exponent	62.90323	98.94412	98.84138
Variance	Permutation Entropy	Renyie Entropy	64.51613	98.8934	98.7954
Variance	Permutation Entropy	Spectral Entropy	64.51613	98.91645	98.81839
Variance	Permutation Entropy	Shannon Entropy	64.51613	98.92106	98.82299
Variance	Permutation Entropy	Approximate Entropy	66.12903	98.88418	98.79081
Skew	Min Absolute Value	Root Mean Square	85.48387	97.88823	97.85287
Skew	Max Absolute Value	Root Mean Square	82.25807	98.26632	98.22069
Skew	Max Absolute Value	Min Absolute Value	0	100	99.71494
Skew	Mean Absolute Value	Root Mean Square	87.09677	98.11878	98.08736
Skew	Mean Absolute Value	Min Absolute Value	87.09677	97.79141	97.76092
Skew	Mean Absolute Value	Max Absolute Value	87.09677	98.128	98.09655
Skew	Hjorth Complexity	Root Mean Square	80.64516	98.43692	98.38621
Skew	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Skew	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Skew	Hjorth Complexity	Mean Absolute Value	83.87097	98.24327	98.2023
Skew	Hjorth Mobility	Root Mean Square	83.87097	98.42309	98.38161
Skew	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Skew	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Skew	Hjorth Mobility	Mean Absolute Value	85.48387	98.32626	98.28966
Skew	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Skew	Coastline	Root Mean Square	80.64516	98.24327	98.1931

Skew	Coastline	Min Absolute Value	0	100	99.71494
Skew	Coastline	Max Absolute Value	0	100	99.71494
Skew	Coastline	Mean Absolute Value	85.48387	98.11417	98.07816
Skew	Coastline	Hjorth Complexity	0	100	99.71494
Skew	Coastline	Hjorth Mobility	0	100	99.71494
Skew	Average Energy	Root Mean Square	74.19355	98.81962	98.74943
Skew	Average Energy	Min Absolute Value	61.29032	98.79657	98.68966
Skew	Average Energy	Max Absolute Value	64.51613	98.99484	98.89655
Skew	Average Energy	Mean Absolute Value	77.41936	98.62597	98.56552
Skew	Average Energy	Hjorth Complexity	66.12903	99.01789	98.92414
Skew	Average Energy	Hjorth Mobility	64.51613	99.04094	98.94253
Skew	Average Energy	Coastline	56.45161	98.95334	98.83218
Skew	Hurst Exponent	Root Mean Square	85.48387	97.99428	97.95862
Skew	Hurst Exponent	Min Absolute Value	0	100	99.71494
Skew	Hurst Exponent	Max Absolute Value	0	100	99.71494
Skew	Hurst Exponent	Mean Absolute Value	87.09677	97.8744	97.84368
Skew	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Skew	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Skew	Hurst Exponent	Coastline	0	100	99.71494
Skew	Hurst Exponent	Average Energy	62.90323	98.86112	98.75862
Skew	Renyie Entropy	Root Mean Square	82.25807	98.26632	98.22069
Skew	Renyie Entropy	Min Absolute Value	0	100	99.71494
Skew	Renyie Entropy	Max Absolute Value	0	100	99.71494
Skew	Renyie Entropy	Mean Absolute Value	87.09677	98.13261	98.10115
Skew	Renyie Entropy	Hjorth Complexity	0	100	99.71494
Skew	Renyie Entropy	Hjorth Mobility	0	100	99.71494
Skew	Renyie Entropy	Coastline	0	100	99.71494
Skew	Renyie Entropy	Average Energy	67.74194	98.9349	98.84598
Skew	Renyie Entropy	Hurst Exponent	0	100	99.71494

Skew	Spectral Entropy	Root Mean Square	83.87097	98.22944	98.18851
Skew	Spectral Entropy	Min Absolute Value	0	100	99.71494
Skew	Spectral Entropy	Max Absolute Value	0	100	99.71494
Skew	Spectral Entropy	Mean Absolute Value	87.09677	98.06345	98.03218
Skew	Spectral Entropy	Hjorth Complexity	0	100	99.71494
Skew	Spectral Entropy	Hjorth Mobility	0	100	99.71494
Skew	Spectral Entropy	Coastline	0	100	99.71494
Skew	Spectral Entropy	Average Energy	66.12903	99.00867	98.91494
Skew	Spectral Entropy	Hurst Exponent	0	100	99.71494
Skew	Spectral Entropy	Renyie Entropy	0	100	99.71494
Skew	Shannon Entropy	Root Mean Square	82.25807	98.2571	98.21149
Skew	Shannon Entropy	Min Absolute Value	0	100	99.71494
Skew	Shannon Entropy	Max Absolute Value	0	100	99.71494
Skew	Shannon Entropy	Mean Absolute Value	87.09677	98.10033	98.06897
Skew	Shannon Entropy	Hjorth Complexity	0	100	99.71494
Skew	Shannon Entropy	Hjorth Mobility	0	100	99.71494
Skew	Shannon Entropy	Coastline	0	100	99.71494
Skew	Shannon Entropy	Average Energy	67.74194	98.90262	98.81379
Skew	Shannon Entropy	Hurst Exponent	0	100	99.71494
Skew	Shannon Entropy	Renyie Entropy	0	100	99.71494
Skew	Shannon Entropy	Spectral Entropy	0	100	99.71494
Skew	Approximate Entropy	Root Mean Square	82.25807	98.21099	98.16552
Skew	Approximate Entropy	Min Absolute Value	0	100	99.71494
Skew	Approximate Entropy	Max Absolute Value	0	100	99.71494
Skew	Approximate Entropy	Mean Absolute Value	87.09677	98.11417	98.08276
Skew	Approximate Entropy	Hjorth Complexity	0	100	99.71494
Skew	Approximate Entropy	Hjorth Mobility	0	100	99.71494
Skew	Approximate Entropy	Coastline	0	100	99.71494
Skew	Approximate Entropy	Average Energy	67.74194	98.9349	98.84598
Skew	Approximate Entropy	Hurst Exponent	0	100	99.71494

Skew	Approximate Entropy	Renyie Entropy	0	100	99.71494
Skew	Approximate Entropy	Spectral Entropy	0	100	99.71494
Skew	Approximate Entropy	Shannon Entropy	0	100	99.71494
Skew	Permutation Entropy	Root Mean Square	83.87097	98.23405	98.1931
Skew	Permutation Entropy	Min Absolute Value	0	100	99.71494
Skew	Permutation Entropy	Max Absolute Value	0	100	99.71494
Skew	Permutation Entropy	Mean Absolute Value	87.09677	98.12339	98.09195
Skew	Permutation Entropy	Hjorth Complexity	0	100	99.71494
Skew	Permutation Entropy	Hjorth Mobility	0	100	99.71494
Skew	Permutation Entropy	Coastline	0	100	99.71494
Skew	Permutation Entropy	Average Energy	67.74194	98.92106	98.83218
Skew	Permutation Entropy	Hurst Exponent	0	100	99.71494
Skew	Permutation Entropy	Renyie Entropy	0	100	99.71494
Skew	Permutation Entropy	Spectral Entropy	0	100	99.71494
Skew	Permutation Entropy	Shannon Entropy	0	100	99.71494
Skew	Permutation Entropy	Approximate Entropy	0	100	99.71494
Skew	Variance	Root Mean Square	74.19355	98.81962	98.74943
Skew	Variance	Min Absolute Value	62.90323	98.80579	98.70345
Skew	Variance	Max Absolute Value	64.51613	99.01328	98.91494
Skew	Variance	Mean Absolute Value	77.41936	98.63058	98.57012
Skew	Variance	Hjorth Complexity	66.12903	99.0225	98.92874
Skew	Variance	Hjorth Mobility	64.51613	99.05939	98.96092
Skew	Variance	Coastline	58.06452	98.97178	98.85517
Skew	Variance	Average Energy	67.74194	98.94412	98.85517
Skew	Variance	Hurst Exponent	66.12903	98.82424	98.73103
Skew	Variance	Renyie Entropy	67.74194	98.9349	98.84598
Skew	Variance	Spectral Entropy	66.12903	99.01328	98.91954
Skew	Variance	Shannon Entropy	67.74194	98.91645	98.82759

