People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER

In **Computer Engineering**

Option: **Computer Engineering**

Title:

ECG Heartbeat Arrhythmias Classification Using Convolutional Neural Networks

Presented by:

- **TIGHILT Kahina**

Supervisor:

- **Mr. A. MOHAMMED-SAHNOUN**

Registration Number:...../2022

Dedication and Acknowledgement

I dedicate this humble work to my family, a special feeling of gratitude to my loving parents, my two brothers and two sisters. I also dedicate this dissertation to my friends, who have supported me throughout the process. I want to give a special thanks to two of my closest friends AYACHE Ilyes and BELLIL Sara for providing all sorts of support when needed; I will always appreciate what they have done. Finally, I'd like to acknowledge the assistance of my supervisor Mr. A. MOHAMMED-SAHNOUN who guided me throughout this project.

Abstract

While cardiac diseases are increasing in the past years, heart monitoring has become crucial to assess the heart behavior and detect any arrhythmia if available. Electrocardiograms or ECG signals, are records of the electrical activity of the heart that illustrates the way the depolarization wave flow in each heartbeat; A proper study of an ECG signal's characteristics is the gold standard of providing effective diagnostics for cardiac diseases.

The aim of this work is to provide an automatic approach of analyzing and detecting arrhythmias using deep neural networks or to be more specific, 2-dimensional convolutional neural networks. The great performance in extracting the spatial features of input image data is what contributed in CNNs popularity and is what makes it more suitable than any other model. In this study a CNN architecture was proposed and discussed in terms of performance, and the impact of deep learning techniques which are batch normalization and dropout on it.

Table of contents

List of figures

List of tables

Abbreviations

WHO	World Health Organization
CVD	Cardiovascular disease
PVCs	Premature ventricular contraction
VT	Ventricular tachycardia
VF	Ventricular fibrillation
ECG/EKG	Electrocardiogram
2D CNN	2-dimensional convolutional neural network
N	Normal heartbeat
LBBB	Left bundle branch block
RBBB	Right bundle branch block
SA node	The Sinoatrial node
AV node	The Atrioventricular node
VL	Augmented vector left
aVR	Augmented vector right
aVF	Augmented vector foot
BW	Baseline wander
PLI	Power line interference
CNN	Convolutional neural network
AI	Artificial intelligence
ANN	Artificial Neural networks
MSE	Mean squared error
ReLU	Rectified linear unit
TP	True positive
TN	True negative
FP	False positive
FN	False negative
BN	Batch normalization

General introduction

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs in the year 2017 all over the world which is about 31% of all deaths. Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmias including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances. For example, prolonged premature ventricular contraction (PVCs) beats occasionally turn into a ventricular tachycardia (VT) or ventricular fibrillation (VF) beats which can immediately lead to heart failure [1].

Monitoring the heart health state and early detection of cardiac arrhythmias is a key factor in being able to properly manage them and avoid later complications, for that matter we use ECG (Electrocardiogram) signals which are a graph of voltage versus time of consecutive heartbeats that allows us to visualize and diagnose the human heart behavior.

In this work we focus on analyzing and classifying heartbeat arrythmias. A deep learning approach using 2-dimensional convolutional neural network (2D CNN) is proposed to classify the ECG heartbeats extracted from the recordings of the MIT BIH arrythmia database into four different classes, being Normal heartbeat (N), Left bundle branch block (LBBB), Right bundle branch block (RBBB) and Premature ventricular contraction (PVC). However, before feeding our data to the neural network we need to do some preprocessing including filtering and segmentation, also since 2D CNNs accept only 2D images as inputs we need to convert the segmented heartbeats into grayscale images of dimension 200×200.

Prior works have been done to classify ECG heartbeat images into different classes, different 2D CNNs architectures have been used and achieved good accuracies, for instance, Nahom Ghebremeskel1 and Vahid Emamian2 [2] have achieved accuracy of 90%, also XUEXIANG XU and HONGXING LIU [3] have achieved 99.43% accuracy after using 4 convolutional layers , 2 pooling layers and 2 fully connected layers. These previous work's results illustrate the outstanding performance of 2D CNNs on classifying input images.

This report mainly consists of three chapters in which we introduce all the theories and methodologies used to obtain the results that are finally demonstrated and discussed in the last chapter. The first chapter titled ***"Generalities about deep learning and electrocardiograms"*** gives us an understanding about the cardiovascular system in all its aspects and ECG signals characteristics in addition all the needed details to understand how neural networks learn from given data. ***"ECG Preprocessing and CNNs"*** being the second chapter in which we discuss how from an ECG recording we get a grayscale image of single heartbeats, as well as describing the underlying working of a CNN model. The third chapter is the final chapter where the experimental results are displayed and discussed.

Chapter 1

Generalities about deep learning and electrocardiograms

1.1 Introduction

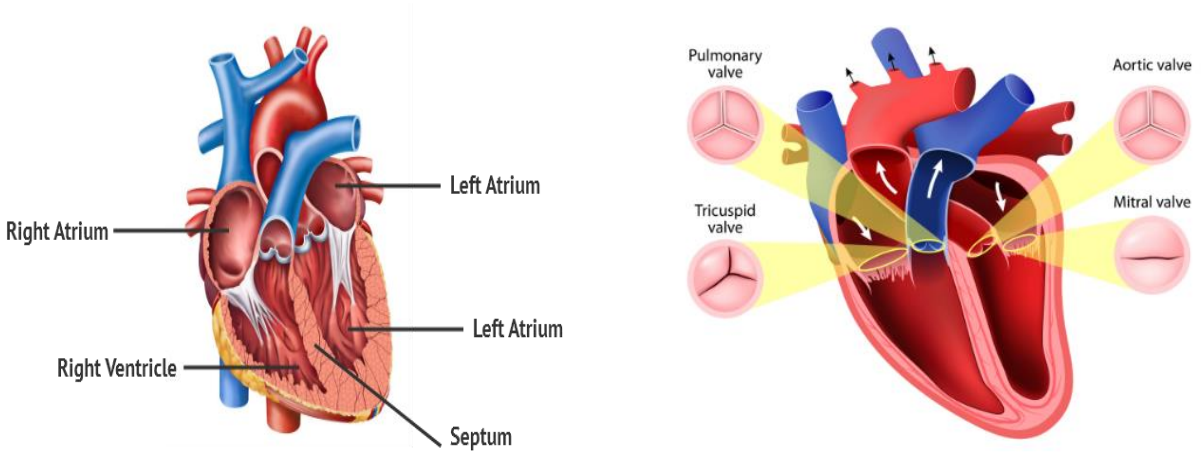
The heart is a vital organ whose function is to pump blood all over the body. Any irregularity in the heart's pumping mechanism might be a sign of a serious problem, therefore, early detection of any abnormality might save lives. Luckily, we have been able to assess the heart's behavior by means of electrocardiogram signals or ECG, where cardiologists interpret the signal and conclude whether the heart beating rhythm is normal or not. However, going through every patient's heart recording is time consuming and exhausting. For that, artificial intelligence can substitute for human intelligence. By using deep learning models to diagnose the patient's ECG recording, cardiologists can devote their time and energy on more complex tasks. In this chapter, we will focus on getting familiar with electrocardiograms and the different types of heart arrhythmias, in addition to understanding the deep learning structure and its pillar algorithms.

