REPUBLIC OF TURKEY

ALTINBAŞ UNIVERSİTY

Institute of Graduate Studies

Information Technology

MACHINE LEARNING FOR ABNORMALITIES OF THE HEART RHYTHM CLASSIFICATION

FIRAS HAMAD AHMED

Master Thesis

Supervisor

Prof. Dr. Sefer Kurnaz

Istanbul 2021

MACHINE LEARNING FOR ABNORMALITIES OF THE HEART RHYTHM CLASSIFICATION

AHMED, FIRAS HAMAD

Information Technology

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science

ALTINBAŞ UNIVERSITY

2021

The thesis titled “Machine learning for Abnormalities of the heart rhythm classification “prepared and presented by “FIRAS HAMAD AHMED“ and submitted on 1/08/2021 has been **accepted by majority of votes** for the degree of Master of Science Thesis in Information Technology.

Prof.Dr.Sefer Kurnaz
Supervisor

Thesis Defense Jury Members:

Asst.Prof.Dr. Sefer KURNAZ	School of Engineering and Natural Sciences, Altinbas University	
Prof.Dr. Osman Nuri Uçan	School of Engineering and Natural Sciences, Altinbas University	
Prof.Dr. Mesut Razbonyalı	School of Engineering and Natural Sciences, Maltepe university	

I hereby declare that this thesis/dissertation meets all format and submission requirements of a Master of Science.

Accepted by Altınbaş University Institute of Graduate Studies on the following date:

____/____/____

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Firas Hamad AHMED

Signature

DEDICATION

First I would like to Allah Almighty for the power of mind , health, strength , guidance knowledge and skills to complete this study .This thesis is wholeheartedly dedicated to my beloved father, who have been my source of inspiration, he tells me that" every success in your life will be best gift for me". To my mother, who have been supporting me with the kind and pure love.

ACKNOWLEDGEMENTS

I would like to thank my supervisor professor Prof. Dr. Osman Nuri Ucan for all support during my study. It's great pleasure to express my deepest gratitude to my friends who have shared with me best moments during my study for the Master degree.

ABSTRACT

MACHINE LEARNING FOR ABNORMALITIES OF THE HEART RHYTHM CLASSIFICATION

AHMED, FIRAS HAMAD AHMED

M.Sc, Computer Engineering , Altinbas University

Supervisor: Sefer KURNAZ

Date: 08/2021

Pages: 60

Artificial Neural Networks (ANNs) are inspired by biological neural connections in the brain. In ANNs, biological neurons are replaced by artificial neurons (perceptrons), which were first described by Rosenblatt in 1958. Abnormalities of the heart rhythm (arrhythmia) are among the most common health problems in the population. They range in severity from benign palpitations to total loss of cardiac function and death .Electrocardiogram signal is a recording of the electrical activity of the heart. Electrodes on the skin detect small changes of voltage which are caused by heart muscle depolarization and the following repolariza- tion. Two electrodes can measure the electric potential between them. This pair of electrodes is called an ECG lead. Every lead provides a different viewpoint of a heart's electrical activity. The most clinically used lead formation is 12-lead from ten electrodes. This work focuses on 1-lead (two electrodes) ECG often used in telemonitoring devices.

One of the methods to diagnose heart arrhythmia is an analysis of electrocardiogram. However, a trained human expert is needed for precise diagnosis of a heart condition. Some arrhythmia, such as atrial fibrillation, require long-term monitoring for days or even weeks. However, long-term cardiac monitoring signals are too long to be thoroughly analyzed by a human effectively, so the analysis has to be done automatically by a model. With automated arrhythmia classification, the expert only has to look at parts of the signal suggested by the model.

Keywords: EEG Data, ANN, Diagnosis , Biomedical Engineering.

TABLE OF CONTENTS

ABSTRACT ..... **vii**

LIST OF TABLES ..... **viii**

LIST OF FIGURES ..... **ix**

LIST OF ABBREVIATIONS ..... **xi**

1. INTRODUCTION 1

1.1 ELECTROCARDIOGRAM 2

1.2 SELECTED ARRHYTHMIA DESCRIPTIONS 4

1.3 AIMS OF THE THESIS 6

2. NEURAL NETWORK **7**

2.1 PERCEPTRON 8

2.2 PERCEPTRON LEARNING ALGORITHM 9

2.3 FAST PERCEPTRON TRAINING 11

2.4 MULTI-LAYER FEED FORWARD NETWORKS 14

2.5 BACK-PROPAGATION LEARNING ALGORITHM 16

3. MACHINE LEARNING **17**

3.1 UNSUPERVISED LEARNING 19

3.2 REINFORCEMENT LEARNING 20

3.3 SEMI-SUPERVISED LEARNING 22

3.4 SUPERVISED LEARNING 25

3.5 FEED FORWARD NEURAL NETWORK 27

4. EVALUATION 29

4.1 DATASETS USED 29

4.2 TRAINING 29

4.3 MODELS WITH DIRECT SIGNAL INPUT 32

4.3 MODEL PERSISTENCE 33

4.4 METRICS 34

4.5 RESULTS 34

5. CONCLUSION 38

REFERENCES 39

LIST OF FIGURES

Pages

Figure 1.1: Electrocardiograms of normal rhythm, atrial fibrillation and a noisy signal (CINC2017 dataset [2]).. 2

Figure 1.2: Holter monitor, a type of ECG telemonitoring device [3].. 3

Figure 1.3: ECG of a single beat in normal sinus rhythm [6].. 4

Figure 1.4: Premature ventricular contraction (CPSC2018/A0080) occuring at 1.1 seconds from the start of the recording.. 5

Figure 1.5: Ventricular tachycardia / fibrillation (CUDB/cu05) [12].. 6

Figure 2.1: An example neuron unit with five inputs and bias. 8

Figure 2.2: Geometric interpretation of perceptron classification and learning 10

Figure 2.3: Initialization of weights according to 13

Figure 2.4: Illustration of one step in fast perceptron training 15

Figure 2.5: An example MLP network with two hidden layers 16

Figure 4.1: The F1 macro averaged scores on the test partitions, CINC2017 dataset 36

Figure 4.2: The F1 macro averaged scores on the test partitions, CPSC2018. 37

LIST OF ABBREVIATIONS

ML	Machine Learning
NLP	Natural Language Processing
AI	Artificial Intelligence
NB	Naïve Bays
SVM	Support Victor Machine
LSI	Latin Semantic Index
GOM	Goals of Matrices

1. INTRODUCTION

Machine Learning is a field of study in Artificial Intelligence (AI) that enables programing of intelligent computer software that is able to learn from experience to improve its performance [16]. This definition fits a very broad amount of techniques including linear and logistic regression, clustering, decision tree learning, probabilistic classification, deep learning using artificial neural networks and many more [1]. The advancements at the end of 20th and the beginning of 21st century brought many improvements in Machine Learning (ML) and Natural Language Processing (NLP) algorithms allowing us to predict and determine the correct action in many domains based solely on prior knowledge with almost no human interaction [2]. The possible applications of these techniques are very broad ranging from medicine to transportation, engineering, research and many more [2, 3].

Abnormalities of the heart rhythm (arrhythmia) are among the most common health problems in the population. They range in severity from benign palpitations to total loss of cardiac function and death [3]. One of the methods to diagnose heart arrhythmia is an analysis of electrocardiogram. However, a trained human expert is needed for precise diagnosis of a heart condition. Some arrhythmia, such as atrial fibrilla- tion, require long-term monitoring for days or even weeks. However, long-term cardiac monitoring signals are too long to be thoroughly analysed by a human effectively, so the analysis has to be done auto- matically by a model. With automated arrhythmia classification, the expert only has to look at parts of the signal suggested by the model.

The model has to be able to classify heart rate abnormalities, the normal rhythm and noisy ECGs.

1.1 ELECTROCARDIOGRAM

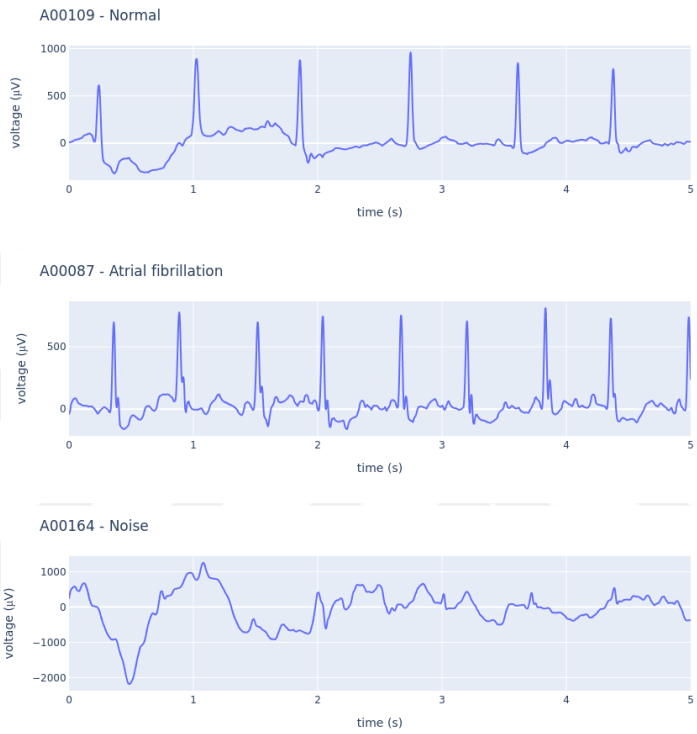

Figure 1.1: Electrocardiograms of normal rhythm, atrial fibrillation and a noisy signal (CINC2017 dataset [2]).

Electrocardiogram signal is a recording of the electrical activity of the heart. Electrodes on the skin detect small changes of voltage which are caused by heart muscle depolarization and the following repolariza- tion. Two electrodes can measure the electric potential between them. This pair of electrodes is called an ECG lead. Every lead provides a different viewpoint of a heart's electrical activity. The most clinically used lead formation is 12-lead from ten electrodes. This work focuses on 1-lead (two electrodes) ECG often used in telemonitoring devices.

1.1.1 Telemonitoring

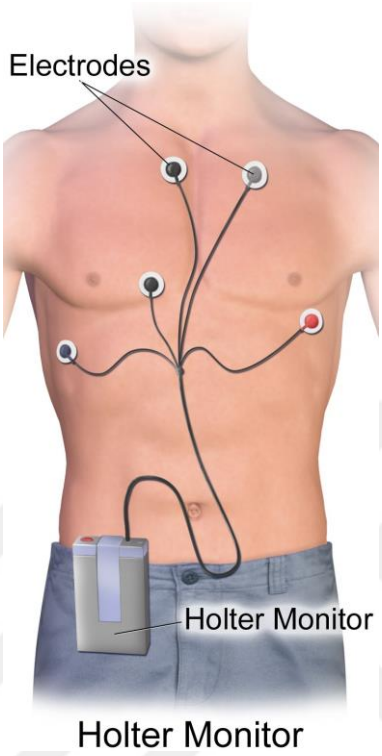

Figure 1.2: Holter monitor, a type of ECG telemonitoring device [3].

Telemonitoring is defined as "the use of information technology to monitor patients at a distance" [4]. In the context of ECG monitoring, a telemonitoring device is usually a small, wearable device such as a Holter monitor (Figure 1.2). Telemonitoring system can help decrease hospital admissions in cardiac patients [5].

1.1.2 Specific ECG elements

In order to identify heart arrhythmias, we have to recognize the characteristic patterns in ECG. The essential elements of an ECG include P wave, QRS complex and T wave [1]. All these waves are shown together in Figure 1.3.