Skew	Variance	Approximate Entropy	67.74194	98.93951	98.85058
Skew	Variance	Permutation Entropy	67.74194	98.9349	98.84598
Kurtosis	Min Absolute Value	Root Mean Square	85.48387	98.16027	98.12414
Kurtosis	Max Absolute Value	Root Mean Square	85.48387	98.81962	98.78161
Kurtosis	Max Absolute Value	Min Absolute Value	0	100	99.71494
Kurtosis	Mean Absolute Value	Root Mean Square	85.48387	98.39082	98.35402
Kurtosis	Mean Absolute Value	Min Absolute Value	87.09677	97.76374	97.73333
Kurtosis	Mean Absolute Value	Max Absolute Value	85.48387	98.6398	98.6023
Kurtosis	Hjorth Complexity	Root Mean Square	82.25807	98.54297	98.49655
Kurtosis	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Kurtosis	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Kurtosis	Hjorth Complexity	Mean Absolute Value	85.48387	98.19255	98.15632
Kurtosis	Hjorth Mobility	Root Mean Square	83.87097	98.56603	98.52414
Kurtosis	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Kurtosis	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Kurtosis	Hjorth Mobility	Mean Absolute Value	85.48387	98.27554	98.23908
Kurtosis	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Kurtosis	Coastline	Root Mean Square	82.25807	98.58447	98.53793
Kurtosis	Coastline	Min Absolute Value	0	100	99.71494
Kurtosis	Coastline	Max Absolute Value	0	100	99.71494
Kurtosis	Coastline	Mean Absolute Value	85.48387	98.27554	98.23908
Kurtosis	Coastline	Hjorth Complexity	0	100	99.71494
Kurtosis	Coastline	Hjorth Mobility	0	100	99.71494
Kurtosis	Average Energy	Root Mean Square	70.96774	98.88418	98.8046
Kurtosis	Average Energy	Min Absolute Value	62.90323	99.16544	99.06207

Kurtosis	Average Energy	Max Absolute Value	66.12903	99.42364	99.32874
Kurtosis	Average Energy	Mean Absolute Value	77.41936	98.71357	98.65287
Kurtosis	Average Energy	Hjorth Complexity	62.90323	99.13777	99.03448
Kurtosis	Average Energy	Hjorth Mobility	62.90323	99.30376	99.2
Kurtosis	Average Energy	Coastline	66.12903	99.20232	99.10805
Kurtosis	Hurst Exponent	Root Mean Square	85.48387	98.128	98.09195
Kurtosis	Hurst Exponent	Min Absolute Value	0	100	99.71494
Kurtosis	Hurst Exponent	Max Absolute Value	0	100	99.71494
Kurtosis	Hurst Exponent	Mean Absolute Value	87.09677	97.80524	97.77471
Kurtosis	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Kurtosis	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Kurtosis	Hurst Exponent	Coastline	0	100	99.71494
Kurtosis	Hurst Exponent	Average Energy	62.90323	99.30376	99.2
Kurtosis	Renyie Entropy	Root Mean Square	82.25807	98.37698	98.33103
Kurtosis	Renyie Entropy	Min Absolute Value	0	100	99.71494
Kurtosis	Renyie Entropy	Max Absolute Value	0	100	99.71494
Kurtosis	Renyie Entropy	Mean Absolute Value	82.25807	98.26171	98.21609
Kurtosis	Renyie Entropy	Hjorth Complexity	0	100	99.71494
Kurtosis	Renyie Entropy	Hjorth Mobility	0	100	99.71494
Kurtosis	Renyie Entropy	Coastline	0	100	99.71494
Kurtosis	Renyie Entropy	Average Energy	66.12903	98.87495	98.78161
Kurtosis	Renyie Entropy	Hurst Exponent	0	100	99.71494
Kurtosis	Spectral Entropy	Root Mean Square	83.87097	98.61214	98.57012
Kurtosis	Spectral Entropy	Min Absolute Value	0	100	99.71494
Kurtosis	Spectral Entropy	Max Absolute Value	0	100	99.71494
Kurtosis	Spectral Entropy	Mean Absolute Value	87.09677	98.31704	98.28506
Kurtosis	Spectral Entropy	Hjorth Complexity	0	100	99.71494
Kurtosis	Spectral Entropy	Hjorth Mobility	0	100	99.71494
Kurtosis	Spectral Entropy	Coastline	0	100	99.71494
Kurtosis	Spectral Entropy	Average Energy	69.35484	99.22077	99.13563
Kurtosis	Spectral Entropy	Hurst Exponent	0	100	99.71494

Kurtosis	Spectral Entropy	Renyie Entropy	0	100	99.71494
Kurtosis	Shannon Entropy	Root Mean Square	83.87097	98.37698	98.33563
Kurtosis	Shannon Entropy	Min Absolute Value	0	100	99.71494
Kurtosis	Shannon Entropy	Max Absolute Value	0	100	99.71494
Kurtosis	Shannon Entropy	Mean Absolute Value	83.87097	98.2156	98.17471
Kurtosis	Shannon Entropy	Hjorth Complexity	0	100	99.71494
Kurtosis	Shannon Entropy	Hjorth Mobility	0	100	99.71494
Kurtosis	Shannon Entropy	Coastline	0	100	99.71494
Kurtosis	Shannon Entropy	Average Energy	67.74194	98.84268	98.75402
Kurtosis	Shannon Entropy	Hurst Exponent	0	100	99.71494
Kurtosis	Shannon Entropy	Renyie Entropy	0	100	99.71494
Kurtosis	Shannon Entropy	Spectral Entropy	0	100	99.71494
Kurtosis	Approximate Entropy	Root Mean Square	83.87097	98.45076	98.4092
Kurtosis	Approximate Entropy	Min Absolute Value	0	100	99.71494
Kurtosis	Approximate Entropy	Max Absolute Value	0	100	99.71494
Kurtosis	Approximate Entropy	Mean Absolute Value	85.48387	98.04039	98.0046
Kurtosis	Approximate Entropy	Hjorth Complexity	0	100	99.71494
Kurtosis	Approximate Entropy	Hjorth Mobility	0	100	99.71494
Kurtosis	Approximate Entropy	Coastline	0	100	99.71494
Kurtosis	Approximate Entropy	Average Energy	69.35484	98.84729	98.76322
Kurtosis	Approximate Entropy	Hurst Exponent	0	100	99.71494
Kurtosis	Approximate Entropy	Renyie Entropy	0	100	99.71494
Kurtosis	Approximate Entropy	Spectral Entropy	0	100	99.71494
Kurtosis	Approximate Entropy	Shannon Entropy	0	100	99.71494
Kurtosis	Permutation Entropy	Root Mean Square	83.87097	98.2571	98.21609
Kurtosis	Permutation Entropy	Min Absolute Value	0	100	99.71494
Kurtosis	Permutation Entropy	Max Absolute Value	0	100	99.71494
Kurtosis	Permutation Entropy	Mean Absolute Value	87.09677	98.0035	97.97241