1.2 Human heart and electrocardiography

In a cardiovascular system the heart is the most important organ, about the size of a fist, located inside the thoracic cage, its function is to pump blood within the body to provide a steady supply of oxygen to the brain and other vital organs; It also works with other body systems to control the heart rate and blood pressure.

1.2.1 Anatomy of the heart

The human heart is located in the middle of the chest between the lungs; It sits behind and slightly to the left of the breastbone; It consists of four chambers two upper chambers called left and right Atria; Two lower chambers called left and right Ventricles; A wall labeled Septum of the heart that separates the right and left sides of the heart, this latter is divided into two parts the Interventricular Septum isolating the two ventricles and interatrial septum isolating the two atria; Four valves that control the blood flow, two of them being the Mitral and the Tricuspid valves, they allow the movement of the blood from the atria to the ventricles, the other two valves being the Aortic and Pulmonary valves, they move blood out of the ventricles to the lungs and the rest of the body. Figure 1.1 illustrates the Chambers and valves of the heart.



Chambers of the heart. ***Valves of the heart.***

Figure 1.1: Chambers and valves of the heart [4].

Human heart also consists of blood vessels through which the heart pumps blood, which are divided into three types, Arteries that carry oxygenated blood from the heart to the body and lungs; Veins that transfer deoxygenated blood to the heart and small blood vessels called Capillaries where the body exchanges oxygenated and deoxygenated blood. Figure 1.2 illustrates the different blood vessels of the heart.

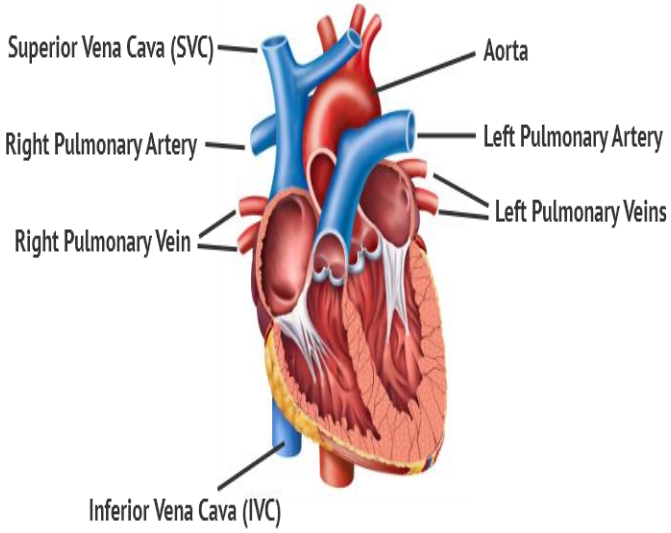


Figure 1.2: Major blood vessels of the heart [4].

1.2.2 Electrical activity of the heart

The heart has to pump blood constantly in order to keep us alive; The heart's electrical conduction system is a network of nodes, cells and signals specialized for this job; It determines both the heart rate and the heart contraction pattern.

The Sinoatrial node (SA node), located in the right Atrium acts as a natural pacemaker for the heart; It leads the beating pattern, it is where the electrical impulse begin and then this latter continues to spread all over the atria causing the muscle in the atrium to contract; The Atrioventricular node (AV node) , also situated on the right atrium receives the impulse and delays it to ensure that the blood has been completely ejected to the ventricles, afterwards the signal follows the pathway through the atrioventricular bundle towards the apex of the heart along the right and left bundle branches spreading all over the ventricles completing the electromechanical cycle of a heartbeat. Figure 1.3 demonstrates the electrical pathway of the heart.

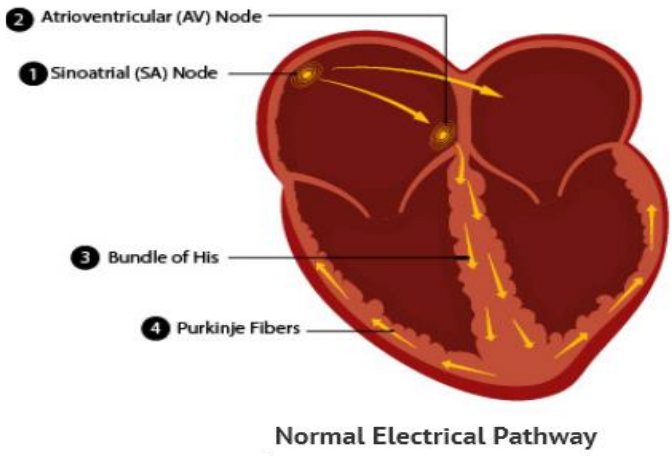


Figure 1.3: Normal electrical pathway of the heart [4].

1.2.3 Electrocardiography and ECG signals

Regular monitoring of the heart's activity plays a major role in early detection of several heart diseases; for that we use Electrocardiography, which is the measurement of electrical activity in the heart and the recording of such activity as a visual trace (on paper or on an oscilloscope screen), using electrodes placed on the skin of the limbs and chest [5]. Resulting in a graph of voltage versus time of consecutive heartbeats called Electrocardiograms or ECG.

1.2.3.1 ECG signals

ECG signal illustrates how the depolarization wave flow during each heartbeat, Figure 1.4 shows an ECG heartbeat. A heartbeat in an ECG signal has different parts namely the P wave which represents the Atrial depolarization in regards to the Sinoatrial node triggering; Followed up by the QRS complex resulted by the ventricular contraction and lastly the T wave representing the ventricular repolarization.

ECG can be divided into various segments and intervals namely the PR interval; The PR segment; The QRS interval; The ST segment and the QT interval.