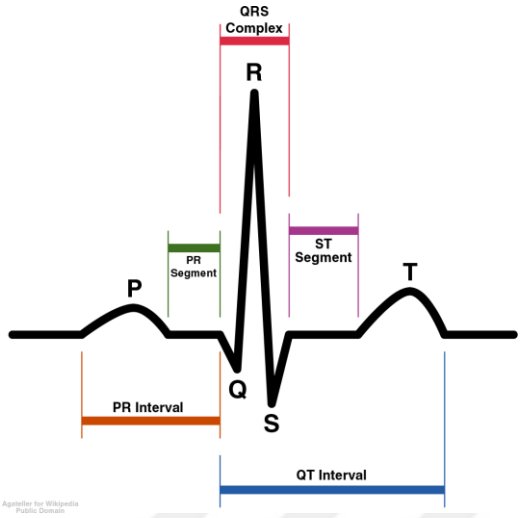

Figure 1.3: ECG of a single beat in normal sinus rhythm [6].

The P wave represents the electrical depolarization and the re- sulting muscle contraction of both atria, the upper chambers of the heart. The QRS complex represents the depolarization of the ventri- cles, lower chambers of the heart. The horizontal ST segment follows, and the cardiac cycle ends with a broad T wave, which represents the ventricular repolarization [7].

RR interval

RR interval, also called the inter-beat interval, is the time elapsed be- tween two consecutive R waves which can be directly used to calculate the average heart rate HR:

$$HR = \frac{60}{\text{RR interval in seconds}} \qquad (1.1)$$

where the RR interval in seconds is averaged over several consecutive R peaks.
Several relevant machine learning features can be extracted from the RR intervals. Heart rate variability methods are concerned pre- cisely with the problem of analysing inter-beat intervals. We discuss these methods more thoroughly in section 2.3.5.

1.2 SELECTED ARRHYTHMIA DESCRIPTIONS

To get a basic idea of the labels that are being classified in this work, a brief description of a few heart arrhythmia follows.

1.2.1 Atrial Fibrillation

Atrial fibrillation (AF) is an arrhythmia usually detected from ECG by irregular heartbeats (RR interval length is unstable) and an absence of a P wave preceding the QRS complex. AF is the most common cardiac rhythm disorder and is dangerous for the patient [8]. Atrial fibrillation increases the risk of heart failure, stroke, coronary heart disease and death [9]. An example of atrial fibrillation electrocardiogram is shown in Figure 1.1. In the figure, we can see the characteristic irregular RR intervals.

1.2.2 Premature Ventricular Contraction

A premature ventricular contraction (PVC) is a premature beat that most often happens by low oxygenation of the ventricles. It is easily recognizable by the usually enlarged QRS on the ECG (Figure 1.4) and the compensatory pause after the PVC which is longer than usual.

Single premature beats normally occur even in healthy people, but six or more PVCs per minute are considered pathological [10].

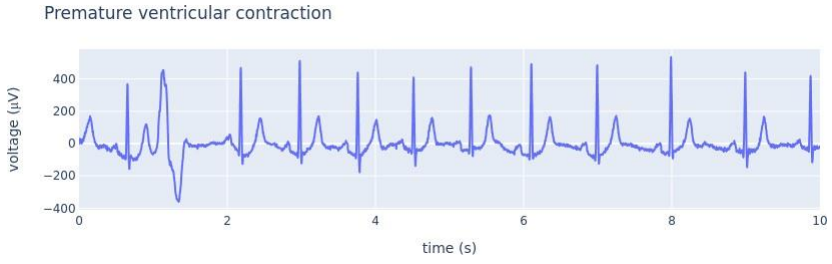

Figure 1.4: Premature ventricular contraction (CPSC2018/A0080) oc- curing at 1.1 seconds from the start of the recording.

1.2.3 Ventricular Tachycardia

Ventricular tachycardia is a run of four or more premature ventricular contractions. An example of this arrhythmia is shown in Figure 1.5.

Ventricular tachycardia is a dangerous arrhythmia as it can progress into a ventricular flutter and ventricular fibrillation, which requires immediate defibrillation as there is no effective cardiac output [11].

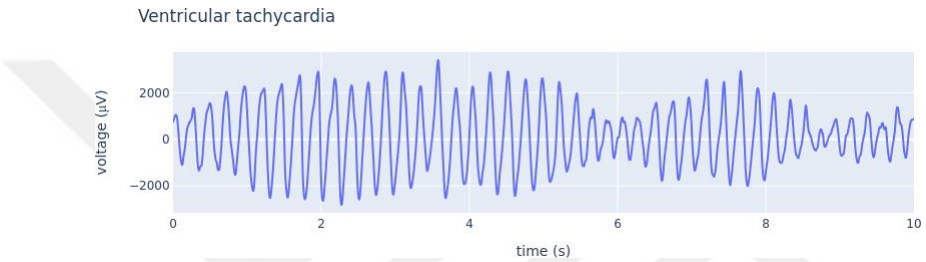

Figure 1.5: Ventricular tachycardia / fibrillation (CUDB/cu05) [12].

1.3 AIMS OF THE THESIS

The thesis aims to design and compare various machine learning mod- els to classify heart rhythm. The resulting experiment is designated with respect to reproducibility and extensibility.

2. NEURAL NETWORKS

Artificial Neural Networks (ANNs) are inspired by biological neural connections in the brain. In ANNs, biological neurons are replaced by artificial neurons (perceptrons), which were first described by Rosen- blatt [53] in 1958. Artificial neurons have multiple weighted inputs that are added and passed to an activation function such as sigmoid or threshold function. The output of a neuron is defined by the function

$$F(\underline{x}) = \phi(\underline{x} \cdot \underline{w}), \tag{2.1}$$

where $\underline{x}$ is the vector of features, $\underline{w}$ is the vector of weights, $\cdot$ is a dot product operation and ϕ is the activation function. For a visual example, see Figure 2.2. The common variation is to add bias, which is implemented by expanding vector $\underline{x}$ by one dimension and inserting 1 at zero dimension. The vector $\underline{w}$ is also expanded, therefore the product of first elements in both vectors gives the bias.

The network consists of hierarchically ordered layers of neurons, as seen in Figure 2.3. The first and the last layers are called input and output layers, respectively. Between the input and output layers, there are hidden layers. The number of hidden layers and the number of neurons in the hidden layers is a hyperparameter. The number of neurons in the input and output layers is given by the number of features and the number of predicted classes, respectively. Since a network consists of multiple layers of artificial neurons (perceptrons), it is sometimes called the Multi-Layered Perceptron (MLP).

At prediction time, neurons from the previous layer pass output to the inputs of neurons in the next layer through learned weighted connections. That is referred to as the feed-forward pass.

The learning process starts with a random weight initialization. The network is then trying to compute weight updates for each batch of data by optimizing Gradient Descent (GD) or its variants with respect to a loss function. Weight updates are computed with a backpropaga- tion algorithm, as described by Goodfellow et al. [23].

The most crucial hyper-parameter, except for the architecture of ANNs, is the learning rate. The learning rate determines what fraction of updates is used in the GD. With a high learning rate, a network is likely to fluctuate around a minimum or even diverge, while a small learning rate leads to slow convergence.

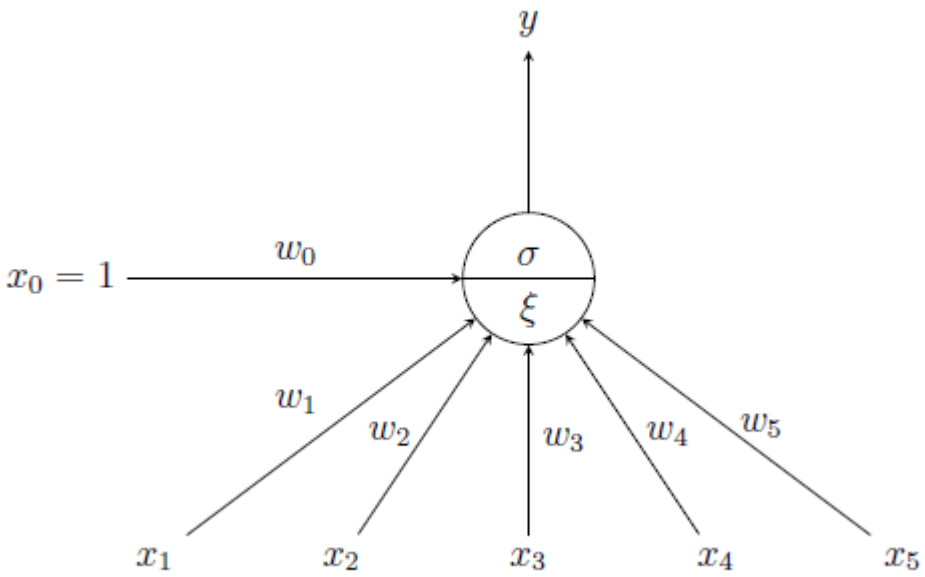

Figure 2.1: An example neuron unit with five inputs and bias

2.2 PERCEPTRON

The basic constituent unit of most neural networks is called a perceptron or simply an (articifical) neuron. It can be viewed as a simplified model of a biological neuron. Much like its biological counterpart, a perceptron is sort of a 'black box' with some fixed number of inputs (sometimes referred to as dendrites) and a single output (also known as the axon).

Each input has a corresponding weight indicating the extent to which the particular connection influences the resulting output.

The rough idea described above can be formalized as follows. Let X = (x1……xn) be the vector of input values and w = (w1……wn) the vector of the corresponding weights. To calculate the output, we need to compute the weighted sum

$$\xi = \mathbf{wX} = \sum_{i=1}^{n} w_i x_i \quad (2.1)$$

(sometimes called the inner potential of the neuron) first. The weighted sum is subsequently passed to the activation function _, yielding the output

$$y = \sigma(\xi) = \sigma\left(\sum_{i=1}^{n} w_i x_i\right). \quad (2.2)$$

The activation function _ may be of various kinds, depending on the task the neuron is designed

to perform. In the case of (two-class) classification, threshold activation functions are widely used. A threshold The (two-dimensional) sample vectors are split into two groups, labeled either as 0 (red points) or 1 (blue points). The current weight vector w(k) defines a hyperplane h(k) which splits the vector space into two half-spaces. Samples in one half-space (the 'left' one in the graph) are classified as 0, samples in the other one as 1.

This configuration gives rise to two classification errors: one false negative (x1) and one false positive (x2). The new weight vector w(k+1) is obtained from the old one by adding all the vectors corresponding to the false negative samples (here only x1) and subtracting the false positive vectors (x2 in our case). (In fact, the error vectors are multiplied by the parameter " first; here $\varepsilon = 1$.) It is the normal vector of a new separating hyperplane h(k+1), which already classifies all the samples correctly.

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq h, \\ 0 & \xi < h. \end{cases} \tag{2.3}$$

activation function has a general form where h is an arbitrary (but fixed) real-valued parameter. Like in a biological neuron, the 'action potential' (i.e. non-zero output) is generated whenever the weighted sum of inputs exceeds a pre-defined threshold value and vice versa.

A variant of an artificial neuron is a neuron with bias. Such a neuron contains an extra formal input x0 whose value is always 1. This input has a corresponding weight w0. When the weighted sum is computed, the product w0x0 = w0 is also included in it, effectively adding w0 to the weighted sum of the remaining inputs. Note that if w0 = h where h is the threshold mentioned above, the activation function _ can be equivalently defined as

$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 0, \\ 0 & \xi < 0. \end{cases} \tag{2.4}$$

This approach is more convenient for many uses, notably for learning, as will be shown in a short while. Henceforth, any mention of the term 'neuron' is to be understood as a neuron with bias unless specified otherwise.