Kurtosis	Permutation Entropy	Hjorth Complexity	0	100	99.71494
Kurtosis	Permutation Entropy	Hjorth Mobility	0	100	99.71494
Kurtosis	Permutation Entropy	Coastline	0	100	99.71494
Kurtosis	Permutation Entropy	Average Energy	66.12903	99.11011	99.01609
Kurtosis	Permutation Entropy	Hurst Exponent	0	100	99.71494
Kurtosis	Permutation Entropy	Renyie Entropy	0	100	99.71494
Kurtosis	Permutation Entropy	Spectral Entropy	0	100	99.71494
Kurtosis	Permutation Entropy	Shannon Entropy	0	100	99.71494
Kurtosis	Permutation Entropy	Approximate Entropy	0	100	99.71494
Kurtosis	Variance	Root Mean Square	70.96774	98.89801	98.81839
Kurtosis	Variance	Min Absolute Value	64.51613	99.17927	99.08046
Kurtosis	Variance	Max Absolute Value	66.12903	99.43287	99.33793
Kurtosis	Variance	Mean Absolute Value	77.41936	98.70435	98.64368
Kurtosis	Variance	Hjorth Complexity	64.51613	99.17927	99.08046
Kurtosis	Variance	Hjorth Mobility	62.90323	99.3176	99.21379
Kurtosis	Variance	Coastline	64.51613	99.21155	99.11264
Kurtosis	Variance	Average Energy	69.35484	98.9349	98.85058
Kurtosis	Variance	Hurst Exponent	62.90323	99.26227	99.15862
Kurtosis	Variance	Renyie Entropy	66.12903	98.87034	98.77701
Kurtosis	Variance	Spectral Entropy	69.35484	99.2346	99.14943
Kurtosis	Variance	Shannon Entropy	67.74194	98.86112	98.77241
Kurtosis	Variance	Approximate Entropy	69.35484	98.84729	98.76322
Kurtosis	Variance	Permutation Entropy	66.12903	99.11933	99.02529
Kurtosis	Skew	Root Mean Square	82.25807	98.53375	98.48736
Kurtosis	Skew	Min Absolute Value	0	100	99.71494
Kurtosis	Skew	Max Absolute Value	0	100	99.71494
Kurtosis	Skew	Mean Absolute Value	87.09677	98.18794	98.15632
Kurtosis	Skew	Hjorth Complexity	0	100	99.71494

Kurtosis	Skew	Hjorth Mobility	0	100	99.71494
Kurtosis	Skew	Coastline	0	100	99.71494
Kurtosis	Skew	Average Energy	67.74194	99.03633	98.94713
Kurtosis	Skew	Hurst Exponent	0	100	99.71494
Kurtosis	Skew	Renyie Entropy	0	100	99.71494
Kurtosis	Skew	Spectral Entropy	0	100	99.71494
Kurtosis	Skew	Shannon Entropy	0	100	99.71494
Kurtosis	Skew	Approximate Entropy	0	100	99.71494
Kurtosis	Skew	Permutation Entropy	0	100	99.71494
Kurtosis	Skew	Variance	67.74194	99.04556	98.95632
Modified Hurst Exponent	Min Absolute Value	Root Mean Square	83.87097	96.43121	96.3954
Modified Hurst Exponent	Max Absolute Value	Root Mean Square	80.64516	98.8104	98.75862
Modified Hurst Exponent	Max Absolute Value	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Mean Absolute Value	Root Mean Square	87.09677	97.04445	97.01609
Modified Hurst Exponent	Mean Absolute Value	Min Absolute Value	88.70968	96.54648	96.52414
Modified Hurst Exponent	Mean Absolute Value	Max Absolute Value	82.25807	98.45076	98.4046
Modified Hurst Exponent	Hjorth Complexity	Root Mean Square	85.48387	97.84213	97.8069
Modified Hurst Exponent	Hjorth Complexity	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Hjorth Complexity	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Hjorth Complexity	Mean Absolute Value	85.48387	97.65769	97.62299
Modified Hurst Exponent	Hjorth Mobility	Root Mean Square	85.48387	97.82829	97.7931
Modified Hurst Exponent	Hjorth Mobility	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Hjorth Mobility	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Hjorth Mobility	Mean Absolute Value	87.09677	97.69458	97.66437
Modified Hurst Exponent	Hjorth Mobility	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Coastline	Root Mean Square	85.48387	97.2381	97.2046
Modified Hurst Exponent	Coastline	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Coastline	Max Absolute Value	0	100	99.71494

Modified Hurst Exponent	Coastline	Mean Absolute Value	87.09677	96.92918	96.90115
Modified Hurst Exponent	Coastline	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Coastline	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Average Energy	Root Mean Square	74.19355	98.37237	98.30345
Modified Hurst Exponent	Average Energy	Min Absolute Value	56.45161	98.23866	98.11954
Modified Hurst Exponent	Average Energy	Max Absolute Value	62.90323	99.28532	99.18161
Modified Hurst Exponent	Average Energy	Mean Absolute Value	79.03226	98.28477	98.22989
Modified Hurst Exponent	Average Energy	Hjorth Complexity	62.90323	98.71357	98.61149
Modified Hurst Exponent	Average Energy	Hjorth Mobility	62.90323	98.79196	98.68966
Modified Hurst Exponent	Average Energy	Coastline	56.45161	98.74585	98.62529
Modified Hurst Exponent	Hurst Exponent	Root Mean Square	87.09677	97.51937	97.48966
Modified Hurst Exponent	Hurst Exponent	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Hurst Exponent	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Hurst Exponent	Mean Absolute Value	88.70968	97.31188	97.28736
Modified Hurst Exponent	Hurst Exponent	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Hurst Exponent	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Hurst Exponent	Coastline	0	100	99.71494
Modified Hurst Exponent	Hurst Exponent	Average Energy	62.90323	98.97639	98.87356
Modified Hurst Exponent	Renyie Entropy	Root Mean Square	82.25807	97.4502	97.4069
Modified Hurst Exponent	Renyie Entropy	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Renyie Entropy	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Renyie Entropy	Mean Absolute Value	87.09677	97.49631	97.46667
Modified Hurst Exponent	Renyie Entropy	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Renyie Entropy	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Renyie Entropy	Coastline	0	100	99.71494

Modified Hurst Exponent	Renyie Entropy	Average Energy	62.90323	98.78735	98.68506
Modified Hurst Exponent	Renyie Entropy	Hurst Exponent	0	100	99.71494
Modified Hurst Exponent	Spectral Entropy	Root Mean Square	87.09677	97.22888	97.2
Modified Hurst Exponent	Spectral Entropy	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Spectral Entropy	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Spectral Entropy	Mean Absolute Value	87.09677	97.13667	97.10805
Modified Hurst Exponent	Spectral Entropy	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Spectral Entropy	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Spectral Entropy	Coastline	0	100	99.71494
Modified Hurst Exponent	Spectral Entropy	Average Energy	64.51613	98.86112	98.76322
Modified Hurst Exponent	Spectral Entropy	Hurst Exponent	0	100	99.71494
Modified Hurst Exponent	Spectral Entropy	Renyie Entropy	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Root Mean Square	83.87097	97.66691	97.62759
Modified Hurst Exponent	Shannon Entropy	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Mean Absolute Value	87.09677	97.5332	97.50345
Modified Hurst Exponent	Shannon Entropy	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Coastline	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Average Energy	66.12903	98.78274	98.68966
Modified Hurst Exponent	Shannon Entropy	Hurst Exponent	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Renyie Entropy	0	100	99.71494
Modified Hurst Exponent	Shannon Entropy	Spectral Entropy	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Root Mean Square	88.70968	97.43176	97.4069
Modified Hurst Exponent	Approximate Entropy	Min Absolute Value	0	100	99.71494

Modified Hurst Exponent	Approximate Entropy	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Mean Absolute Value	88.70968	97.45943	97.43448
Modified Hurst Exponent	Approximate Entropy	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Coastline	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Average Energy	66.12903	98.80118	98.70805
Modified Hurst Exponent	Approximate Entropy	Hurst Exponent	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Renyie Entropy	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Spectral Entropy	0	100	99.71494
Modified Hurst Exponent	Approximate Entropy	Shannon Entropy	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Root Mean Square	85.48387	97.25655	97.22299
Modified Hurst Exponent	Permutation Entropy	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Mean Absolute Value	87.09677	97.24733	97.21839
Modified Hurst Exponent	Permutation Entropy	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Coastline	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Average Energy	64.51613	98.91645	98.81839
Modified Hurst Exponent	Permutation Entropy	Hurst Exponent	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Renyie Entropy	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Spectral Entropy	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Shannon Entropy	0	100	99.71494
Modified Hurst Exponent	Permutation Entropy	Approximate Entropy	0	100	99.71494
Modified Hurst Exponent	Variance	Root Mean Square	74.19355	98.40926	98.34023
Modified Hurst Exponent	Variance	Min Absolute Value	58.06452	98.27093	98.15632