PR interval: The PR interval is the time from the onset of the P wave to the start of the QRS complex. It reflects conduction through the AV node. It is about 0.12 to 0.2 seconds in duration [6].

PR segment: The PR segment is the portion of the ECG from the end of the P wave to the beginning of the QRS complex [7].

QRS interval: It is the time duration that covers the QRS complex from beginning to end.

ST segment: During the ST segment, all the ventricular myocardium is depolarized. All have positive charges. So, there is nothing potential difference to be recorded by the voltmeter (ECG machine). So, you have a flat line [7].

QT interval: Important because it captures the beginning of ventricular depolarization to the ventricular repolarization. It covers the entire ventricular activity [7].

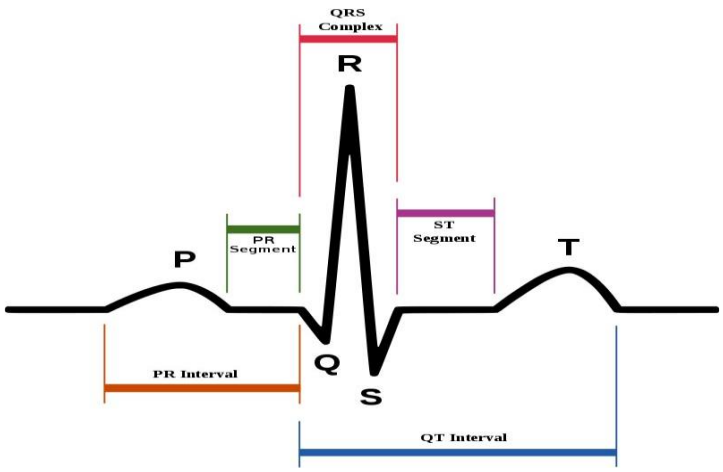


Figure 1.4: ECG heartbeat [8]**.**

1.2.3.2 ECG derivations

An ECG is recorded using either electrodes or sticky patches, placed on the body surface, typically over the chest, the left arm, the right arm and the left leg. These electrode wires are connected to the EKG machine with recordings from twelve different angles, called **12-lead EKG**, resulting in six limb leads and six chest leads.

The six limb leads are obtained from three electrodes attached to the right arm; Left arm and left leg; These latter are divided into two sets:

Three bipolar limb leads: Figure 1.5 demonstrates how to obtain bipolar limb leads. The measurement of the voltage requires two poles, negative and positive; The ECG machine uses the negative pole as zero reference hence the position of the positive pole is the point of view [9] ; Depending on which pair is captured Lead I, II and III are obtained.

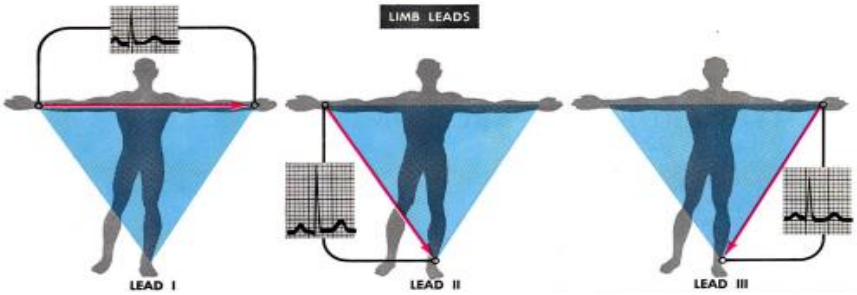


Figure 1.5: Three bipolar limb leads demonstration [10].

Three augmented leads: These leads are unipolar, figure 1.6 demonstrates how to obtain augmented leads, they use one limb electrode as positive pole and takes the average of inputs from the other two limbs as the zero reference [9], depending on the point of view aVL (augmented vector left), aVR (augmented vector right) and aVF (augmented vector foot) are obtained.

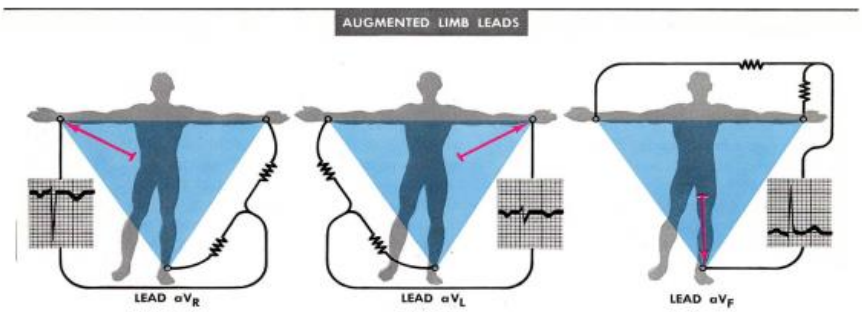


Figure 1.6: Three augmented leads demonstration [10].

The chest leads are unipolar leads, the corresponding chest electrode serve as the positive pole, the reference negative value is the same for all the chest leads and is calculated as the average of inputs from the three limb electrodes [9]. Figure 1.7 demonstrates how to obtain chest leads

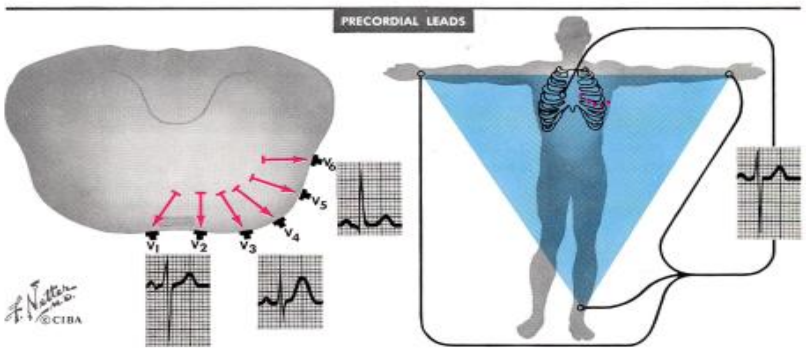


Figure 1.7: chest leads demonstration [10].

Depolarization towards a lead produces positive deflection; Depolarization away from a lead gives a negative deflection, the reverse is true for repolarization. Thus, leads that look at the heart from different angles may have waves pointing in different directions [9].