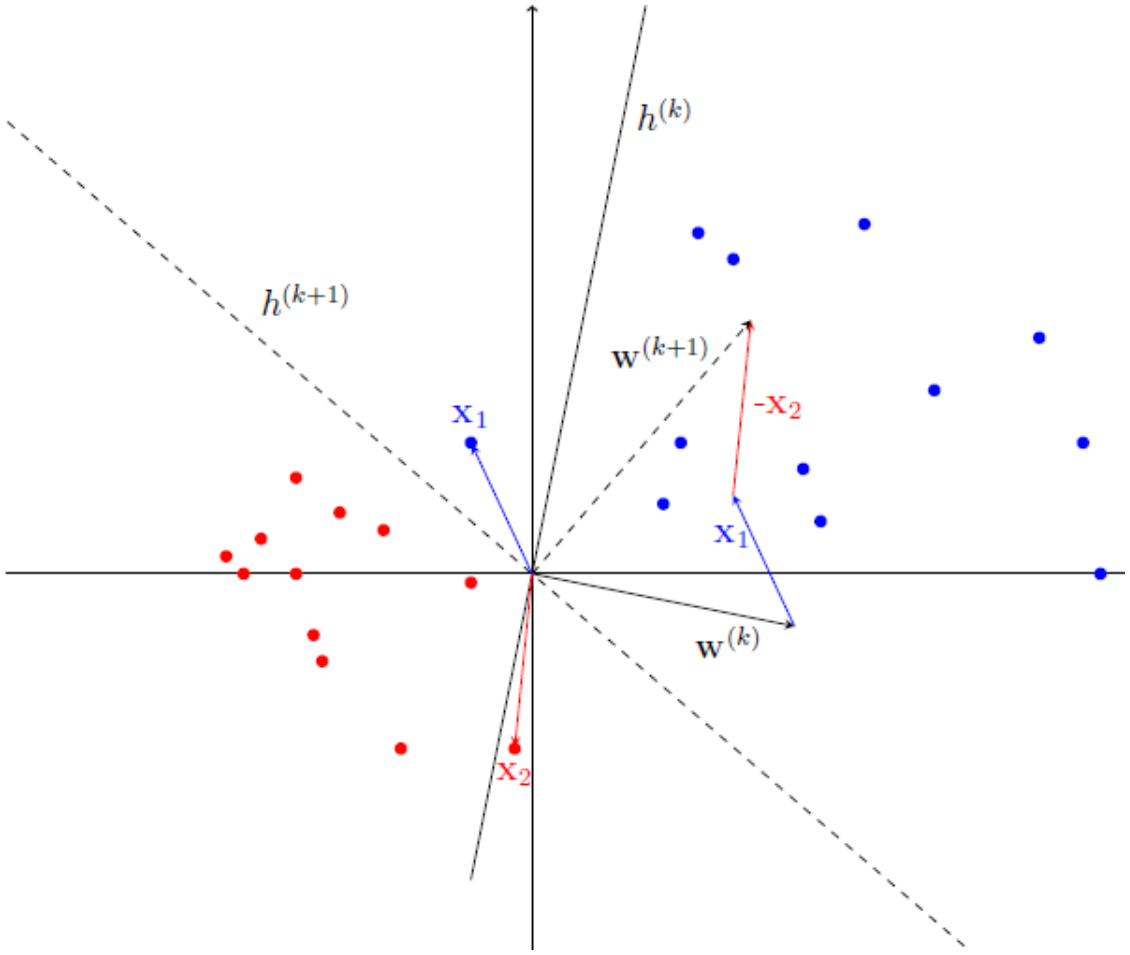

Figure 2.2: Geometric interpretation of perceptron classification and learning.

2.3 PERCEPTRON LEARNING ALGORITHM

Considering what has been said so far, a neuron can be viewed as an nary function from Rn to the set f0; 1g. For each n-dimensional real vector, it returns either 0 or 1, the choice being made as defined in 3.2. Note that _ = w0x0 + …… + wnxn = 0 is an equation of an (n - 1)-dimensional hyperplane which separates the space Rn into two half-spaces. The first half-space contains precisely those vectors X for which £(X) ≥ 0; the other half-space those for which £(X) < 0. This hyperplane is fully defined by the vector w = (w0……wn): the components (w1……wn) constitute the normal vector of the hyperplane, while the terms w0 w1……w0 wn define its intersection points with the corresponding axes. In every subsequent step, the weights are computed as

$$\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} + \varepsilon \cdot \sum_{k=1}^{p} (d_k - \sigma(\mathbf{w}^{(t-1)}\mathbf{X}_k)) \cdot \mathbf{X}_k, \tag{2.4}$$

It was shown already by Frank Rosenblatt that the algorithm always converges, provided that the sets of examples labeled with 0, resp. 1 are linearly separable. That is to say, if a hyperplane exists such that all examples labeled as 0 are located on one side of it and all examples
labeled as 1 on the other, the algorithm guarantees to find it in a finite time.

The condition of linear separability imposes a strong limitation on the data to which the perceptron algorithm can be applied. Not only it fails to learn functions as simple as the logical XOR, but—more importantly—the algorithm is not robust enough to cope with outliers.

A single outlier in the vicinity of the other cluster is enough to make the algorithm run ad infinitum without success. Since real-life data are generally not linearly separable, other techniques must be employed to build a succesful classifier.

2.4 FAST PERCEPTRON TRAINING

The basic perceptron algorithm, as formulated in the previous section, raises a couple of questions which ought to be discussed first in order to boost the algorithm's effectively (most importantly, the rate of convergence).

First, what values should the weights be set to during the initialization? And second, how to choose the optimal value for the parameter ", i.e. learning rate? Should it be constant, or is it better for it to change in each iteration?

One method to tackle these issues was proposed by Gkanogiannis and Kalamboukis (2009). In their approach, the centers of both classes of samples are computed first in the following manner:

$$\mathbf{C}_0 = \frac{1}{|C_0|} \sum_{\mathbf{x}_i \in C_0} \mathbf{x_i} \tag{2.4}$$

$$\mathbf{C}_1 = \frac{1}{|C_1|} \sum_{\mathbf{x}_i \in C_1} \mathbf{x_i} \tag{2.5}$$

The vector w = C1 - C0j is then used as the initial weight vector (w1…..wn). The method to find

the initial weight vector is illustrated

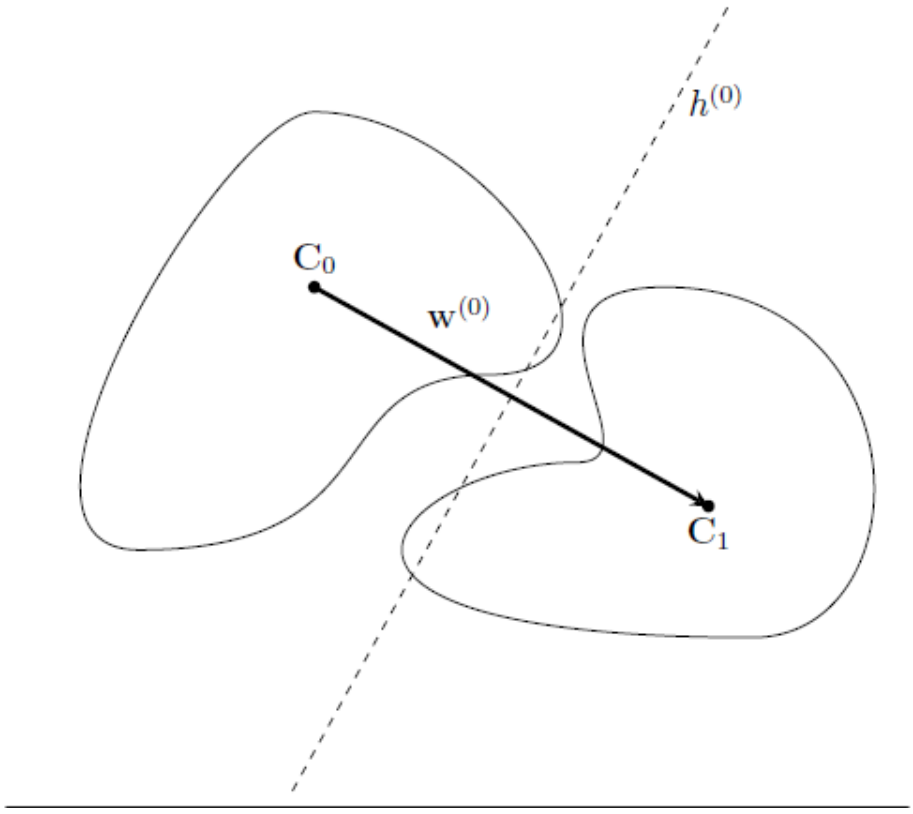

Figure 2.3: Initialization of weights

According to the centres (C0;C1) were computed for both classes of samples. The initial weight vector w(0) was then set to C1-C0. Finally, the bias w0 was set to C0+ C1-C02 which ensures that the initial separating hyperplane h(0) passes halfway between C0 and C1. In this setup, a substantial number of samples is already assigned to the correct class, even though a handful of samples are still classified incorrectly.

In Figure 2.3. As regards the initial bias w0, the authors suggest using the Scut method, originally developed by [12]. Roughly speaking, it means to go through the training samples one by one, shift the separating hyperplane to pass through the particular sample by choosing an appropriate bias value and compute the 'quality' of such a configuration (e.g. accuracy or F-measure). The best scoring bias value is then used as the actual bias in the initialization.

Apart from the initialization, the authors also suggest a modified stepwise learning rule for the adaptation of the weights, see Figure 2.4.

In the traditional approach, the output values are computed for each sample and the weights are subsequently updated in such a way that the separating hyperplane is rotated in the direction of the misclassified samples. The extent to which the weights are modified is co-determined by the parameter ", which has to be assessed experimentally. The modified variant is both similar and different. It also determines the incorrectly classified samples of both types—false positives (FP) and false negatives (FN)—using the current weights first. After that, the respective centers FP and FN are computed in the following fashion:

$$\mathbf{FP} = \frac{1}{|FP|} \sum_{x_k \in FP} x_k \tag{2.5}$$

$$\mathbf{FN} = \frac{1}{|FN|} \sum_{x_k \in FN} x_k \tag{2.6}$$

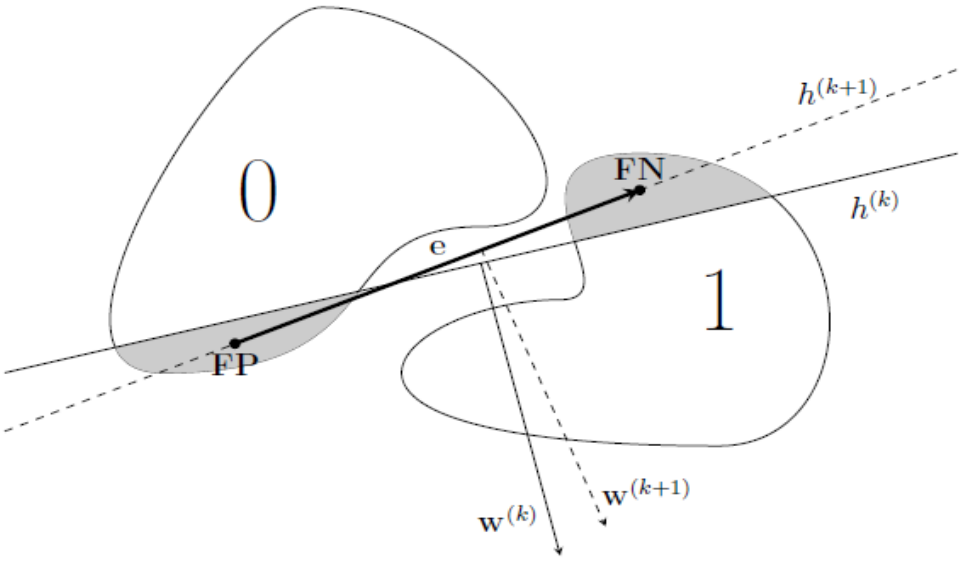

Figure 2.4: Illustration of one step in fast perceptron training

We are trying to find a hyperplane (in this case, line) which separates two classes of examples from each other. The separating line corresponding to the current weight vector w(k) is labeled as h(k). Apparently, a certain number of samples from the class 0 are incorrectly classified as 1 (“false positives” in the left grey area) and contrarily, other samples from the class 1 are misclassified as 0 (“false negatives” in the right grey area). The centers FP, resp. FN are

computed for all false positives, resp. false negatives, following which the error vector e can be obtained. Finally, the weights are adapted in such a way that both FP and FN lie on the new separating hyperplane h(k+1).