Modified Hurst Exponent	Variance	Max Absolute Value	64.51613	99.26227	99.16322
Modified Hurst Exponent	Variance	Mean Absolute Value	79.03226	98.30782	98.25287
Modified Hurst Exponent	Variance	Hjorth Complexity	61.29032	98.74585	98.63908
Modified Hurst Exponent	Variance	Hjorth Mobility	62.90323	98.79196	98.68966
Modified Hurst Exponent	Variance	Coastline	56.45161	98.74585	98.62529
Modified Hurst Exponent	Variance	Average Energy	64.51613	98.81501	98.71724
Modified Hurst Exponent	Variance	Hurst Exponent	62.90323	98.99023	98.88736
Modified Hurst Exponent	Variance	Renyie Entropy	64.51613	98.79196	98.69425
Modified Hurst Exponent	Variance	Spectral Entropy	64.51613	98.86112	98.76322
Modified Hurst Exponent	Variance	Shannon Entropy	66.12903	98.79657	98.70345
Modified Hurst Exponent	Variance	Approximate Entropy	66.12903	98.77813	98.68506
Modified Hurst Exponent	Variance	Permutation Entropy	64.51613	98.8934	98.7954
Modified Hurst Exponent	Skew	Root Mean Square	80.64516	98.37698	98.32644
Modified Hurst Exponent	Skew	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Skew	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Skew	Mean Absolute Value	87.09677	98.13722	98.10575
Modified Hurst Exponent	Skew	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Skew	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Skew	Coastline	0	100	99.71494
Modified Hurst Exponent	Skew	Average Energy	67.74194	98.97178	98.88276
Modified Hurst Exponent	Skew	Hurst Exponent	0	100	99.71494
Modified Hurst Exponent	Skew	Renyie Entropy	0	100	99.71494
Modified Hurst Exponent	Skew	Spectral Entropy	0	100	99.71494
Modified Hurst Exponent	Skew	Shannon Entropy	0	100	99.71494
Modified Hurst Exponent	Skew	Approximate Entropy	0	100	99.71494

Modified Hurst Exponent	Skew	Permutation Entropy	0	100	99.71494
Modified Hurst Exponent	Skew	Variance	67.74194	98.97639	98.88736
Modified Hurst Exponent	Kurtosis	Root Mean Square	87.09677	98.19716	98.16552
Modified Hurst Exponent	Kurtosis	Min Absolute Value	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Max Absolute Value	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Mean Absolute Value	88.70968	97.95279	97.92644
Modified Hurst Exponent	Kurtosis	Hjorth Complexity	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Hjorth Mobility	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Coastline	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Average Energy	66.12903	99.00406	98.91035
Modified Hurst Exponent	Kurtosis	Hurst Exponent	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Renyie Entropy	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Spectral Entropy	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Shannon Entropy	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Approximate Entropy	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Permutation Entropy	0	100	99.71494
Modified Hurst Exponent	Kurtosis	Variance	66.12903	99.02711	98.93333
Modified Hurst Exponent	Kurtosis	Skew	0	100	99.71494
Fractal Dimension	Min Absolute Value	Root Mean Square	93.54839	98.03578	98.02299
Fractal Dimension	Max Absolute Value	Root Mean Square	91.93548	98.44153	98.42299
Fractal Dimension	Max Absolute Value	Min Absolute Value	83.87097	98.58447	98.54253
Fractal Dimension	Mean Absolute Value	Root Mean Square	93.54839	98.28016	98.26667
Fractal Dimension	Mean Absolute Value	Min Absolute Value	91.93548	98.07728	98.05977
Fractal Dimension	Mean Absolute Value	Max Absolute Value	91.93548	98.43692	98.41839
Fractal Dimension	Hjorth Complexity	Root Mean Square	93.54839	98.22021	98.2069

Fractal Dimension	Hjorth Complexity	Min Absolute Value	91.93548	98.41848	98.4
Fractal Dimension	Hjorth Complexity	Max Absolute Value	85.48387	98.74124	98.70345
Fractal Dimension	Hjorth Complexity	Mean Absolute Value	91.93548	98.15105	98.13333
Fractal Dimension	Hjorth Mobility	Root Mean Square	91.93548	98.14644	98.12874
Fractal Dimension	Hjorth Mobility	Min Absolute Value	95.16129	98.24788	98.23908
Fractal Dimension	Hjorth Mobility	Max Absolute Value	88.70968	98.67208	98.64368
Fractal Dimension	Hjorth Mobility	Mean Absolute Value	93.54839	98.06806	98.05517
Fractal Dimension	Hjorth Mobility	Hjorth Complexity	91.93548	98.37237	98.35402
Fractal Dimension	Coastline	Root Mean Square	93.54839	98.16027	98.14713
Fractal Dimension	Coastline	Min Absolute Value	95.16129	98.13261	98.12414
Fractal Dimension	Coastline	Max Absolute Value	93.54839	98.31243	98.29885
Fractal Dimension	Coastline	Mean Absolute Value	93.54839	98.11417	98.10115
Fractal Dimension	Coastline	Hjorth Complexity	95.16129	98.29399	98.28506
Fractal Dimension	Coastline	Hjorth Mobility	95.16129	98.22483	98.21609
Fractal Dimension	Average Energy	Root Mean Square	93.54839	98.52453	98.51035
Fractal Dimension	Average Energy	Min Absolute Value	83.87097	98.51531	98.47356
Fractal Dimension	Average Energy	Max Absolute Value	90.32258	98.7228	98.69885
Fractal Dimension	Average Energy	Mean Absolute Value	93.54839	98.46459	98.45058
Fractal Dimension	Average Energy	Hjorth Complexity	88.70968	98.82424	98.7954
Fractal Dimension	Average Energy	Hjorth Mobility	87.09677	98.74585	98.71264
Fractal Dimension	Average Energy	Coastline	93.54839	98.49225	98.47816
Fractal Dimension	Hurst Exponent	Root Mean Square	93.54839	97.79602	97.78391
Fractal Dimension	Hurst Exponent	Min Absolute Value	95.16129	97.5747	97.56782
Fractal Dimension	Hurst Exponent	Max Absolute Value	91.93548	98.15566	98.13793
Fractal Dimension	Hurst Exponent	Mean Absolute Value	93.54839	97.6623	97.65058

Fractal Dimension	Hurst Exponent	Hjorth Complexity	93.54839	97.95279	97.94023
Fractal Dimension	Hurst Exponent	Hjorth Mobility	95.16129	97.96662	97.95862
Fractal Dimension	Hurst Exponent	Coastline	96.77419	97.89746	97.89425
Fractal Dimension	Hurst Exponent	Average Energy	91.93548	98.24788	98.22989
Fractal Dimension	Renyie Entropy	Root Mean Square	93.54839	98.18794	98.17471
Fractal Dimension	Renyie Entropy	Min Absolute Value	93.54839	98.07267	98.05977
Fractal Dimension	Renyie Entropy	Max Absolute Value	87.09677	98.5107	98.47816
Fractal Dimension	Renyie Entropy	Mean Absolute Value	93.54839	98.1695	98.15632
Fractal Dimension	Renyie Entropy	Hjorth Complexity	91.93548	98.35854	98.34023
Fractal Dimension	Renyie Entropy	Hjorth Mobility	93.54839	98.36776	98.35402
Fractal Dimension	Renyie Entropy	Coastline	95.16129	98.12339	98.11494
Fractal Dimension	Renyie Entropy	Average Energy	93.54839	98.64902	98.63448
Fractal Dimension	Renyie Entropy	Hurst Exponent	95.16129	97.59314	97.58621
Fractal Dimension	Spectral Entropy	Root Mean Square	93.54839	98.27554	98.26207
Fractal Dimension	Spectral Entropy	Min Absolute Value	93.54839	98.03117	98.01839
Fractal Dimension	Spectral Entropy	Max Absolute Value	87.09677	98.49687	98.46437
Fractal Dimension	Spectral Entropy	Mean Absolute Value	93.54839	98.23405	98.22069
Fractal Dimension	Spectral Entropy	Hjorth Complexity	91.93548	98.33549	98.31724
Fractal Dimension	Spectral Entropy	Hjorth Mobility	95.16129	98.28477	98.27586
Fractal Dimension	Spectral Entropy	Coastline	95.16129	97.98967	97.98161
Fractal Dimension	Spectral Entropy	Average Energy	91.93548	98.88879	98.86897
Fractal Dimension	Spectral Entropy	Hurst Exponent	95.16129	97.69919	97.69195
Fractal Dimension	Spectral Entropy	Renyie Entropy	93.54839	98.06806	98.05517
Fractal Dimension	Shannon Entropy	Root Mean Square	93.54839	98.3401	98.32644
Fractal Dimension	Shannon Entropy	Min Absolute Value	93.54839	98.03117	98.01839