1.2.4 Heart arrhythmias

An arrhythmia or cardiac arrhythmia is defined as an abnormality in the rate or the rhythm of the heartbeat. Meaning the heart beats too fast, too slowly or with abnormality. Arrhythmias are divided into three categories, Supraventricular arrhythmias; Ventricular arrhythmias and Bradyarrhythmia.

1.2.4.1 Supraventricular arrhythmias

Supraventricular arrhythmias are defined as an irregularly fast heartbeat that affects the heart's upper chambers; It exists in various types such as:

- **Paroxysmal supraventricular tachycardia (PSVT):** A rapid but regular heart rhythm that comes from the atria. This type of arrhythmia begins and ends

suddenly [11].

- **bypass tract tachycardias:** A fast heart rhythm caused by an extra, abnormal electrical pathway or connection between the atria and ventricles [11].

- **AV nodal re-entrant tachycardia (AVNRT):** A fast heart rhythm caused by the presence of more than one pathway through the atrioventricular (AV) node [11].
- **Atrial tachycardia:** A rapid heart rhythm that starts in the atria [11].

- **Atrial fibrillation:** A very common irregular heart rhythm. This happens when many impulses begin and spread through the atria, competing for a chance to travel through the AV node. This results in a disorganized rapid and irregular rhythm [11].

- **Atrial flutter:** An atrial arrhythmia caused by one or more rapid circuits in the atrium [11].

1.2.4.2 ventricular arrhythmias

A ventricular arrhythmia is a heart rhythm problem that begins in the heart's ventricles Different kinds exist namely:

- **Premature ventricular contractions (PVCs):** Early, extra heartbeats that start out in the ventricles. Most of the time, PVCs don't cause any symptoms or require treatment, but Frequent PVCs may increase the risk of developing other, more serious cardiac arrhythmias. This type of arrhythmia is common and can be related to stress, too much caffeine or nicotine, or exercise. They can be also be caused by heart disease or electrolyte imbalance [11].

- **Ventricular tachycardia (V-tach):** A rapid heartbeat that begins in the ventricles. The rapid rhythm keeps the heart from adequately filling with blood, and less blood is able to pump through the body [11].

- **Ventricular fibrillation (V-fib):** An erratic, disorganized firing of impulses from the ventricles. The ventricles quiver and can't generate an effective contraction, which results in a lack of blood being delivered to your body [11].

- **Long QT:** While this is not an arrhythmia, a longer QT interval than normal increases the risk for "torsade de pointes," a life-threatening form of ventricular tachycardia [11].

1.2.4.3 Bradyarrhythmia

A bradyarrhythmia is a slow heart rhythm that is usually caused by an abnormality in the heart's conduction system. Types of bradyarrhythmia include:

- **Sinus node dysfunction:** Slow heart rhythms due to an abnormal SA node [11].
- **Heart block:** A delay or complete block of the electrical impulse as it travels from the sinus node to the ventricles [11]:

  Left bundle branch block: It is a partial or a total block of the conducting system in the left bundle branch causing a delayed ventricle contraction.

  Right bundle branch block: It is a partial or a total block of the conducting system in the right bundle branch causing a delayed ventricle contraction.

1.3 Artificial intelligence and neural networks

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions, using various complex algorithms and mathematical functions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving [12].

1.3.1 Machine learning and deep learning

Machine learning is a subset of AI used to build intelligent machines to learn from a huge amount of data using statistical techniques, and make predictions and decisions; Machine learning can be divided into three approaches which are Supervised learning; Unsupervised learning and Reinforcement learning:

- **Supervised learning:** In supervised learning the machine learns from a labeled dataset, meaning, the input data is already associated with an appropriate output, and based on this concept the machine learns how to link the correct output to the input. Types of Supervised learning include Regression and Classification.
- **Unsupervised learning:** As the name indicates, unsupervised learning is a technique in which datasets are not labeled; rather the learning model itself has to find the patterns and the underlying structure of the data and group this latter into

categories based on similarities. Types of unsupervised learning include Clustering and Dimensionality reduction.

- **Reinforcement learning**: Reinforcement Learning is a feedback-based Machine learning technique, an agent learns and adjusts to its environment, each action done by this agent gets feedback depending on whether the action is good or bad.

Deep Learning is a subset of machine learning that uses neural networks inspired by the biological neural network of the human brain, in the aim of imitating human like decision making and problem solving. What makes deep learning particular is that it requires less human intervention to the detriment of large data requirements and a mathematically more complex algorithms compared to machine learning.

1.3.2 Artificial Neural networks

1.3.2.1 What is an artificial Neural network?

Artificial Neural networks or ANN for short, are the foundation of deep learning models. Established on a set of nodes and weighted connections, they are designed to mimic the way biological neurons signal to one another.

The human brain consists of a large number of communicating neurons interconnected to each other creating a neural network. Figure below represents an illustration of a single biological neuron, consisting of a Cell body, a Nucleus and a set of Dendrites designed to receive information as an input to the neuron, the information is then transmitted along the Axon as an output for the succeeding neuron.

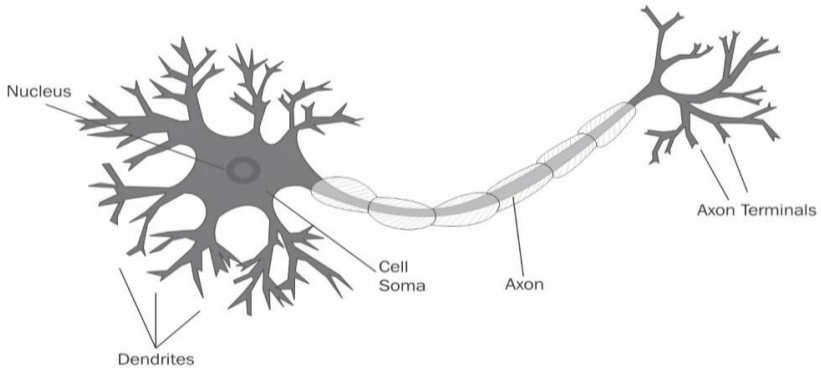


Figure 1.8: Biological neuron [13].

Similarly, an artificial neural network is composed of several Perceptrons interconnected to each other. A perceptron is a mathematical representation of a biological neuron, also known as an artificial neuron, it consists of a set of inputs; weights; A bias; A weighted sum and an activation function:

- **Weight:** A parameter that determines how much influence the input will have on the output.
- **Bias:** A constant whose role is to shift the activation function to the right or to the left in order to fit the prediction with the data better.
- **Weighted sum:** The sum of the inputs multiplied by their weights.
- **Activation function:** A mathematical function that maps the input to the output, and decides whether the neuron should be activated or not.