The major benefit of the new approach is that the value of " can be calculated precisely. It is done by adopting the assumption that a hyperplane passing through the points FP and FN will lead to an improvement in the classification rate. Since these points are the centers of the respective misclassified sets, it follows that about half of the previously misclassified samples will now be classified correctly. (Naturally, new erroneously classified samples may emerge instead.)

2.5 MULTI-LAYER FEED FORWARD NETWORKS

In the case of a single neuron unit described above, the output value was directly used as the final result indicating the class assigned to the input vector. An alternative option to make use of the output is to pass it as an input of another neuron instead. In this way it is possible to create a network of interconnected neurons, in certain aspects analogous to biological neural networks. Numerous different topologies have been used in practice for different purposes. One of the most widely used type of neural networks are the so-called multi-layer feedforward networks. Since these have been employed in part-of-speech tagging as well, they are of a particular importance for this thesis. It is therefore worth saying a few words about their structure, computation and mechanism of learning first.

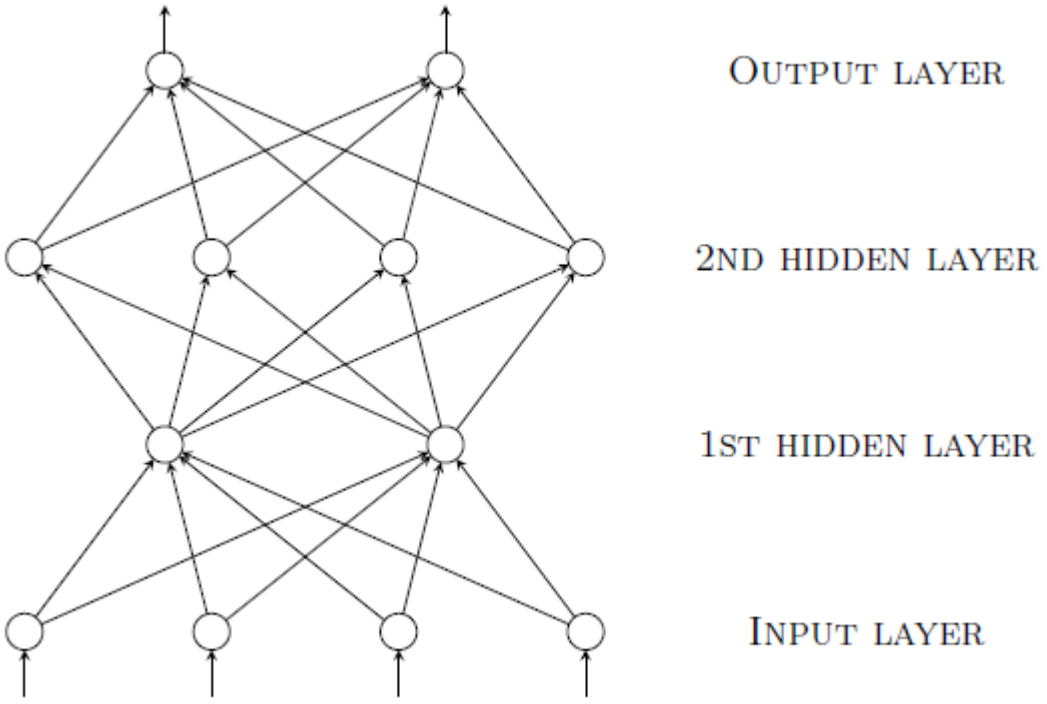

Figure 2.5: An example MLP network with two hidden layers

As the name hints, multi-layer feedforward networks, also known as multi-layer perceptrons (MLPs) consist of a certain number of neurons which are split into several disjoint sets, so called layers. Every MLP contains an input layer, to which the input vector is fed, and an output layer, which produces the vector of output values. In addition, the network may contain one or more hidden layers, located between the input and the output layer. Each neuron is connected to all the neurons in the adjacent layers: the outputs of the neurons from the previous layer serve as inputs and the output is passed to input of the neurons in the next layer.

As with a single perceptron unit, various activation functions can be used. For reasons that will be shown shortly, it is required that it be differentiable. As the threshold function cannot be used (it is discontinuous in 0), other functions of a similar shape are often employed instead.

These functions behave in an analogous way to the threshold function in that they return a value close to the maximum (1 in this case) in for sufficiently large inputs; similarly, a value close to the minimum (0, resp. -1) is returned for inputs low enough. Between these two extremes, the functions grow continuously, their growth rate being determined by the parameter _ (in the logistic sigmoid), resp. b (in the hyperbolic tangent).

2.6 BACK-PROPAGATION LEARNING ALGORITHM

Let us consider an arbitrary multilayer perceptron network. Let N be the set of all the neurons it consists of, IN N the set of the neurons in the input layer and OUT _ N the set of the neurons in the output layer. The individual neurons are indexed by numbers from 1 to jNj. The weight of the connection from a neuron i to a neuron j is denoted by wji, the set of all the weights by W. The notation i! refers to the set of all the neurons to which a connection from i exists, and contrarily, i is the set of all the neurons which are connected to the input of i.

Once all the output vectors yk have been obtained, they need to be compared to the desired output vectors dk to determine how well the network does and in what way the weights should be modified to improve the performance.

The more training samples generated incorrect outputs, the higher the value of the squared error function is, and vice versa. If the network produces correct outputs for all the samples, the squared error is zero. The problem of finding the best weights for a network is therefore equivalent to the problem of finding the global minimum of the function E. One way to find the global minimum is to compute the gradient $\nabla E(\mathbf{w}) = \left(\frac{\partial}{\partial w_{ji}}\right)_{w_{ji} \in W}$ in the point w, corresponding to the current vector of all the weights wji in the network. The gradient is a vector which determines the direction of the fastest growth of the error function in the specified point, as well as the steepness of the slope in the respective direction. If the gradient is subtracted from the current weight vector, we obtain a new vector which is likely to lie 'below' the original one in terms of the value of E. Therefore, the vector w is updated in the t-th iteration (t = 1; 2….)

3. MACHINE LEARNING ALGORITHMS

Machine learning is a subset of the Artificial Intelligence (AI) field which used to build a model or algorithm for specific purpose base on given data using special technique which includes programs and statistics computation.

The main type of ML:

a) Supervised learning.
b) Unsupervised learning.
c) Reinforcement learning.
d) Semi-supervised learning.

3.1 UNSUPERVISED LEARNING:

In this type of ML it contains data without a label.

Training set = {(X1), (X2), (X3), (Xn)}.

The main target is to understand the structure of that data given by algorithms.

Types of unsupervised learning: clustering, association.

3.2 REINFORCEMENT LEARNING

In this type of ML, the main goal is to maximize the reward action when the agent is performing the right path (increase the behavior), otherwise, decrease the behavior considered to result in a punishment. And the next step of the agent path will be according to the kind of reward of the previous step if it positive keep on the same path, otherwise change the path.

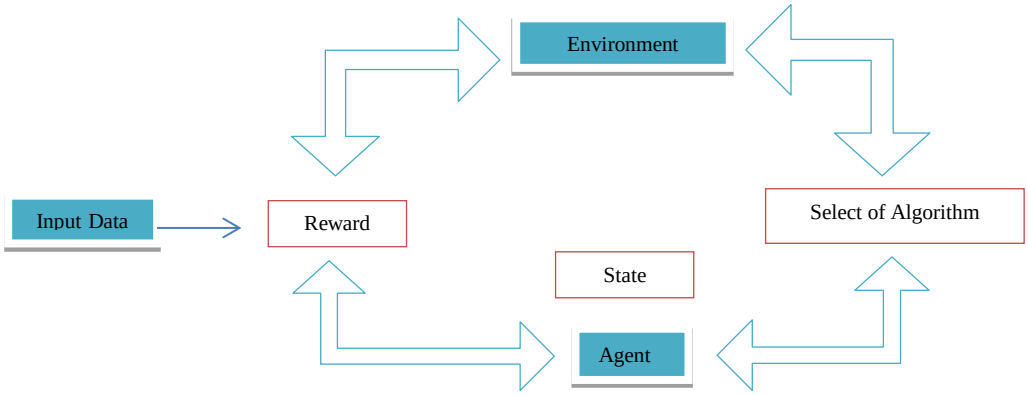

Figure 3.1 Reinforcement Learning

Types of Reinforcement Learning:

a) Positive Reinforcement.
b) Negative Reinforcement.

The main factor of RL:

a) Reward (R): denote to the feedback signal that indicated how well the step that the agent makes in a particular time.
b) Action (A): is the function of reward (R) and state (S).
c) State (S): that describes the environment.
d) Policy (P): the transfer from the environment to action is (P)
e) Value Function (V): a measurement tool to indicate how good the step is.
f) Model (M): is the demonstration of the agent's environment.

3.3 SEMI-SUPERVISED LEARNING:

The techniques which combine both of supervised learning and unsupervised learning to build a model which able to predict a large number of unlabeled data, with supervised learning which use a small number of label data at first time to train the model with a known target to build the model. While unsupervised learning is used by unlabeled data to the same model which trained before predicting this kind of data, this operation named semi-supervised learning.

This technique is used in web mining, text mining, and video mining. Where there are huge numbers of unlabeled data and a small number of label data.

The main reasons to use semi-supervised learning is the number of label data is less than the number of unlabeled data because obtain label data very expensive and difficult, so by using this method that will make use of a large number of unlabeled data as well as to increase the model accuracy.

3.4 SUPERVISED LEARNING:

It also called "learning with a master" in this type of ML contains data and label.

Training set = {(X1, Y1), (X2, Y2), (X3, Y3), (Xn, Yn)}.

The main target is to build a model0 that map X to Y label.

Type of supervised learning:

a) classification, regression :Classification is a supervised learning use to predict category using specific algorithms and label data, by takes data as input and map it to which group or category it is.

In binary classification each input image will classify to one of two classes (like predict whether the animal is a dog or cat or the mail is spam or non-spam) . In the other hand multi-classification the input image will classify to one of the number of classes (like classify MNIST handwritten digit, DR levels). Some samples of classification issue are speech recognition, handwriting recognition, biometric identification, image recognition, etc.

Most commonly algorithms which used for image classification are:

a) Support Vector Machines
b) Decision Tree
c) Feed-Forward neural network
d) Back-propagation network
e) Deep Learning

On my research, we will pre-trained deep learning convolutional neural network which is widely used and considered as a state-of-art for image classification, shows high accuracy and spend less time than other machine learning algorithms, using a lot of data and make a good performance despite a noise of data and require less image pre-processing.

3.4.1 Support Vector Machine

3.4.2 History

SVM was imagined by Vladimir N.Vapnik and Alexey Ya. Chervonenkis (Institute of Control Sciences of the Russian Academy of Sciences, Moscow, Russia) in the system of the "Summed up Portrait Method" for PC learning and example acknowledgment it was for straightly information isolated. The advancement of these thoughts began in 1962, and they were first distributed in 1964. In 1992 Vapnik et al. had the plan to apply what is known as the piece stunt which permits utilizing the SVM to characterize directly non-distinguishable information hard edge. The delicate edge is given by Vapnik et al. 1995 it is broadened the adaptation of hard edge SVM.