Fractal Dimension	Shannon Entropy	Max Absolute Value	87.09677	98.39543	98.36322
Fractal Dimension	Shannon Entropy	Mean Absolute Value	93.54839	98.26171	98.24828
Fractal Dimension	Shannon Entropy	Hjorth Complexity	91.93548	98.34471	98.32644
Fractal Dimension	Shannon Entropy	Hjorth Mobility	93.54839	98.26632	98.25287
Fractal Dimension	Shannon Entropy	Coastline	95.16129	98.09572	98.08736
Fractal Dimension	Shannon Entropy	Average Energy	93.54839	98.8519	98.83678
Fractal Dimension	Shannon Entropy	Hurst Exponent	95.16129	97.61158	97.6046
Fractal Dimension	Shannon Entropy	Renyie Entropy	93.54839	97.96662	97.95402
Fractal Dimension	Shannon Entropy	Spectral Entropy	93.54839	98.045	98.03218
Fractal Dimension	Approximate Entropy	Root Mean Square	93.54839	98.26171	98.24828
Fractal Dimension	Approximate Entropy	Min Absolute Value	93.54839	97.90207	97.88966
Fractal Dimension	Approximate Entropy	Max Absolute Value	87.09677	98.61214	98.57931
Fractal Dimension	Approximate Entropy	Mean Absolute Value	93.54839	98.23405	98.22069
Fractal Dimension	Approximate Entropy	Hjorth Complexity	91.93548	98.41848	98.4
Fractal Dimension	Approximate Entropy	Hjorth Mobility	95.16129	98.43231	98.42299
Fractal Dimension	Approximate Entropy	Coastline	95.16129	98.10494	98.09655
Fractal Dimension	Approximate Entropy	Average Energy	93.54839	98.79657	98.78161
Fractal Dimension	Approximate Entropy	Hurst Exponent	95.16129	97.79602	97.78851
Fractal Dimension	Approximate Entropy	Renyie Entropy	93.54839	98.04039	98.02759
Fractal Dimension	Approximate Entropy	Spectral Entropy	93.54839	98.04039	98.02759
Fractal Dimension	Approximate Entropy	Shannon Entropy	93.54839	98.03117	98.01839
Fractal Dimension	Permutation Entropy	Root Mean Square	93.54839	98.01273	98
Fractal Dimension	Permutation Entropy	Min Absolute Value	93.54839	97.88362	97.87126
Fractal Dimension	Permutation Entropy	Max Absolute Value	87.09677	98.49225	98.45977
Fractal Dimension	Permutation Entropy	Mean Absolute Value	93.54839	97.90207	97.88966

Fractal Dimension	Permutation Entropy	Hjorth Complexity	91.93548	98.34471	98.32644
Fractal Dimension	Permutation Entropy	Hjorth Mobility	93.54839	98.31704	98.30345
Fractal Dimension	Permutation Entropy	Coastline	95.16129	98.14183	98.13333
Fractal Dimension	Permutation Entropy	Average Energy	93.54839	98.33549	98.32184
Fractal Dimension	Permutation Entropy	Hurst Exponent	95.16129	97.77757	97.77012
Fractal Dimension	Permutation Entropy	Renyie Entropy	93.54839	97.84674	97.83448
Fractal Dimension	Permutation Entropy	Spectral Entropy	93.54839	97.91129	97.89885
Fractal Dimension	Permutation Entropy	Shannon Entropy	93.54839	97.8744	97.86207
Fractal Dimension	Variance	Min Absolute Value	83.87097	98.51992	98.47816
Fractal Dimension	Variance	Max Absolute Value	90.32258	98.7228	98.69885
Fractal Dimension	Variance	Mean Absolute Value	93.54839	98.48303	98.46897
Fractal Dimension	Variance	Hjorth Complexity	87.09677	98.83346	98.8
Fractal Dimension	Variance	Hjorth Mobility	87.09677	98.74585	98.71264
Fractal Dimension	Variance	Coastline	93.54839	98.48764	98.47356
Fractal Dimension	Variance	Average Energy	93.54839	98.7689	98.75402
Fractal Dimension	Variance	Hurst Exponent	91.93548	98.24788	98.22989
Fractal Dimension	Variance	Renyie Entropy	93.54839	98.66286	98.64828
Fractal Dimension	Variance	Spectral Entropy	90.32258	98.8934	98.86897
Fractal Dimension	Variance	Shannon Entropy	93.54839	98.87495	98.85977
Fractal Dimension	Variance	Min Absolute Value	83.87097	98.51992	98.47816
Fractal Dimension	Variance	Max Absolute Value	90.32258	98.7228	98.69885
Fractal Dimension	Variance	Mean Absolute Value	93.54839	98.48303	98.46897
Fractal Dimension	Variance	Hjorth Complexity	87.09677	98.83346	98.8
Fractal Dimension	Variance	Hjorth Mobility	87.09677	98.74585	98.71264

Fractal Dimension	Variance	Coastline	93.54839	98.48764	98.47356
Fractal Dimension	Variance	Average Energy	93.54839	98.7689	98.75402
Fractal Dimension	Variance	Hurst Exponent	91.93548	98.24788	98.22989
Fractal Dimension	Variance	Renyie Entropy	93.54839	98.66286	98.64828
Fractal Dimension	Variance	Spectral Entropy	90.32258	98.8934	98.86897
Fractal Dimension	Variance	Shannon Entropy	93.54839	98.87495	98.85977
Fractal Dimension	Skew	Renyie Entropy	91.93548	97.98045	97.96322
Fractal Dimension	Skew	Spectral Entropy	91.93548	98.05883	98.04138
Fractal Dimension	Skew	Shannon Entropy	91.93548	97.97584	97.95862
Fractal Dimension	Skew	Approximate Entropy	91.93548	97.91129	97.89425
Fractal Dimension	Skew	Permutation Entropy	91.93548	98.00812	97.99081
Fractal Dimension	Skew	Variance	90.32258	98.56603	98.54253
Fractal Dimension	Kurtosis	Root Mean Square	93.54839	98.62136	98.6069
Fractal Dimension	Kurtosis	Min Absolute Value	93.54839	97.9574	97.94483
Fractal Dimension	Kurtosis	Max Absolute Value	87.09677	98.49225	98.45977
Fractal Dimension	Kurtosis	Mean Absolute Value	91.93548	98.45076	98.43218
Fractal Dimension	Kurtosis	Hjorth Complexity	91.93548	98.51531	98.49655
Fractal Dimension	Kurtosis	Hjorth Mobility	95.16129	98.45998	98.45058
Fractal Dimension	Kurtosis	Coastline	95.16129	98.32626	98.31724
Fractal Dimension	Kurtosis	Average Energy	93.54839	98.95334	98.93793
Fractal Dimension	Kurtosis	Hurst Exponent	95.16129	97.79602	97.78851
Fractal Dimension	Kurtosis	Renyie Entropy	93.54839	98.01734	98.0046
Fractal Dimension	Kurtosis	Spectral Entropy	93.54839	98.14183	98.12874
Fractal Dimension	Kurtosis	Shannon Entropy	93.54839	98.01273	98
Fractal Dimension	Kurtosis	Approximate Entropy	93.54839	97.9574	97.94483