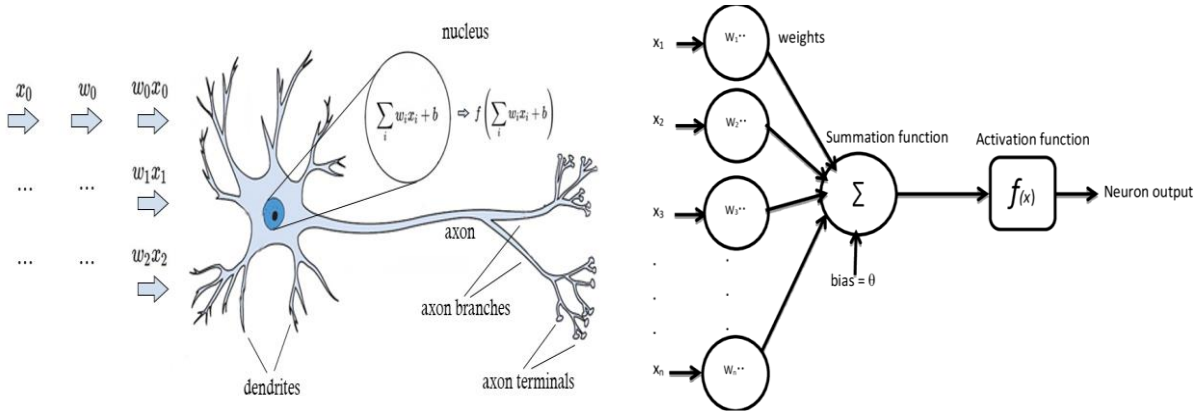


Figure 1.9: Artificial neuron demonstration.

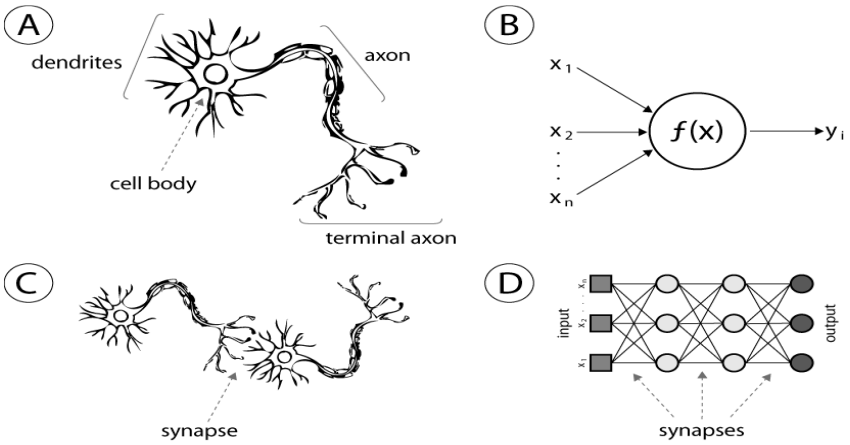


Figure 1.10: Biological and artificial neurons [14].

Figure 1.10 demonstrates a single perceptron, which takes a set of numerical inputs, multiply each input with the corresponding weight and adds them together with the bias, the result is then passed through an activation function yielding to the final output which is subsequently transmitted to the following perceptron.

$$\boldsymbol{y = f(\sum_{i=0}^{n} w_i x_i + \Theta)} \qquad \textbf{Equation 1.1}$$

By combining multiple perceptrons we get a single layer as illustrated in figure below.

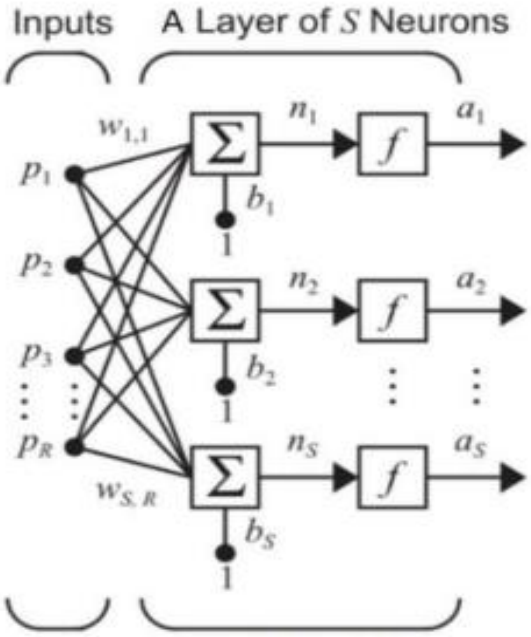


Figure 1.11: Single neural network layer.

Where $p = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix}$ is the input vector, $w^T = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_s^T \end{bmatrix}$ is the corresponding weight vector of each perceptron, $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{bmatrix}$ is the set of biases and $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_s \end{bmatrix}$ is the output vector that is calculated as follow:

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a_1} \\ \boldsymbol{a_2} \\ \vdots \\ \boldsymbol{a_s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{w_1^T p} + \boldsymbol{b_1}) \\ \boldsymbol{f}(\boldsymbol{w_2^T p} + \boldsymbol{b_2}) \\ \vdots \\ \boldsymbol{f}(\boldsymbol{w_s^T p} + \boldsymbol{b_s}) \end{bmatrix} = \boldsymbol{f}(\boldsymbol{w^T p} + \boldsymbol{b}) \qquad \textit{Equation 1.2}$$

1.3.2.2 How does neural networks work?

Neural networks take a large set of input data called the training data, train themselves into identifying the patterns within this data and subsequently being able to predict the appropriate output for each input of a new set of data called test data. But how does the neural network accomplish that?

To help visualize the learning process of a neural network, let us examine its underlying structure; A neural network is a collection of nodes arranged in layers. Neurons of each layer are connected to the next layer through weighted arrows. There exist three types of layers in a neural network, an input layer, a hidden layer and an output layer:

- **Input layer:** The first layer in the network and is responsible for passing the input data to the subsequent layers in the network for further processing.

- **Hidden layer:** A hidden layer is in between the input and output layers. thus, hidden. And it is where all the computations are done; There can exist either one or multiple hidden layers in a network, the greater the number of hidden layers the more complex the network is.

- **Output layer:** The last layer in the network, it predicts the final result of the input.