Hard edge SVM can work just when information is totally straightly distinguishable with no mistakes (commotion or exceptions). If there should be an occurrence of blunders either the edge is littler or hard edge SVM falls flat. Then again, delicate edge SVM was proposed by Vapnik to take care of this issue by presenting slack factors. SVM becomes popular because of its success in both regression and classification task:

a) Support Vector Machine (SVM): is used for classification.
b) Support Vector Regression (SVR): is used for regression.

But it is widely used in classification objectives.

3.4.3 Kernel function

Used to convert non-linear space into linear space so we can use a linear model to separate non-linear separated data and reduce the computation cost, it takes the inner product of the new vectors.

Let we have binary classification with x input and single feature (non-linear separated data).

By map the input example to new representation or too high dimensionality we can use a linear model to classify the non- linear dataset.

$$\boldsymbol{g(x) = w^t \emptyset(x) + b} \tag{3.1}$$

$$\boldsymbol{g(x) = \sum_{i \in sv} \propto_i \emptyset(x_i)^t \emptyset(x) + b} \tag{3.2}$$

$$\boldsymbol{K(x_a, x_b) = \emptyset(x_a)\, \emptyset(x_b)} \tag{3.3}$$

Where:

K is kernel function.

$\emptyset$ Is mapping from X to an (inner product) feature space.

3.4.4 Kernel trick:

It implies the bit work change the information into a higher dimensional component space to make it conceivable to play out the straight division, which has the ability to compute inner product of defines space without visit it and make the number of dimensions depends on the number of examples, not on the dimension of space that defines.

Let y = $\{y_1, y_2\}$ with two feature space and it's non-linear separable.

The inner product is a function between the transformation of x and x':

$$z^t z' = K(x, x') \boldsymbol{the\ kernel} \tag{3.4}$$

Let

y = $\emptyset(x)$

y = $\emptyset(x)$

$$l(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{M=1}^{m} z_n z_m \alpha_n \alpha_m \boldsymbol{y_n^t\, y_m} \tag{3.5}$$

Conditions: $\alpha_n \geq 0$ for n = 1, 2… N and $\sum_{n=1}^{N} \propto_n y_n = 0$

$$g(x) = \sin(w^t.z + b)$$

We need $\boldsymbol{y_n^t y}$

$$b = z_m(w^t y_m + b) = 1$$

Use the original method for classification without using kernel function:

By using transformation function will convert from non-linear separable to linear separable as follow:

$\Phi(y) \rightarrow \quad y_1^2, y_2^2, \sqrt{2y_1}y_2$

Where: $\Phi(y)$ is a transformation function convert from 2- dimension to 3-dimension.

By using the decision boundary in 3-dimension space:

$$\beta_0 + \beta_1 y_1^2 + \beta_3\sqrt{2y}y_2 = 0 \tag{3.6}$$

For i point

$$\Phi(y_i) = (y_{i1}, y_{i2})$$
$$= (y_{i1}^2, y_{i2}^2, \sqrt{2y_{i1}}y_{i2})$$

here we use 4 operations for i point.

On the other hand, we have j point:

$$\Phi(y_j) = (y_{j1}, y_{j2})$$
$$= (y_{j1}^2, y_{j2}^2, \sqrt{2y_{j1}}y_{j2})$$

Again we use 4 operations for j point.

Finally to compute the dot product between the vector (similarity measure)

$$(\Phi(yi),\Phi(yj)) = y_{i1}^2 y_{j1}^2 + y_{i2}^2 y_{j2}^2 + 2y_{i1}y_{i2}2y_{j1}y_{j2} \tag{3.7}$$

Here we use 3 operations of the product and 2 additional operations.

Total number of operation using original method is: 4+4+3+2 = 13 operations

Use the kernel trick method.

$$(y_i, y_j)^2 = (\{y_{i1}, y_{i2}\}, \{y_{j1}, y_{j2}\})^2 \tag{3.8}$$
$$= (y_{i1}y_{i2} + y_{j1}y_{j2})^2 \tag{3.9}$$
$$= y_{i1}^2 y_{j1}^2 + y_{i2}^2 y_{j2}^2 + 2y_{i1}y_{i2}2y_{j1}y_{j2} \tag{3.10}$$

The number of operation used by the kernel as follow:

a) Used 3 multiplication operations (two for dot product & one for square operation).
b) Used 1 additional operation

The total number is 3+1 = 4 operation used by kernel less than the original method that mapping data from 2-dimension space to 3-dimension space then apply the required operations while by using kernel trick is about to stay in same 2-dimension space and compute the same result as in 3-dimensional space.

3.4.5 Kernel Types

a) Polynomial
b) Gaussian.
c) Gaussian Radial Basis Function(RBF)
d) Laplace RBF
e) Hyperbolic tangent
f) Sigmoid
g) Bessel function of the first kernel
h) Anova radial basis
i) Linear splines kernel in one-dimension.

3.5 FEED FORWARD NEURAL NETWORK:

3.5.1 History

FFNN has been around for a long time, the idea of model neural network of human brain was begin in 1943 by WARREN S.McCULLOCH and WALTER PITTS by define threshold logic and introduced neural network model, it was kind of trying to simulate the human brain, this lead to understanding to structure of brain especially on external cortex of the brain which consists of huge number of neurons that connection between each other in parallel distributed process so as to speed up and robust learning process.

Using weights that need to be tuned by human interaction manually.

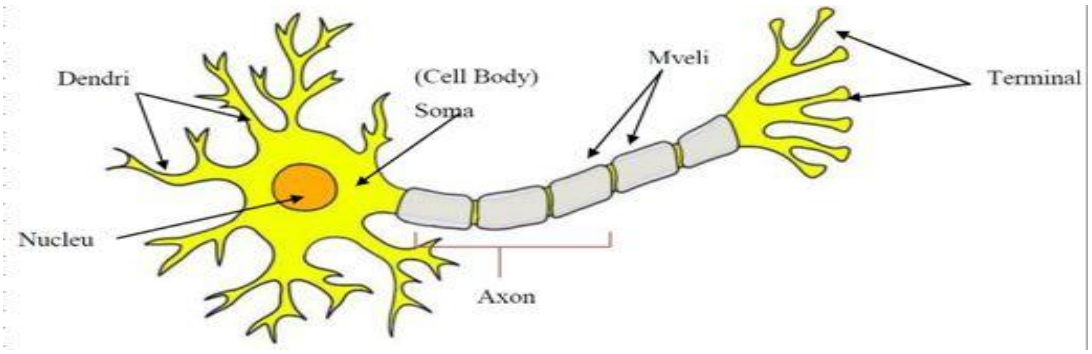

Figure 3.2 Single Neuron of Human Brain

As shown in Figure 2.2 every single neuron in the human brain has three operations[33]:

a) input operation using synapse (receive signals from another neuron)
b) Excitation or damping collected signals using the cell body (soma) depends on the chemistry of the cell body.
c) Output operation using Axon send the final signal to another neuron.

So each of the neurons is fired (On or off) with a non-linear function, where the output of neuron will be the input to another neuron. In the human brain, around 100 billion neurons that connect to each other by 100 trillion connections and transmission rate of the synapse is around 100 bits per second.

On 1949 Donald Hebb defines the first learning methods formulated named “Hebbian Learning” introduce the idea of "relationship learning" this is the possibility that the heaviness of association is balanced dependent on the estimations of the neurons it interfaces:

$$\Delta w_{ij} = \alpha a_i a_j \tag{3.11}$$

On 1959 Rosenblat introduce perceptron algorithm with one layer that solves linear classification problems (binary image classification), as shown in **Figure 2.3**.

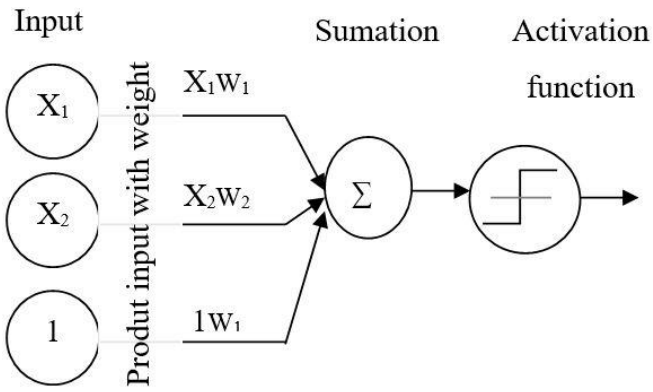

Figure 3.3 Perceptron Architecture

The limitation of perceptron algorithm it could not solve non-linear classification (XOR) problem so people like Marvin Minsky was skeptical of neuron nets for a long time when he declares the limitation of perceptron algorithm for solving the non-linear problem this was in 1969. That coincided with an AL winter, neural network had a revival on 1974 by PAUL J. WERBOS define Backpropagation algorithm with three-layer perceptron network. On 80s decade the interest to

Backpropagation algorithm is increase and developed to Backpropagation algorithm with MLP (multi-layer perceptron) by Rummelhart and Mclelland which can deal with non-linearity problem and consist of three main components (input layer, hidden layers, output layers), and open the new window to more research and development for neural network field and an ability to solve many problems which were difficult to solve.

3.5.2 Activation Function:

It translates the input signals to output signals, after computes the weight summation and add the bias and pass it to activation function which control the output and manage the fire operation (on, off) of the nodes in the specific layer (hidden layer, output layer) where

$$\text{Output} = ActivationFunction \sum [(weight * input) + bias] \tag{3.12}$$

If the output more than the threshold then the output will pass and be as input to another neuron otherwise the will not. By choosing appropriate activation function on the neural network that will effect on the result and the performance of the network.

Two main type of activation function:-

3.5.3 Linear activation (identity) function:

It has simple structure:

$$Z = cx$$

It receipts the inputs then multiplied it by the weights for each neuron, and generate an output signal comparable to the input.

The main drawback of linear activation function:

a) With constant derivative it cannot be use in BP (GD).
b) Whatever the numbers of hidden layers, with linear activation function it will be as one layer and the final output of a network is still a linear function of the input in spite of the number of the hidden layers that was the limitation of using the linear function on a hidden layer because it does not allow us to have many hidden Layer as we use in deep learning.

Also, it performs a limitation with high-dimension of data and various type (images, audio, speech, etc...) also a limitation with big data.

The main use of this type of activation function is at the output layer for the classification task to separate the data into classes, while using a non- linear activation function in the hidden layer.

3.5.4 Non-linear activation function:

The factor that make ANNs and BP develop and widely used in many research solving large numbers of complicated problems is the non-linearity activation function.

The main feature of this type of activation function:

a) It is derivative that will allow to the network to be learn from the weights.
b) Allow to use many hidden layers and use with a different type of data and can perform with big data also allow us to learn non-linear problems.

Most popular no-linear activation function:-

Sigmoid (logistic) function:

$$f(x) = \frac{1}{1 + e^{x}} \tag{3.13}$$

With curve shape like "S" letter, the output of this function change continuously but not change linearly and we can observe that the value of output is between (0, 1) . as the information is bit by bit changed from negative to positive endlessness. So it utilizes when the yield is relied upon to be a positive number (MLP, Backpropagation calculations) and is viewed as a sensible guess of genuine neurons.

By using sigmoid it will be bad choice because of the vanishing gradient issue which happens when the function directs the input to small range [0, 1] of output so any change on the input parameter even it big the result is very small change on the value of the output and the gradient will be very small which cause vanishing gradient obstacle [34]

Hyperbolic Tangent (Tanh) function:

It commonly used activation function, as it works with both negative and positive number.so the output is range between [-1, +1] .the equation is:

$$f(x) = \frac{1 - e^{-2x}}{1 + 3e^{-2x}} \tag{2.13}$$

And it also has a vanishing gradient manner.

Rectified Linear Unit (ReLU):

Is a familiar activation function because of it simple and has good effect where it gets rid of vanishing gradients, and used in hidden layers but the week point here is dead neurons.