Fractal Dimension	Kurtosis	Permutation Entropy	93.54839	97.94356	97.93103
Fractal Dimension	Kurtosis	Variance	91.93548	98.95795	98.93793
Fractal Dimension	Kurtosis	Skew	91.93548	98.128	98.11035
Fractal Dimension	Modified Hurst Exponent	Root Mean Square	93.54839	97.65308	97.64138
Fractal Dimension	Modified Hurst Exponent	Min Absolute Value	93.54839	97.96201	97.94943
Fractal Dimension	Modified Hurst Exponent	Max Absolute Value	90.32258	99.30376	99.27816
Fractal Dimension	Modified Hurst Exponent	Mean Absolute Value	93.54839	97.66691	97.65517
Fractal Dimension	Modified Hurst Exponent	Hjorth Complexity	91.93548	98.17872	98.16092
Fractal Dimension	Modified Hurst Exponent	Hjorth Mobility	95.16129	98.26632	98.25747
Fractal Dimension	Modified Hurst Exponent	Coastline	95.16129	97.71763	97.71035
Fractal Dimension	Modified Hurst Exponent	Average Energy	93.54839	98.49687	98.48276
Fractal Dimension	Modified Hurst Exponent	Hurst Exponent	95.16129	97.80985	97.8023
Fractal Dimension	Modified Hurst Exponent	Renyie Entropy	93.54839	97.98506	97.97241
Fractal Dimension	Modified Hurst Exponent	Spectral Entropy	93.54839	98.0035	97.99081
Fractal Dimension	Modified Hurst Exponent	Shannon Entropy	93.54839	97.96201	97.94943
Fractal Dimension	Modified Hurst Exponent	Approximate Entropy	93.54839	97.97123	97.95862
Fractal Dimension	Modified Hurst Exponent	Permutation Entropy	93.54839	98.045	98.03218
Fractal Dimension	Modified Hurst Exponent	Variance	93.54839	98.48764	98.47356
Fractal Dimension	Modified Hurst Exponent	Skew	90.32258	98.82424	98.8
Fractal Dimension	Modified Hurst Exponent	Kurtosis	93.54839	97.90668	97.89425
Standard Deviation	Min Absolute Value	Root Mean Square	85.48387	97.92973	97.89425
Standard Deviation	Max Absolute Value	Root Mean Square	83.87097	98.27554	98.23448
Standard Deviation	Max Absolute Value	Min Absolute Value	82.25807	98.27554	98.22989
Standard Deviation	Mean Absolute Value	Root Mean Square	87.09677	98.04961	98.01839
Standard Deviation	Mean Absolute Value	Min Absolute Value	87.09677	97.86979	97.83908

Standard Deviation	Mean Absolute Value	Max Absolute Value	83.87097	98.23405	98.1931
Standard Deviation	Hjorth Complexity	Root Mean Square	80.64516	98.36315	98.31264
Standard Deviation	Hjorth Complexity	Min Absolute Value	82.25807	98.22483	98.17931
Standard Deviation	Hjorth Complexity	Max Absolute Value	80.64516	98.50148	98.45058
Standard Deviation	Hjorth Complexity	Mean Absolute Value	83.87097	98.33087	98.28966
Standard Deviation	Hjorth Mobility	Root Mean Square	83.87097	98.35393	98.31264
Standard Deviation	Hjorth Mobility	Min Absolute Value	83.87097	98.26171	98.22069
Standard Deviation	Hjorth Mobility	Max Absolute Value	80.64516	98.5107	98.45977
Standard Deviation	Hjorth Mobility	Mean Absolute Value	82.25807	98.33087	98.28506
Standard Deviation	Hjorth Mobility	Hjorth Complexity	82.25807	98.3401	98.29425
Standard Deviation	Coastline	Root Mean Square	82.25807	98.09111	98.04598
Standard Deviation	Coastline	Min Absolute Value	85.48387	98.00812	97.97241
Standard Deviation	Coastline	Max Absolute Value	80.64516	98.35393	98.30345
Standard Deviation	Coastline	Mean Absolute Value	85.48387	98.08189	98.04598
Standard Deviation	Coastline	Hjorth Complexity	80.64516	98.40926	98.35862
Standard Deviation	Coastline	Hjorth Mobility	83.87097	98.36776	98.32644
Standard Deviation	Average Energy	Root Mean Square	69.35484	98.87034	98.78621
Standard Deviation	Average Energy	Min Absolute Value	70.96774	98.61675	98.53793
Standard Deviation	Average Energy	Max Absolute Value	67.74194	98.99484	98.90575
Standard Deviation	Average Energy	Mean Absolute Value	74.19355	98.72741	98.65747
Standard Deviation	Average Energy	Hjorth Complexity	66.12903	98.99023	98.89655
Standard Deviation	Average Energy	Hjorth Mobility	67.74194	98.94873	98.85977
Standard Deviation	Average Energy	Coastline	59.67742	98.87957	98.76782
Standard Deviation	Hurst Exponent	Root Mean Square	83.87097	97.94817	97.90805
Standard Deviation	Hurst Exponent	Min Absolute Value	83.87097	97.89746	97.85747

Standard Deviation	Hurst Exponent	Max Absolute Value	83.87097	98.20638	98.16552
Standard Deviation	Hurst Exponent	Mean Absolute Value	85.48387	97.82829	97.7931
Standard Deviation	Hurst Exponent	Hjorth Complexity	83.87097	98.2156	98.17471
Standard Deviation	Hurst Exponent	Hjorth Mobility	85.48387	97.7038	97.66897
Standard Deviation	Hurst Exponent	Coastline	85.48387	97.87901	97.84368
Standard Deviation	Hurst Exponent	Average Energy	72.58065	98.69974	98.62529
Standard Deviation	Renyie Entropy	Root Mean Square	82.25807	98.20177	98.15632
Standard Deviation	Renyie Entropy	Min Absolute Value	82.25807	97.82829	97.78391
Standard Deviation	Renyie Entropy	Max Absolute Value	82.25807	98.51992	98.47356
Standard Deviation	Renyie Entropy	Mean Absolute Value	82.25807	98.17411	98.12874
Standard Deviation	Renyie Entropy	Hjorth Complexity	80.64516	98.37237	98.32184
Standard Deviation	Renyie Entropy	Hjorth Mobility	83.87097	98.35393	98.31264
Standard Deviation	Renyie Entropy	Coastline	79.03226	98.14183	98.08736
Standard Deviation	Renyie Entropy	Average Energy	72.58065	98.6398	98.56552
Standard Deviation	Renyie Entropy	Hurst Exponent	85.48387	97.8329	97.7977
Standard Deviation	Spectral Entropy	Root Mean Square	83.87097	98.09572	98.05517
Standard Deviation	Spectral Entropy	Min Absolute Value	85.48387	97.85135	97.81609
Standard Deviation	Spectral Entropy	Max Absolute Value	83.87097	98.42309	98.38161
Standard Deviation	Spectral Entropy	Mean Absolute Value	87.09677	98.08189	98.05058
Standard Deviation	Spectral Entropy	Hjorth Complexity	80.64516	98.23405	98.18391
Standard Deviation	Spectral Entropy	Hjorth Mobility	83.87097	98.18794	98.14713
Standard Deviation	Spectral Entropy	Coastline	82.25807	98.05422	98.0092
Standard Deviation	Spectral Entropy	Average Energy	67.74194	98.86573	98.77701
Standard Deviation	Spectral Entropy	Hurst Exponent	83.87097	97.84213	97.8023
Standard Deviation	Spectral Entropy	Renyie Entropy	82.25807	98.22944	98.18391