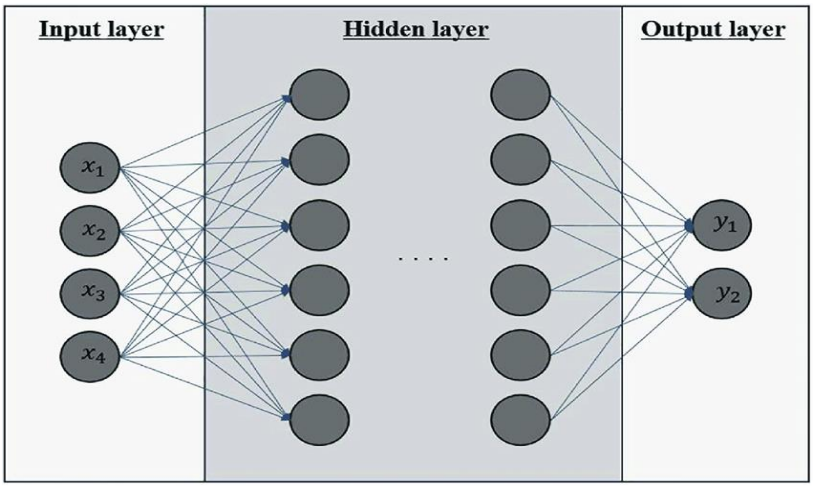


Figure 1.12: Structure of a neural network.

Samples of data are fed to the neurons of the first layer, they get multiplied to the corresponding weights, added together and transmitted to the next layer's neurons, these neurons are associated with a numerical value called the bias which is added to the weighted sum, the overall sum is passed through an activation function which decides whether the current neuron will be activated or not; The activated neuron will transmit the output through the weighted tensors to the succeeding neuron and the same process will proceed throughout all the layers. This is known as the Forward propagation of data.

The prediction is made in the output layer, where the neuron with the highest probability determines the output. However, the prediction can be wrong as the network is yet to be trained. The training process of supervised learning differ from that of unsupervised learning.

In supervised learning, along with the input, the corresponding output is also fed to the network, this output is then compared to the resulting output producing an error value, this latter is passed backwards through the network which is known as Back propagation. Back propagation is the key factor in the training process, as the error is back propagated, the weights are tuned in a way that the error is reduced resulting in a more accurate prediction.

This pattern of forward propagation and back propagation is performed on all the inputs in the training dataset until the weights are well adjusted and the network can make precise predictions. After all samples of the training dataset are passed through the network, we can say that the network has done one epoch.

While supervised learning algorithm learns by continuously making predictions on the input data and analyzing the model's prediction error, in unsupervised learning the model is uninformed of the actual output, The model learns features in the data on its own and will attempt to categorize the unlabeled data based on feature similarity.

1.3.3 Backpropagation

As mentioned earlier back propagation is an important algorithm in supervised learning, it is an optimization process that helps the model achieve better accuracy. Before addressing the working of backpropagation, we first need to define two related terms, Loss function and Gradient descent.

1.3.3.1 Loss function

The loss function also called the cost function, is a mathematical method used to determine how well the deep learning model is performing, it assesses the difference between the expected output and the output produced by the model; The smaller the loss the better the model is performing. For instance, let us consider the Mean squared error loss function (MSE).

The loss or the error in MSE is calculated by taking the difference between the expected output and the model's output for each input, the difference then is squared and divided by the number of data points:

$$\boldsymbol{MSE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{y_i} - \hat{\mathbf{y}}_{\boldsymbol{i}})^{\mathbf{2}} \qquad \textbf{Equation 1.3}$$

where:

n: number of data points.

$\hat{\mathbf{y}}_{\boldsymbol{i}}$: the models output.

y_i: the expected output.

MSE is a simple loss function. There exist different types of loss functions suitable for different models, the general idea holds the same for all of them, but each function has its own way for evaluating the loss.

Cross entropy or log loss, is one of the most used functions to measure the loss of a classification model whose output is a probability between 0 and 1. Its general formula for M given classes in a multi-class classification problem is given by:

$$\boldsymbol{Cross\ entropy} = -\sum_{\boldsymbol{c=1}}^{\boldsymbol{M}} \boldsymbol{Opserved_c} \times \boldsymbol{Log}(\boldsymbol{Predicted_c}) \qquad \textbf{Equation 1.4}$$

where:

$M:$ number of classes.

$Opserved_c$: The actual output

$Predicted_c$: The model's output.

1.3.3.2 Gradient descent

After calculating the loss function, our goal is to minimize it to the greatest extent, for that one more method is required which is gradient descent. Since weights in the network indirectly impact the loss, gradient descent works on tuning these weights until achieving better accuracy and minimum loss.

In mathematics gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function [15]. The concept is we calculate the gradient of the loss function with respect to a chosen weight, and then we take steps away from that gradient by a certain learning rate resulting in an updated weight. By doing this iteratively for all weights, we update the weights at each iteration until achieving optimal values that result in a minimum loss function. Figure 1.13 illustrates gradient descent.

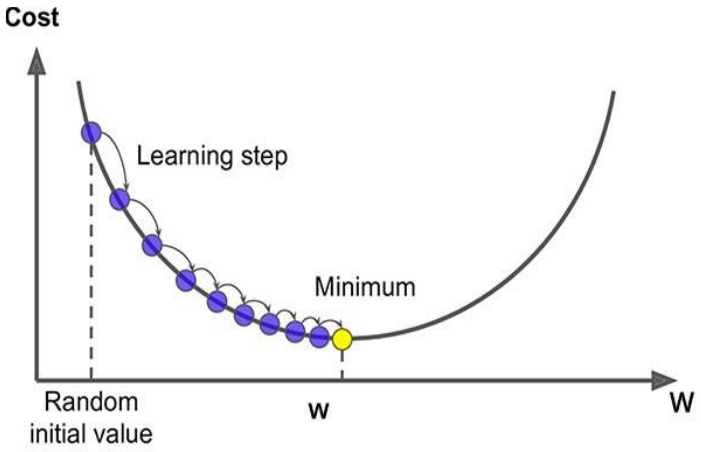


Figure 1.13: Gradient descent [16].

The equation by which the weights are updated:

$$W_{new} = W_{current} - \alpha \frac{\partial L}{\partial W_{current}} \qquad \textit{Equation 1.5}$$

where:

α: the learning rate, it determines by how much we move towards the minimum.

L: the loss function.

$\frac{\sigma L}{\sigma W_{current}}$: The gradient.