In 2017 Shumin Kong et al. demonstrate that "ReLU edges all negative an incentive to zero (f(x) = max (0, x)), its positive part has a fixed angle of 1. Henceforth, ReLU won't soak at the positive part. In any case, on the negative part, the angle of ReLU concerning the information is zero, which implies once ReLU neuron produces negative yield, the slope stream into the neurons will consistently be zero. The heaviness of the neuron would in this way never be enhanced, because of zero inclination." [2 p.2563]

$$f(x) = \begin{Bmatrix} 0\,, x < 0 \\ x\,, x \geq 0 \end{Bmatrix} \tag{3.14}$$

EXPONENTIAL LINEAR UNIT (ELU):

This function overcomes the dead neurons problem by using exponential process so it is better than ReLU function. This function is used in hidden layers.

$$f(x) = \begin{Bmatrix} \propto (e^x - 1), x < 0 \\ x\,, x \geq 0 \end{Bmatrix} \tag{3.15}$$

Softmax Function:

this function is kind of sigmoid function with range between [0, 1], and the main difference is that sigmoid function was used to classify two classes while Softmax use to classify more than two classes (multiclass) and it placed on the final layer(O/P layer) that turn logits which is the vector of numbers to a probability for each class, higher probability could indicate the predicted class where the sum of these probability is equal to one.

$$(\sigma)(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}} \tag{3.16}$$

3.5.5 Feedforward neural network overview

A Feedforward neural network is a non-recurrent range which contains information sources, yields, and shrouded layers; the signs can just go one way. What's more, the information is passed onto a layer of handling components where it performs the computations by weighted summation of its inputs and add the bais value, after that apply activation function where the new calculated output value becomes the new information esteem that takes care of the following layer, this procedure proceeds until it has experienced all the layers and decides the yield. A limit move work is now and again used to measure the yield of a neuron in the yield layer.

4. EVALUATION

4.1 DATASETS USED

We use three datasets in total for evaluating the models. Several ar- rhythmia public datasets exist, but their structure is variable. In par- ticular, they are varied by the lengths of recordings, the number of classes and number of ECG leads. From these, we have chosen The PhysioNet Computing in Cardiology Challenge 2017 and The China Physiological Signal Challenge 2018 dataset. For an interested reader, comprehensive lists of arrhythmia datasets is a part of the PTB-XL publication [36]. We also used a private dataset which is larger than any open one we have found.

4.1.1 The PhysioNet Computing in Cardiology Challenge 2017

AF Classification from a short single lead ECG recording was the theme of The PhysioNet Computing in Cardiology Challenge (CINC) 2017 [2]. It is of our interest since the samples provided resemble our use case – single lead data from a telemonitoring device. The 8528 recordings ranging from 9 to 61 seconds are sampled at 300 Hz. The files are already preprocessed by a band-pass filter with a bandwidth of 0.5- 40 Hz. There are only three classes included in the challenge original scoring metric – the noise class *X* was added subsequently. Five teams shared the first place in the competition. The winners mainly used ensemble methods: XGBoost + RNN [37], random forests [38, 39] and custom ensemble algorithms [40, 41].

Table 4.1: CINC2017 labels

Label	Condition	Count
N	Sinus Rhythm	5154
A	Atrial Fibrillation	771
O	Other Rhythm	2557
X	Noisy	46
Total		8528

4.1.2 The China Physiological Signal Challenge 2018

A set of 12-lead ECGs was made public for The China Physiological Signal Challenge (CPSC) 2018 [42]. There is a total of 9831 recordings from 9458 different patients. Out of those, 6877 is available in the form of a training set. The recordings are sampled at 500 Hz. The length of the recording ranges from six to 60 seconds. There is a total of nine distinct classes. There is a total of 477 multi-label recordings (the patient had more than one heart condition); we do not use them as some of our models are not fit for multi-label classification. We used only the first lead I as it is similar to a single-lead ECG. Moreover, the winners of the challenge achieved only slightly worse performance when using a single lead for classification [43].

Table 4.2: CPSC2018 labels

Label	Condition	Count
Normal	Sinus Rhythm	918
AF	Atrial Fibrillation	1098
I-AVB	Atrioventricular Block	704
LBBB	Left Bundle Branch Block	207
RBBB	Right Bundle Branch Block	1695
PAC	Premature Atrial Contraction	556
PVC	Premature Ventricular Contraction	672
STD	ST Segment Depression	825
STE	ST Segment Elevation	202
Total		6877

4.2 TRAINING

This chapter is concerned about the training of the models. The models are trained on each dataset separately as there is little overlap between the labels in the datasets.
The precise model hyperparameters and architectures are also available in the source code.

4.2.1 Train-test Split

We split all the datasets into train and test partitions which are same for all the models. The models are trained on the train partition, and the final evaluation is done on the test partition, which the model has not seen so that we can evaluate the generalization power of the model. For neural network-based models, we further separate a hold-out validation set out of the training set to evaluate learning performance during training.
The split is 90–10 % of data for the train and test partitions, respec- tively. When using the validation partition, the final split is 80–10–10 % of train, validation and test partitions. All

the models are trained and evaluated on the same train-test split.
Because the private datasets contain a large number of recordings from the same patients, the train-validation-test partitions are split using GroupShuffleSplit from scikit-learn by the telemonitoring device ID. Therefore, it should not happen that the same patient's record is both in the training and test partition.

4.2.2 Cross-validation

Cross-validation is an alternative way of evaluating the generalization performance of a model. The *K*-fold cross-validation splits a dataset into *K* partitions (folds), then trains the model *K´* times training on *K* 1 of the folds and evaluating on the last one. The *K* scores can be averaged to get the validation score.

For all models except CNN and RNN models, we use 5-fold cross-validation as the validation score. For CNN and RNN, we use the hold-out validation set score.

4.2.3 Models with hand-engineered Features

This family of models is trained using the 5-fold cross-validation on 90 % of the data. All the models use balanced class weights – the fewer samples the class has, the larger the class weight is. The parameters used were found by a cross-validated grid search on the training partition of the CINC2017 dataset. If a parameter of the classifier is not stated, we use the default value provided by the library.

Logistic regression

The missing features are imputed by the mean of the column and then standardized (Section 2.2.3). The regularization parameter $C = 100.0$ is used.

Random forest

The missing features were imputed but not scaled, as random forests can deal with unscaled features easily.

The max_depth of the trees in the forest is set to 20, the minimum number of samples at a leaf node (min_samples_leaf) is 5, and the number of estimators in the forest is 1000.

XGBoost

We use a learning rate eta of 0.1, a minimum loss reduction (gamma) of 0.1. The maximum allowed depth (max_depth) of a tree is 7. The min_child_weight parameter is 4, number of estimators is 1000, the same as in the random forest classifier. The subsample parameter is set to 0.8.

MLP

The default learning rate of 0.001 with the Adam optimizer [64] is used for minimizing the cross-entropy loss function. A batch size of 128 is used.
Early stopping is used – when the validation score is not improving for ten consecutive epochs, the training stops. Early stopping can help the generalization power of the neural network [65].

4.3 MODELS WITH DIRECT SIGNAL INPUT

4.3.1 Data Augmentation

Data augmentation techniques help us to increase the model's training dataset size by introducing transforms that do not change the label of the recording. We used random flipping of the sign of the signal, as it commonly happens that the patient switches the electrodes, and the resulting signal is inverted.
Random cropping of the signal would be possible if we had the information about the location of the arrhythmia. As the information is not available, random cropping could crop out the arrhythmic part, and the label would be wrong. Thus, we decided not to use random cropping.

ResNet

We use a ResNet implementation from the torchvision library used for computer vision tasks. The convolutional layers are rewritten to be one-dimensional. We use an 18-layer

and 50-layer convolutional residual networks. We train the ResNets for 20 epochs and after each epoch measure the validation score. We take the model with the best validation score as the final trained model.

We use a batch size of 32 for the ResNet18 model. The batch size is dependent mainly on the available CUDA memory. We use a cross-entropy loss function, the Adam optimizer with a learning rate of 0.0003. For regularization of the networks, we use a weight decay of 0.0001. We do not use weight decay for the batch normalization parameters, as it is shown to degrade the performance of learning [66]. We use gradient norm clipping [67], which helps us with unstable gradients problem.

CnnGru

For training the CnnGru classifier, we use learning hyperparameters same as the ResNet ones – Adam, a learning rate of 0.0003, weight decay of 0.0001, cross-entropy loss. We train with a batch size of 32 for ten epochs and again save the model trained on the epoch with the highest validation score.

k-NN with DTW

The n_neighbors parameter is set to 1, as we want to use the 1-nearest neighbour classifier. However, the evaluation takes too much time to calculate scores on even the smallest of our datasets. Therefore, we do not have the scores for this baseline. For our use case, the inference time is crucial, so this classifier is apparently not applicable.

4.4 MODEL PERSISTENCE

To save the models for future evaluation, we used the PyTorch and joblib libraries, which both use the pickle module under the hood. This solution is not perfect because the saved models have no guarantee of being transferable to the future versions of the machine learning libraries. Our mitigation is to use the reproducible Python virtual environment which has exactly the versions of libraries we used. Also, the exported ONNX models are independent of any library and should be usable on any platform supporting the ONNX Runtime.

After training of the models, we compare them against each other. In this chapter, we will describe the F_1 score metric, which we use in the final comparison table and then discuss the final results.

4.5 METRICS

We need to evaluate the performance different models on the same data. For a binary classification problem with two classes, say *Pos* and *Neg*, a confusion matrix has two rows and two columns. True positives are the examples labelled as *Pos* and classified as *Pos*, true negatives are *Neg* classified as *Neg*, false positives are *Neg* classified as *Pos*, and false negatives are *Pos* classified as *Neg*.

Table 4.3: A binary classification task confusion matrix.

True Positives (TP)	True Negatives(TN)
False Positives (FP)	False Negatives (FN)

A confusion matrix for a multi-class problem has shape *N N*, where *N* is the number of classes. A single score instead of a full matrix is preferable for comparing the classifiers' performance. We calculate the F_1 score for all classifiers and use it for a rough comparison.

4.5.1 F1 Score

The F_1 score is the harmonic mean of precision (positive predictive value) and recall (sensitivity)[68]:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{4.1}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \text{ and recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{4.2}$$

The above definition is valid for binary classification. With multi- class classification, we do macro averaging. That means, we take separate F_1 score for each class versus all

the others as a binary prob- lem; then, we get the resulting macro F_1 score as the mean of all the class scores.

F_1 score is a simple way to compare classifiers. It favours classifiers with similar precision and recall; this is not always what we want when our task is classifying heart pathologies. We use F_1 score for simplicity, but some types of arrhythmia are more dangerous than the others. For example, ventricular tachycardia is life-threatening as it can lead to deadly ventricular fibrillation. Therefore, a metric that takes into account the severity of different classes could prove useful in future work.

4.6 RESULTS

In Table 4.3 and figures 4.1 and 4, we can see the F_1 scores, macro averaged over all classes of the datasets. Our best trained model on the CINC2017 dataset is the MLPClassifier, the best model on the CPSC2018 dataset is the ResNet18Classifier and on the private dataset it is the CnnGruClassifer On the CINC2017 dataset with four classes and thousands of recordings, the methods based on hand-engineered features and on the direct signal are comparable. However, when the dataset has more different classes or is bigger in size, the automated feature extractors can provide better performance and good inference time.

As the private dataset is the most similar to the telemonitoring data, we will likely use the methods trained on the direct signal. The hand-engineered feature methods could probably be tuned and have better results with more domain knowledge and expert-designed features.

Table 4.4: The F1 macro averaged scores on the test partitions.