Standard Deviation	Shannon Entropy	Root Mean Square	83.87097	98.04961	98.0092
Standard Deviation	Shannon Entropy	Min Absolute Value	85.48387	97.7038	97.66897
Standard Deviation	Shannon Entropy	Max Absolute Value	82.25807	98.41387	98.36782
Standard Deviation	Shannon Entropy	Mean Absolute Value	83.87097	97.99428	97.95402
Standard Deviation	Shannon Entropy	Hjorth Complexity	82.25807	98.34932	98.30345
Standard Deviation	Shannon Entropy	Hjorth Mobility	83.87097	98.34471	98.30345
Standard Deviation	Shannon Entropy	Coastline	82.25807	98.03578	97.99081
Standard Deviation	Shannon Entropy	Average Energy	72.58065	98.57525	98.50115
Standard Deviation	Shannon Entropy	Hurst Exponent	85.48387	97.90207	97.86667
Standard Deviation	Shannon Entropy	Renyie Entropy	83.87097	98.045	98.0046
Standard Deviation	Shannon Entropy	Spectral Entropy	83.87097	98.045	98.0046
Standard Deviation	Approximate Entropy	Root Mean Square	83.87097	98.08189	98.04138
Standard Deviation	Approximate Entropy	Min Absolute Value	85.48387	97.88823	97.85287
Standard Deviation	Approximate Entropy	Max Absolute Value	83.87097	98.30782	98.26667
Standard Deviation	Approximate Entropy	Mean Absolute Value	85.48387	98.06345	98.02759
Standard Deviation	Approximate Entropy	Hjorth Complexity	82.25807	98.15105	98.10575
Standard Deviation	Approximate Entropy	Hjorth Mobility	83.87097	98.18333	98.14253
Standard Deviation	Approximate Entropy	Coastline	83.87097	98.03117	97.99081
Standard Deviation	Approximate Entropy	Average Energy	70.96774	98.74585	98.66667
Standard Deviation	Approximate Entropy	Hurst Exponent	85.48387	97.83751	97.8023
Standard Deviation	Approximate Entropy	Renyie Entropy	82.25807	98.14644	98.10115
Standard Deviation	Approximate Entropy	Spectral Entropy	85.48387	98.07267	98.03678
Standard Deviation	Approximate Entropy	Shannon Entropy	83.87097	98.05883	98.01839
Standard Deviation	Permutation Entropy	Root Mean Square	82.25807	98.045	98
Standard Deviation	Permutation Entropy	Min Absolute Value	85.48387	97.93434	97.89885

Standard Deviation	Permutation Entropy	Max Absolute Value	82.25807	98.26632	98.22069
Standard Deviation	Permutation Entropy	Mean Absolute Value	82.25807	98.03117	97.98621
Standard Deviation	Permutation Entropy	Hjorth Complexity	82.25807	98.36776	98.32184
Standard Deviation	Permutation Entropy	Hjorth Mobility	80.64516	98.34932	98.29885
Standard Deviation	Permutation Entropy	Coastline	82.25807	98.10955	98.06437
Standard Deviation	Permutation Entropy	Average Energy	67.74194	98.83807	98.74943
Standard Deviation	Permutation Entropy	Hurst Exponent	83.87097	97.94817	97.90805
Standard Deviation	Permutation Entropy	Renyie Entropy	80.64516	98.19255	98.14253
Standard Deviation	Permutation Entropy	Spectral Entropy	82.25807	97.97123	97.92644
Standard Deviation	Permutation Entropy	Shannon Entropy	83.87097	98.07267	98.03218
Standard Deviation	Permutation Entropy	Approximate Entropy	82.25807	98.04039	97.9954
Standard Deviation	Variance	Root Mean Square	69.35484	98.90262	98.81839
Standard Deviation	Variance	Min Absolute Value	70.96774	98.61675	98.53793
Standard Deviation	Variance	Max Absolute Value	67.74194	99.00867	98.91954
Standard Deviation	Variance	Mean Absolute Value	74.19355	98.73663	98.66667
Standard Deviation	Variance	Hjorth Complexity	66.12903	98.99484	98.90115
Standard Deviation	Variance	Hjorth Mobility	64.51613	99.0225	98.92414
Standard Deviation	Variance	Coastline	62.90323	98.87034	98.76782
Standard Deviation	Variance	Average Energy	69.35484	98.90262	98.81839
Standard Deviation	Variance	Hurst Exponent	72.58065	98.72741	98.65287
Standard Deviation	Variance	Renyie Entropy	72.58065	98.64902	98.57471
Standard Deviation	Variance	Spectral Entropy	69.35484	98.88418	98.8
Standard Deviation	Variance	Shannon Entropy	72.58065	98.60753	98.53333
Standard Deviation	Variance	Approximate Entropy	70.96774	98.74585	98.66667
Standard Deviation	Variance	Permutation Entropy	69.35484	98.84729	98.76322

Standard Deviation	Skew	Root Mean Square	83.87097	98.19255	98.15172
Standard Deviation	Skew	Min Absolute Value	85.48387	98.01273	97.97701
Standard Deviation	Skew	Max Absolute Value	82.25807	98.27554	98.22989
Standard Deviation	Skew	Mean Absolute Value	87.09677	98.128	98.09655
Standard Deviation	Skew	Hjorth Complexity	80.64516	98.45076	98.4
Standard Deviation	Skew	Hjorth Mobility	83.87097	98.4277	98.38621
Standard Deviation	Skew	Coastline	80.64516	98.26632	98.21609
Standard Deviation	Skew	Average Energy	74.19355	98.82424	98.75402
Standard Deviation	Skew	Hurst Exponent	85.48387	98.03578	98
Standard Deviation	Skew	Renyie Entropy	82.25807	98.29399	98.24828
Standard Deviation	Skew	Spectral Entropy	83.87097	98.23866	98.1977
Standard Deviation	Skew	Shannon Entropy	82.25807	98.28016	98.23448
Standard Deviation	Skew	Approximate Entropy	82.25807	98.24327	98.1977
Standard Deviation	Skew	Permutation Entropy	83.87097	98.24327	98.2023
Standard Deviation	Skew	Variance	74.19355	98.81501	98.74483
Standard Deviation	Kurtosis	Root Mean Square	83.87097	98.51531	98.47356
Standard Deviation	Kurtosis	Min Absolute Value	87.09677	98.17872	98.14713
Standard Deviation	Kurtosis	Max Absolute Value	85.48387	98.80579	98.76782
Standard Deviation	Kurtosis	Mean Absolute Value	87.09677	98.40926	98.37701
Standard Deviation	Kurtosis	Hjorth Complexity	82.25807	98.54758	98.50115
Standard Deviation	Kurtosis	Hjorth Mobility	83.87097	98.57986	98.53793
Standard Deviation	Kurtosis	Coastline	82.25807	98.62136	98.57471
Standard Deviation	Kurtosis	Average Energy	70.96774	98.8934	98.81379
Standard Deviation	Kurtosis	Hurst Exponent	85.48387	98.16027	98.12414
Standard Deviation	Kurtosis	Renyie Entropy	82.25807	98.39543	98.34943