Gradient descent can be divided into three variants:

- **Batch gradient descent:** given n data points in the training set. In batch gradient descent we perform the forward pass on all the n data points consecutively, calculate the loss function and then update the weights in one iteration. In other words, we update the weights only once per epoch. Using this method, we obtain a smooth convergence of the loss function towards the local minimum but this would require high computation cost.

- **Stochastic gradient descent:** in this case forward and backward passes are performed for each data point, and the weights are updated n times per epoch. Although this method is not computationally expensive, it results in noisy variations of the loss function.

- **Mini-batch gradient descent:** the training set is divided into batches, where the batch size is greater than one and smaller than n, the forward pass is done after completing one batch, and the weights are updated n/batch-size times per epoch. This method is less noisy and more computationally expensive than stochastic gradient descent but not as smooth and computationally expensive as the batch gradient descent.

But how do we calculate the gradient? To better understand how the gradient is calculated let us look at the network below:

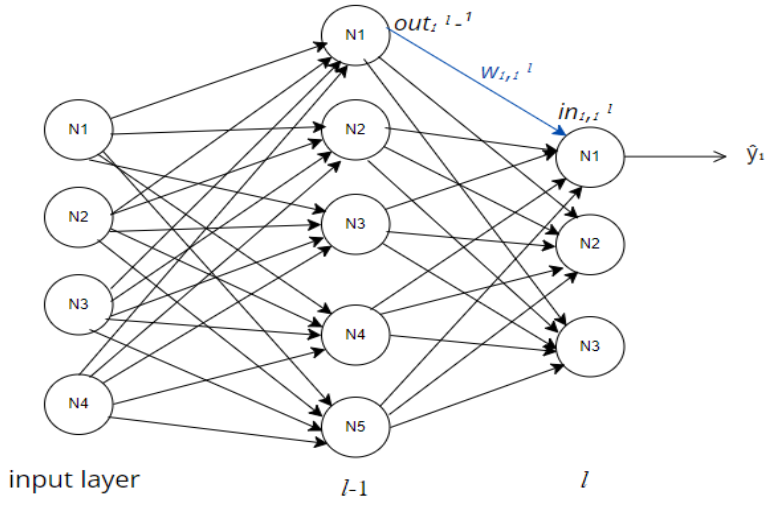


Figure 1.14: example neural network.

Figure 1.14 illustrates a neural network; we will take the weight $w_{1,1}^{l}$ as an example to calculate the gradient of the loss function. The gradient of the loss function is denoted as: $\frac{\partial L}{\partial w_{1,1}^{l}}$.

The loss function is composed of different functions, if we observe we find that the loss function L depends on the activation output $\hat{y}_1$ of node N_1^l, and this output depends on the input coming from the node N_1^{l-1} which is $in_{1,1}^{l}$, and this input depends on the weight $w_{1,1}^{l}$; Based on the chain rule the derivative of L with respect to $w_{1,1}^{l}$ is calculated from the product of the derivatives of the composed functions :

$$\frac{\delta L}{\delta w_{1,1}^{l}} = \left(\frac{\delta L}{\delta \hat{y}_1}\right)\left(\frac{\delta \hat{y}_1}{\delta in_{1,1}^{l}}\right)\left(\frac{\delta in_{1,1}^{l}}{\delta w_{1,1}^{l}}\right) \qquad \textit{Equation 1.6}$$

where:

$w_{1,1}^{l}$: The weight to be updated.

L: the loss function.

$\hat{y}_1$: output of neuron 1 of layer l

$in_{1,1}^{l}$: input of neuron 1 of layer l

$\frac{\delta L}{\delta w_{1,1}^{l}}$: The gradient.

Backpropagation calculates the gradient of the loss function with respect to the weight, this calculation requires moving backwards through the network as the loss function is a composition of previous functions, Then the weight gets updated by means of gradient descent. We repeat the same process for all weights of all layers in the network up until reaching the first layer.

1.4 Conclusion

In this chapter we have seen that a normal heart has a coherent mechanism of beating. Any distortion in that mechanism will lead to an arrhythmia, and detecting arrhythmias is achieved by a diagnosis of an electrocardiogram signal that is acquired using the 12-lead ECG method. We also got acquainted with neural networks, and how the whole learning process of the network is based on the backpropagation algorithm and gradient descent, where these latter are the key features in achieving a better model accuracy.

Chapter 2

ECG preprocessing and CNNs

2.1 Introduction

Early detection of heartbeat arrhythmia is important since it ensures prompt and adequate management to avoid later complications. With the help of deep learning, such detection becomes easier as it is done automatically by computers. Neural networks can implement a classification algorithm that will be fed a large number of heartbeats, and learn to classify them into the corresponding arrhythmia class, hence, detect any arrythmia if available. Different models can be used for classification problems, however, the great performance of CNNs in detecting important features in input data, is what makes it more suitable than any other model; But before feeding any input to any neural network, some preprocessing should be done on that input. In this chapter we will tackle the different preprocessing techniques used on the ECG signals as well as introducing the working mechanism of 2D CNN that will be used later on in our work.

2.2 Preprocessing of ECG signals

Raw ECG data is obtained from the MIT BIH arrhythmia database. Since the neural network's outcome heavily depend on the quality of input data, this raw data needs to go through preprocessing before feeding it into our neural network.

2.2.1 MIT BIH arrhythmia database

The MIT-BIH Arrhythmia Database contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979. Twenty-three recordings were chosen at random from a set of 4000 24-hour ambulatory ECG recordings collected from a mixed population of inpatients (about 60%) and outpatients (about 40%) at Boston's Beth Israel Hospital; the remaining 25 recordings were selected from the same set to include less common but clinically significant arrhythmias that would not be well-represented in a small random sample [17].

The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10mV range. Two or more cardiologists independently annotated each record; disagreements were resolved to obtain the computer-readable reference annotations for each beat (approximately 110,000 annotations in all) included with the database [17].

The data and annotations in most PhysioBank databases are stored in a Waveform Database (WFDB) format, one standard category being the MIT Format:

MIT Signal files (. dat): are binary files containing samples of digitized signals. These store the waveforms.

MIT Header files (.hea): are short text files that describe the contents of associated signal files.

MIT Annotation files (.atr): are binary files containing annotations (labels that generally refer to specific samples in associated signal files).

In order to be able to deal with signals of the MIT BIH database, one should use the WFDB package, which contains a set of functions for reading, writing and processing signals of the physiobank databases.