Model	CINC2017	CPSC2018
LogisticRegression	0.7419	0.5532
RandomForestClassifier	0.7513	0.5323
XGBClassifier	0.7480	0.5824
MLPClassifier	**0.7824**	0.5840
ResNet18Classifier	0.7539	**0.7296**
CnnGruClassifier	0.7470	0.6432

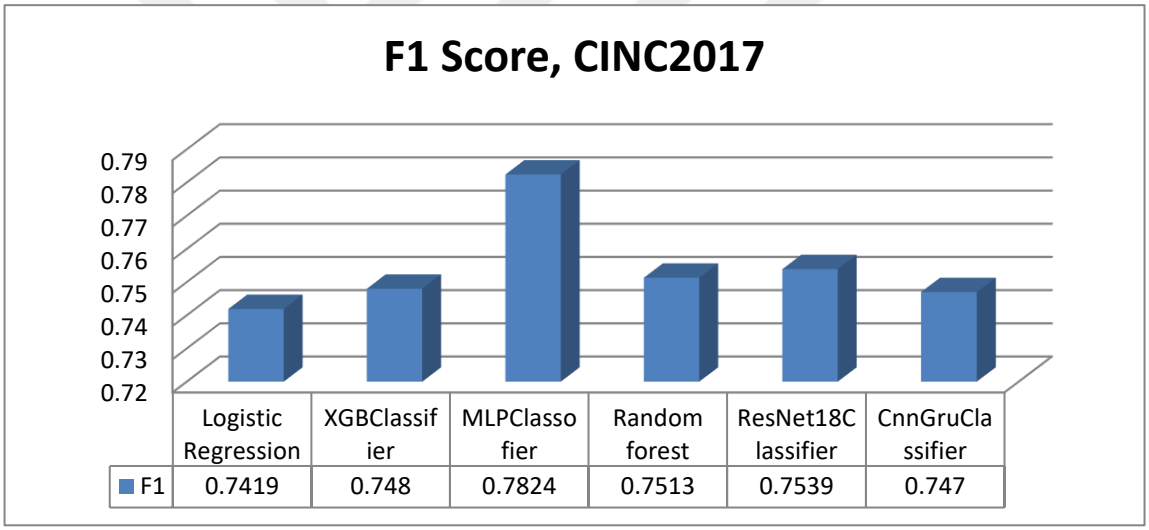

Figure 4.1: The F1 macro averaged scores on the test partitions, CINC2017 dataset

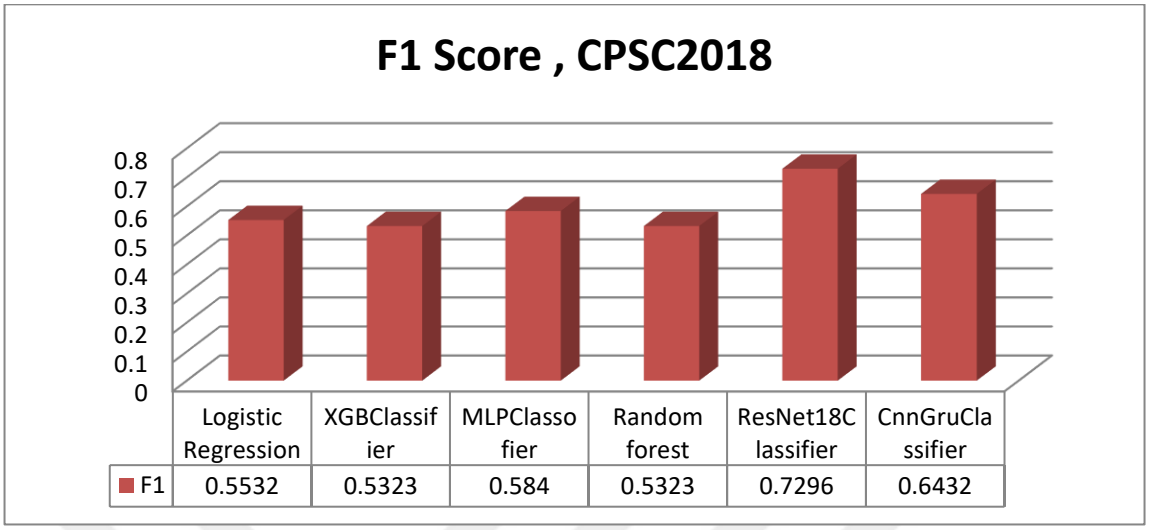

Figure 4.2: The F1 macro averaged scores on the test partitions, CPSC2018

5.CONCLUSION

The EEG signals have a high potential of uncovering many neurological disorders in an early stage. We have shown why it is better than MRI. Being cheap and efficient, it can be used for many researches even with a lower budget as compared to MRI. Many Deep Learning algorithms have been applied on EEG signals in order to classify various neurological disorders and brain interface applications. In this paper we have done an extensive literature survey about application of different neural networks on EEG signals and how using these neural networks one was able to classify several neurological diseases like Seizure, AD and Depression. The preprocessing step greatly affects the performance of the model. In this work, we train seven different models on three different ECG databases and compare their performance between each other. We find that at least in our experiment, both hand-crafted features and the neural network feature extractors each have both its positive and negative aspects.

The next step is the integration of the model to the ECG processing solution running on the telemonitoring server.

REFERENCES

[1]. LILLY, Leonard S. Chapter 11. Mechanisms of Cardiac Arrhyth- mias. In: *Pathophysiology of heart disease: a collaborative project of medical students and faculty*. Wolters Kluwer, 2016, p. 268. ISBN 1605477230.

[2]. CLIFFORD, Gari; LIU, Chengyu; MOODY, Benjamin; LEHMAN, Li-wei; SILVA, Ikaro; LI, Qiao; JOHNSON, Alistair; MARK, Roger. AF Classification from a Short Single Lead ECG Recording: the Physionet Computing in Cardiology Challenge 2017. In: *2017 Computing in Cardiology Conference (CinC)*. Computing in Cardiology, 2017. Available from DOI: 10.22489/cinc.2017.065-469.

[3]. BLAUS, Bruce. *An illustration depicting a Holter monitor.* 2017. Available also from: https://commons.wikimedia.org/wiki/File:Holter_Monitor.png.

[4]. MEYSTRE, Stephane. The Current State of Telemonitoring: A Comment on the Literature. *Telemedicine and e-Health*. 2005, vol. 11, no. 1, pp. 63–69. Available from DOI: 10.1089/tmj.2005.11.63.

[5]. CORDISCO, Marie Elena; BENIAMINOVITZ, Ainat; HAMMOND, Kim; MANCINI, Donna. Use of telemonitoring to decrease the rate of hospitalization in patients with severe congestive heart failure. *The American Journal of Cardiology*. 1999, vol. 84, no. 7, pp. 860–862. Available from DOI: 10.1016/s0002-9149(99)00452-x.

[6]. ATKIELSKI, Anthony. *ECG of a heart in normal sinus rhythm.* 2007. Available also from: https://commons.wikimedia.org/wiki/File:SinusRhythmLabels.svg.

[7]. DUBIN, Dale. Chapter 1: Basic Principles. In: *Rapid Interpretation of EKG's, Sixth Edition*. Tampa, Fla: Cover Pub. Co, 2000, pp. 14–27. ISBN 0912912065.

[8]. LIP, Gregory Y. H. et al. Atrial fibrillation. *Nature Reviews Disease Primers*. 2016, vol. 2, no. 1, pp. 16016. ISSN 2056-676X. Available from DOI: 10.1038/nrdp.2016.16.

[9]. ODUTAYO, Ayodele; WONG, Christopher X; HSIAO, Allan J; HOPEWELL, Sally;

ALTMAN, Douglas G; EMDIN, Connor A. Atrial fibrillation and risks of cardiovascular disease, renal dis- ease, and death: systematic review and meta-analysis. *BMJ*. 2016, pp. i4482. Available from DOI: 10.1136/bmj.i4482.

[10]. DUBIN, Dale. Chapter 5: Rhythm, Part I. Premature Ventricular Contraction (PVC). In: *Rapid Interpretation of EKG's, Sixth Edition*. Tampa, Fla: Cover Pub. Co, 2000, p. 135. ISBN 0912912065.

[11]. DUBIN, Dale. Chapter 5: Rhythm, Part I. Runs of Ventricular Tachycardia. In: *Rapid Interpretation of EKG's, Sixth Edition*. Tampa, Fla: Cover Pub. Co, 2000, p. 156. ISBN 0912912065.

[12]. NOLLE, Floyd M; BOWSER, Richard W. *Creighton University Ven- tricular Tachyarrhythmia Database*. physionet.org, 1992. Available from DOI: 10.13026/C2X59M.

[13]. OPPENHEIM, Alan. *Discrete-time signal processing*. Upper Saddle River: Pearson, 2010. ISBN 0131988425.

[14]. OPPENHEIM, Alan. Chapter 8: The Discrete Fourier Transform. In: *Discrete-time signal processing*. Upper Saddle River: Pearson, 2010, pp. 541–600. ISBN 0131988425.

[15]. OPPENHEIM, Alan. Chapter 7: Filter Design Techniques. In: *Discrete-time signal processing*. Upper Saddle River: Pearson, 2010, pp. 439–511. ISBN 0131988425.

[16]. KWON, Ohhwan; JEONG, Jinwoo; KIM, Hyung Bin; KWON, In Ho; PARK, Song Yi; KIM, Ji Eun; CHOI, Yuri. Electrocardiogram Sampling Frequency Range Acceptable for Heart Rate Variability Analysis. *Healthcare Informatics Research*. 2018, vol. 24, no. 3, pp. 198. Available from DOI: 10.4258/hir.2018.24.3.198.

[17]. *scipy.signal.resample_poly — SciPy v1.5.0 Reference Guide*. 2020. Available also from: https://docs.scipy.org/doc/scipy-1.5. 0/reference/generated/scipy.signal.resample_poly.html.

[18]. GAO, Hongxiang; LIU, Chengyu; WANG, Xingyao; ZHAO, Lina; SHEN, Qin; NG, E. Y. K.; LI, Jianqing. An Open-Access ECG Database for Algorithm Evaluation of QRS Detection and Heart Rate Estimation. *Journal of Medical Imaging and Health Informatics*. 2019, vol. 9, no. 9, pp. 1853–1858. Available from DOI: 10.1166/ jmihi.2019.2800.

[19]. PAN, Jiapu; TOMPKINS, Willis J. A Real-Time QRS Detection Algorithm. *IEEE Transactions*

on Biomedical Engineering. 1985, vol. BME-32, no. 3, pp. 230–236. Available from DOI: 10.1109/
tbme.1985.325532.

[20]. PLESINGER, Filip; ANDRLA, Petr; VISCOR, Ivo; HALAMEK, Josef; BULKOVA, Veronika; JURAK, Pavel. Shape Analysis of Consecutive Beats May Help in the Automated Detection of Atrial Fibrillation. In: *2018 Computing in Cardiology Conference (CinC)*. Computing in Cardiology, 2018. Available from DOI: 10.22489/ cinc.2018.036.

[21]. DUBIN, Dale. Chapter 6: Rhythm, Part II. 1°AV Block. In: *Rapid Interpretation of EKG's, Sixth Edition*. Tampa, Fla: Cover Pub. Co, 2000, p. 178. ISBN 0912912065.

[22]. ACHARYA, U. Rajendra; JOSEPH, K. Paul; KANNATHAL, N.; LIM, Choo Min; SURI, Jasjit S. Heart rate variability: a review.
Medical & Biological Engineering & Computing. 2006, vol. 44, no. 12, pp. 1031–1051. Available from DOI: 10.1007/s11517-006-
0119-0.

[23]. *Heart Rate Variability (HRV) — NeuroKit 0.0.39 documentation*. 2020. Available also from: https://neurokit2.readthedocs.io/en/latest/examples/hrv.html.

[24]. VIRTANEN, Pauli et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020, vol. 17, pp. 261–272. Available from DOI: https://doi.org/10.1038/
s41592-019-0686-2.