Standard Deviation	Kurtosis	Spectral Entropy	82.25807	98.64441	98.5977
Standard Deviation	Kurtosis	Shannon Entropy	83.87097	98.39082	98.34943
Standard Deviation	Modified Hurst Exponent	Approximate Entropy	88.70968	97.42715	97.4023
Standard Deviation	Modified Hurst Exponent	Permutation Entropy	85.48387	97.29805	97.26437
Standard Deviation	Modified Hurst Exponent	Variance	74.19355	98.41387	98.34483
Standard Deviation	Modified Hurst Exponent	Skew	80.64516	98.37698	98.32644
Standard Deviation	Modified Hurst Exponent	Kurtosis	87.09677	98.21099	98.17931
Standard Deviation	Fractal Dimension	Root Mean Square	93.54839	98.27093	98.25747
Standard Deviation	Fractal Dimension	Min Absolute Value	93.54839	98.04039	98.02759
Standard Deviation	Fractal Dimension	Max Absolute Value	91.93548	98.43692	98.41839
Standard Deviation	Fractal Dimension	Mean Absolute Value	93.54839	98.28016	98.26667
Standard Deviation	Fractal Dimension	Hjorth Complexity	93.54839	98.24788	98.23448
Standard Deviation	Fractal Dimension	Hjorth Mobility	91.93548	98.15105	98.13333
Standard Deviation	Fractal Dimension	Coastline	93.54839	98.16027	98.14713
Standard Deviation	Fractal Dimension	Average Energy	93.54839	98.53836	98.52414
Standard Deviation	Fractal Dimension	Hurst Exponent	91.93548	97.80524	97.78851
Standard Deviation	Fractal Dimension	Renyie Entropy	93.54839	98.18333	98.17012
Standard Deviation	Fractal Dimension	Spectral Entropy	93.54839	98.28477	98.27126
Standard Deviation	Fractal Dimension	Shannon Entropy	93.54839	98.36315	98.34943
Standard Deviation	Fractal Dimension	Approximate Entropy	93.54839	98.28016	98.26667
Standard Deviation	Fractal Dimension	Permutation Entropy	93.54839	98.02656	98.01379
Standard Deviation	Fractal Dimension	Variance	93.54839	98.5522	98.53793
Standard Deviation	Fractal Dimension	Skew	91.93548	98.1695	98.15172
Standard Deviation	Fractal Dimension	Kurtosis	93.54839	98.63519	98.62069
Standard Deviation	Fractal Dimension	Modified Hurst Exponent	93.54839	97.6623	97.65058

الملخص

مرض الصرع هو أحد أكثر الأمراض العصبية انتشارا حيث أنه يؤثر على حياة الملايين من البشر حول العالم ولهذا السبب فقد عمل الكثير على تقديم أنظمة للكشف آليا عن نوبات الصرع.

العمل المقترح في هذه الأطروحة يهدف الى تصميم وتنفيذ دائرة الكترونية مدمجة يمكن زرعها داخل المخ لتعمل على الكشف عن نوبات الصرع. هذه النظام المتكامل القادر على الكشف عن الصرع يجب أن يتكون من أربع مراحل: تجهيز البيانات, استخلاص الخصائص, اختيار أفضل الخضائص وأخيرا التصنيف. بالنسبة لمرحلة استخراج الخصائص فقد قمت باستخراج 20 خاصية خطية وغير خطية واختبارهم لقياس مدى كفائتهم في الكشف عن الصرع. وبعد ذلك قمنا بالبحث عن افضل مزيج من هذه الخصائص يمكننا من الوصول لأفضل أداء بأقل عدد ممكن من الخصائص.

أما بالنسبة لمرحلة التصنييف فقد قمنا باستخدام اكثر من تقنية لتعليم الآلة لتصنيف لحظات الصرع وهذه التقنيات هى: الشبكات العصبية الاصطناعية و الات متجه الدعم وبعد ذلك قمنا بمقارنة أداء كل من التقنيتين وكذلك المساحة والطاقة المستهلكة في كل منهما. إضافة إلى ذلك فقد قمنا بتعديل على تصميم الشبكات العصبية الاصطناعية لزيادة كفائتها.

وبما إن الكشف عن الصرع هو مشكلة معقدة وتدريب آلات متجهات الدعم تصبح عملية صعبة جدا في مثل هذه المشاكل فقد قمنا بتصميم دائرة مسرع لتساعد في تدريب الات متجهات الدعم باستخدام اكثر من خوارزمية وهم: الصعود المتدرج و التحسين المتتالى.

وأخيرا قمنا بالتعاون مع فريق بحثى من كلية العلوم جامعة القاهرة و وان لاب بالعمل على استخراج قاعدة بيانات جديدة تحتوى على اشارات كهربية للمخ لعدد من الفئران لاستخدامها في تقييم أنظمة الكشف عن الصرع.

مهندس: محمد عادل عطية الهادى الجمال

تاريخ الميلاد: 1993\12\05

الجنسية: مصرى

تاريخ التسجيل: 2016\10\01

تاريخ المنح: 2018\....\....

القسم: هندسة الإلكترونيات والاتصالات الكهربية

الدرجة: ماجستير العلوم

المشرفون:

ا.د. أحمد نادر محى الدين

د. حسن مصطفى حسن

الممتحنون:

أ.د. يحيي حسن غلاب (الممتحن الخارجي)

أ.د. محمد فتحى أبو اليزيد (الممتحن الداخلي)

أ.د. أحمد نادر محى الدين (المشرف الرئيسي)

عنوان الرسالة:

تصميم وتنفيذ عتاد لتقنيات تعليم الآلة لاستخدامها فى الكشف عن نوبات الصرع العصبية

الكلمات الدالة:

الكشف عن الصرع، تعليم الآلة، آلة متجه الدعم، الشبكات العصبية الاصطناعية، مسرع

ملخص الرسالة:

فى هذه الأطروحة نقدم نظام متكامل للكشف عن نوبات الصرع العصبية. بالنسبة لمرحلة استخلاص الخصائص فقد قمنا باستخراج اكتر من عشرين خاصية خطية وغير خطية لاستخدامها فى تمييز نوبات الصرع. كما قمنا باختبار كفاءة هذه الخصائص ف الكشف عن الصرع. أما بالنسبة لعملية التصنيف فقد قمنا باستخدام اكتر من تقنية وهم: آلات متجه الدعم و الشبكات العصبية الاصطناعية. وللعمل على تسريع عمليه تدريب الات متجه الدعم فقد قمنا بتصميم مسرع لتدريبها بأكتر من خوارزمية وهى: الصعود المتدرج و التحسين المتتالى. أخيرا قمنا بالعمل على استخراج قاعدة بيانات جديدة تحتوى على اشارات كهربية للمخ لعدد من الفئران بالتعاون مع فريق بحثى من كلية العلوم جامعة القاهرة و وان لاب.

تصميم وتنفيذ عتاد لتقنيات تعليم الآلة لاستخدامها فى الكشف عن نوبات الصرع العصبية

اعداد

محمد عادل عطية الهادى الجمال

رسالة مقدمة إلى كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

يعتمد من لجنة الممتحنين:

الاستاذ الدكتور: ا.م.د. أحمد نادر محى الدين — المشرف الرئيسى

الاستاذ الدكتور: ا.د. محمد فتحى أبو اليزيد — الممتحن الداخلي

الاستاذ الدكتور: ا.م.د. يحيي حسن غلاب — الممتحن الخارجي

- أستاذ مساعد بكلية الهندسة جامعة حلوان

كليـة الهندسـة - جامعـة القاهـرة
الجيـزة - جمهوريـة مصـر العربيـة
2018

- *يجب على الطالب الرجوع الى ادارة الدراسات العليا لأختلاف بعض الأقسام حول التخصص

تصميم وتنفيذ عتاد لتقنيات تعليم الآلة لاستخدامها فى الكشف عن نوبات الصرع العصبية

اعداد

محمد عادل عطية الهادى الجمال

رسالة مقدمة إلى كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

تحت اشراف

أ.م.د. أحمد نادر محى الدين	**د. حسن مصطفى حسن**
أستاذ مساعد	مدرس
قسم هندسة الإلكترونيات	قسم هندسة الإلكترونيات
والإتصالات الكهربية	والإتصالات الكهربية
كلية الهندسة – جامعة القاهرة	كلية الهندسة – جامعة القاهرة

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـرالعربيــة
2018

- *يجب على الطالب الرجوع الى ادارة الدراسات العليا لأختلاف بعض الأقسام حول التخصص

تصميم وتنفيذ عتاد لتقنيات تعليم الآلة لاستخدامها فى الكشف عن نوبات الصرع العصبية

اعداد

محمد عادل عطية الهادى الجمال

رسالة مقدمة إلى كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربية

كليـــة الهندســـة - جامعـــة القاهـــرة
الجيـزة - جمهوريـة مصـرالعربيـــة
2018

- *يجب على الطالب الرجوع الى ادارة الدراسات العليا لأختلاف بعض الأقسام حول التخصص