[25]. MAKOWSKI, Dominique; PHAM, Tam; LAU, Zen J.; BRAM- MER, Jan C.; LESPINASSE, François; PHAM, Hung; SCHÖLZEL, Christopher; S H CHEN, Annabel. *NeuroKit2: A Python Toolbox*
for Neurophysiological Signal Processing. Zenodo, 2020. Available from DOI: 10.5281/ZENODO.3597887.

[26]. PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python.
Journal of Machine Learning Research. 2011, vol. 12, pp. 2825–2830.

[27]. PASZKE, Adam et al. PyTorch: An Imperative Style, High Perfor- mance Deep Learning Library. In: WALLACH, H.; LAROCHELLE, H.; BEYGELZIMER, A.; D'ALCHÉ-BUC, F.; FOX, E.; GARNETT,

R. (eds.). *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[28]. BAI, Junjie; LU, Fang; ZHANG, Ke, et al. *ONNX: Open Neural Network Exchange* [https://github.com/onnx/onnx]. GitHub, 2019.

[29]. CORPORATION, Microsoft. *ONNXMLTools*. 2020. Available also from: https://github.com/onnx/onnxmltools.

[30]. REBACK, Jeff et al. *pandas-dev/pandas: Pandas 1.0.5*. Zenodo, 2020. Available from DOI: 10.5281/ZENODO.3898987.

[31]. COSTA-LUIS, Casper O. da. tqdm: A Fast, Extensible Progress Meter for Python and CLI. *Journal of Open Source Software*. 2019, vol. 4, no. 37, pp. 1277. Available from DOI: 10.21105/joss.
01277.

[32]. WALT, Stéfan van der; COLBERT, S Chris; VAROQUAUX, Gaël. The NumPy Array: A Structure for Efficient Numerical Compu- tation. *Computing in Science & Engineering*. 2011, vol. 13, no. 2, pp.
22–30. Available from DOI: 10.1109/mcse.2011.37.

[33]. TAVENARD, Romain et al. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*. 2020, vol. 21, no. 118, pp. 1–6. Available also from: http://jmlr.org/
papers/v21/20-091.html.

[34]. CHEN, Tianqi; GUESTRIN, Carlos. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD Inter- national Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA: ACM, 2016, pp. 785–794. KDD '16.
ISBN 978-1-4503-4232-2. Available from DOI: 10.1145/2939672. 2939785.

[35]. INC., Plotly Technologies. *Collaborative data science*. Montreal, QC: Plotly Technologies Inc., 2015. Available also from: https:
//plot.ly.

[36]. WAGNER, Patrick; STRODTHOFF, Nils; BOUSSELJOT, Ralf-Dieter; KREISELER, Dieter; LUNZE, Fatima I.; SAMEK, Wojciech; SCHA- EFFTER, Tobias. PTB-XL, a large publicly available electrocardio- graphy dataset. *Scientific Data*. 2020, vol. 7, no. 1. Available from

DOI: 10.1038/s41597-020-0495-6.

[37]. TEIJEIRO, Tomas; GARCIA, Constantino A.; CASTRO, Daniel; FÉLIX, Paulo. Arrhythmia Classification from the Abductive In- terpretation of Short Single-Lead ECG Records. In: *2017 Comput- ing in Cardiology Conference (CinC)*. Computing in Cardiology, 2017. Available from DOI: 10.22489/cinc.2017.166-054.

[38]. ZABIHI, Morteza; RAD, Ali Bahrami; KATSAGGELOS, Agge- los K.; KIRANYAZ, Serkan; NARKILAHTI, Susanna; GABBOUJ, Moncef. Detection of Atrial Fibrillation in ECG Hand- held De- vices Using a Random Forest Classifier. In: *2017 Computing in Cardiology Conference (CinC)*. Computing in Cardiology, 2017. Available from DOI: 10.22489/cinc.2017.069-336.

[39]. MAHAJAN, Ruhi; KAMALESWARAN, Rishikesan; HOWE, John Andrew; AKBILGIC, Oguz. Cardiac Rhythm Classification from a Short Single Lead ECG Recording via Random Forest. In: *2017 Computing in Cardiology Conference (CinC)*. Computing in Cardi- ology, 2017. Available from DOI: 10.22489/cinc.2017.179-403.

[40]. DATTA, Shreyasi et al. Identifying Normal, AF and other Abnor- mal ECG Rhythms using a Cascaded Binary Classifier. In: *2017 Computing in Cardiology Conference (CinC)*. Computing in Cardi- ology, 2017. Available from DOI: 10.22489/cinc.2017.173-154.

[41]. HONG, Shenda; WU, Meng; ZHOU, Yuxi; WANG, Qingyun; SHANG, Junyuan; LI, Hongyan; XIE, Junqing. ENCASE: an EN- semble ClASsifiEr for ECG Classification Using Expert Features and Deep Neural Networks. In: *2017 Computing in Cardiology Con- ference (CinC)*. Computing in Cardiology, 2017. Available from DOI: 10.22489/cinc.2017.178-245.

[42]. LIU, Feifei et al. An Open Access Database for Evaluating the Algorithms of Electrocardiogram Rhythm and Morphology Ab- normality Detection. *Journal of Medical Imaging and Health Infor- matics*. 2018, vol. 8, no. 7, pp. 1368–1373. Available from DOI: 10.1166/jmihi.2018.2442.

[43]. CHEN, Tsai-Min; HUANG, Chih-Han; SHIH, Edward S.C.; HU, Yu-Feng; HWANG, Ming-Jing. Detection and Classification of Cardiac Arrhythmias by a Challenge-Best Deep Learning Neural Network Model. *iScience*. 2020, vol. 23, no. 3, pp. 100886. Available

from DOI: 10.1016/j.isci.2020.100886.

[44]. MOODY, George B; MARK, Roger G. *MIT-BIH Arrhythmia Database*. physionet.org, 1992. Available from DOI: 10.13026/C2F305.

[45]. *MIT-BIH Arrhythmia Database - Google Scholar*. 2020. Available also from: https://scholar.google.com/scholar?q=MIT-BIH+Arrhythmia+Database.

[46]. BOUSSELJOT, Ralf-Dieter; KREISELER, D; SCHNABEL, A. *The PTB Diagnostic ECG Database*. physionet.org, 2004. Available from DOI: 10.13026/C28C71.

[47]. STRODTHOFF, Nils; WAGNER, Patrick; SCHAEFFTER, Tobias; SAMEK, Wojciech. *Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL*. 2020. Available from arXiv: 2004.13701
[cs.LG].

[48]. GÉRON, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intel- ligent Systems*. O'Reilly Media, 2019. ISBN 1492032646. Available
also from: https://www.xarg.org/ref/a/1492032646/.

[49]. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook. org.

[50]. CUNNINGHAM, Padraig; DELANY, Sarah Jane. *k-Nearest Neigh- bour Classifiers: 2nd Edition (with Python examples)*. 2020. Available from arXiv: 2004.04523 [cs.LG].

[51]. BAGNALL, Anthony; LINES, Jason; BOSTROM, Aaron; LARGE, James; KEOGH, Eamonn. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*. 2016, vol. 31, no.
3, pp. 606–660. Available from DOI: 10.1007/s10618-016-0483-
9.

[52]. DAU, Hoang Anh; BAGNALL, Anthony; KAMGAR, Kaveh; YEH, Chin-Chia Michael; ZHU, Yan; GHARGHABI, Shaghayegh; RATANAMAHATANA, Chotirat Ann; KEOGH, Eamonn. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*.
2019, vol. 6, no. 6, pp. 1293–1305. Available from DOI: 10.1109/ jas.2019.1911747.

[53]. GÉRON, Aurélien. Chapter 4: Training Models. Logistic Regres- sion. In: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build*

Intelligent Sys- tems. O'Reilly Media, 2019, pp. 142–150. ISBN 1492032646. Available also from: https://www.xarg.org/ref/a/1492032646/.

[54]. GÉRON, Aurélien. Chapter 7: Ensemble Learning and Random Forests. Bagging and Pasting, Random Forests. In: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts,*

Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2019, pp. 192–199. ISBN 1492032646. Available also from: https: //www.xarg.org/ref/a/1492032646/.

[55]. GÉRON, Aurélien. Chapter 7: Ensemble Learning and Random Forests. Boosting. In: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelli-*

gent Systems. O'Reilly Media, 2019, pp. 203–208. ISBN 1492032646. Available also from: https://www.xarg.org/ref/a/1492032646/.

[56]. CHRISLB. *Diagram of a multi-layer feedforward artificial neural net- work.* 2012. Available also from: https://commons.wikimedia.org/wiki/File:MultiLayerNeuralNetworkBigger_english.

png.

[57]. *Course materials and notes for Stanford class CS231n: Convolutional Neural Networks for Visual Recognition.* 2020. Available also from: https://cs231n.github.io/convolutional-networks/.

[58]. HUANG, Jingshan; CHEN, Binqiang; YAO, Bin; HE, Wangpeng. ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Network. *IEEE Access*. 2019, vol. 7,pp. 92871–92880. Available from DOI: 10.1109/access.2019.2928017.

[59]. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

IEEE, 2016. Available from DOI: 10.1109/cvpr.2016.90.

[60]. GÉRON, Aurélien. Chapter 15: Processing Sequences Using RNNs and CNNs. Recurrent Neurons and Layers. In: *Hands-On Ma- chine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts,*

Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2019, pp. 497–501. ISBN

1492032646. Available also from: https:
//www.xarg.org/ref/a/1492032646/.

[61]. GÉRON, Aurélien. Chapter 15: Processing Sequences Using RNNs and CNNs. Handling Long Sequences. In: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019, pp. 511–520. ISBN 1492032646. Available also from: https:// www.xarg.org/ref/a/1492032646/.

[62]. CHO, Kyunghyun; MERRIËNBOER, Bart van; GULCEHRE, Caglar; BAHDANAU, Dzmitry; BOUGARES, Fethi; SCHWENK, Hol- ger; BENGIO, Yoshua. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natu- ral Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734. Available from
DOI: 10.3115/v1/D14-1179.

[63]. BLAD, John Erling. *Gated Recurrent Unit, fully gated version*. 2018. Available also from: https://commons.wikimedia.org/wiki/ File:Gated_Recurrent_Unit,_base_type.svg.

[64]. KINGMA, Diederik P.; BA, Jimmy. *Adam: A Method for Stochastic Optimization*. 2014. Available from arXiv: 1412.6980[cs.LG].

[65]. CARUANA, Rich; LAWRENCE, Steve; GILES, Lee. Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. Denver, CO: MIT Press, 2000, pp. 381–387. NIPS'00.

[66]. LAARHOVEN, Twan van. *L2 Regularization versus Batch and Weight Normalization*. 2017. Available from arXiv: 1706.05350 [cs.LG].

[67]. PASCANU, Razvan; MIKOLOV, Tomas; BENGIO, Yoshua. On the Difficulty of Training Recurrent Neural Networks. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. Atlanta, GA, USA: JMLR.org, 2013, III–1310–III–1318. ICML'13.

[68]. GÉRON, Aurélien. Chapter 3: Classification. Performance Mea- sures. In: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build*

Intelligent Sys- tems. O'Reilly Media, 2019, pp. 88–93. ISBN 1492032646. Available also from: https://www.xarg.org/ref/a/1492032646/.

[69]. PEREZ ALDAY, Erick Andres; GU, Annie; SHAH, Amit; LIU, Chengyu; SHARMA, Ashish; SEYEDI, Salman; BAHRAMI RAD, Ali; REYNA, Matthew; CLIFFORD, Gari. *Classification of 12-lead ECGs: the PhysioNet - Computing in Cardiology Challenge 2020*. PhysioNet, 2020. Available from DOI: 10.13026/F4AB-0814.