2.2.2 Noise in ECG signals

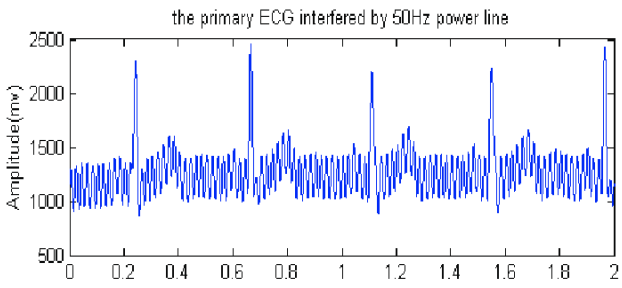
An ECG signal might be affected by the presence of artifacts (noise) that results from either external or internal interference. There exist mainly four types of artifacts:

- **Powerline Interference (PLI):** this kind of noise is due to the electromagnetic interference in the supply and the cables carrying the ECG signal, it is either a 50Hz or 60Hz interference. PLI affects the interpretation of an ECG as it overlaps with T wave and P wave as shown in Figure 2.1.

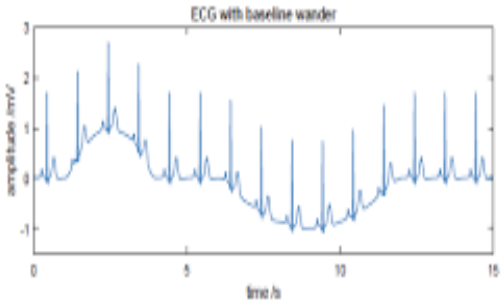
- **Baseline wander (BW):** also known as Base Line Drift and is a low frequency noise generally below 0.5Hz caused by displacement of electrodes with respect to the position of the heart due to breathing or coughing. As a result, ECG is shifted from zero - potential baseline, causing difficulty in ST segments' analysis which is a morphology of very little electric potential. It is generally below 0.5 Hz [18].

- **Muscle artifacts:** caused by electrical activity of muscles in contact with electrodes on the body. it mainly exists between 20 Hz - 100Hz.

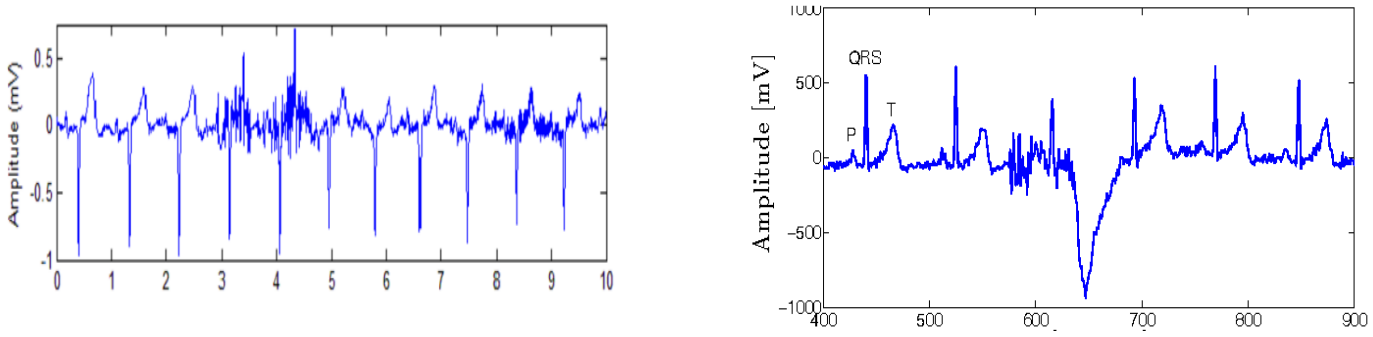
- **Motion artifact**: Abrupt movements such as coughing while ECG is being recorded results in motion artifacts (MA) in ECGs which appear as sudden changes the in electric potentials [18].



Powerline interference [19]



Baseline wander [20]



Muscle artifact ***Motion artifact [21]***

Figure 2.1: Different types of artifacts in ECG signals.

2.2.3 Filtering ECG signals

Eliminating noise from an ECG signal makes the different characteristics and features of an ECG heartbeat more clear, therefore it helps the model in making accurate predictions.

Figure 2.2 illustrates the frequency spectrum of record 111 of the MIT-BIH arrhythmia database:

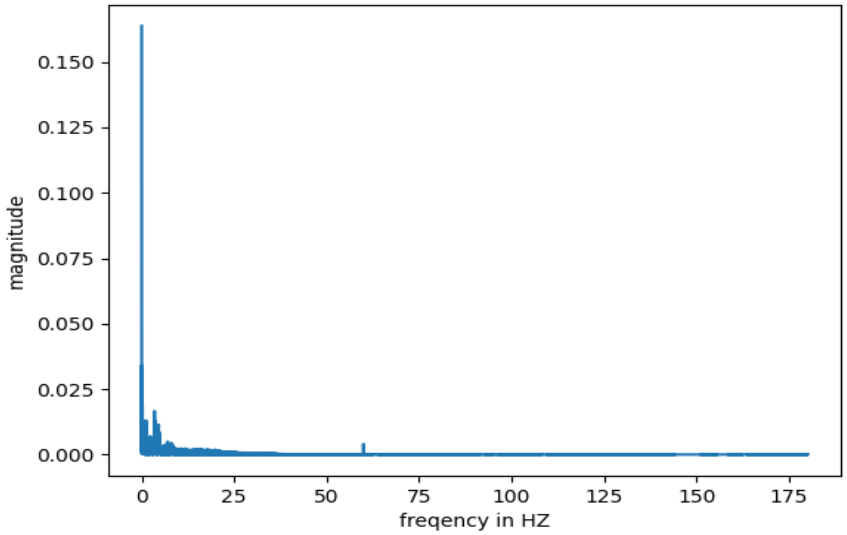


Figure 2.2:Frequency spectrum of record 111 of the MIT-BIH database.

As we can observe there exist two spikes, one at DC and one at 60Hz representing the baseline wander and powerline interference respectively. In order to eliminate these noises, we propose cascading a highpass and lowpass Butterworth IIR digital filters of order 5.

For PLI: butterworth lowpass filter of order 5 and Fc=45Hz.

For BW: butterworth highpass filter of order 5 and Fc=0.5Hz.

Frequency response of the 5th order butterworth filters are shown below:

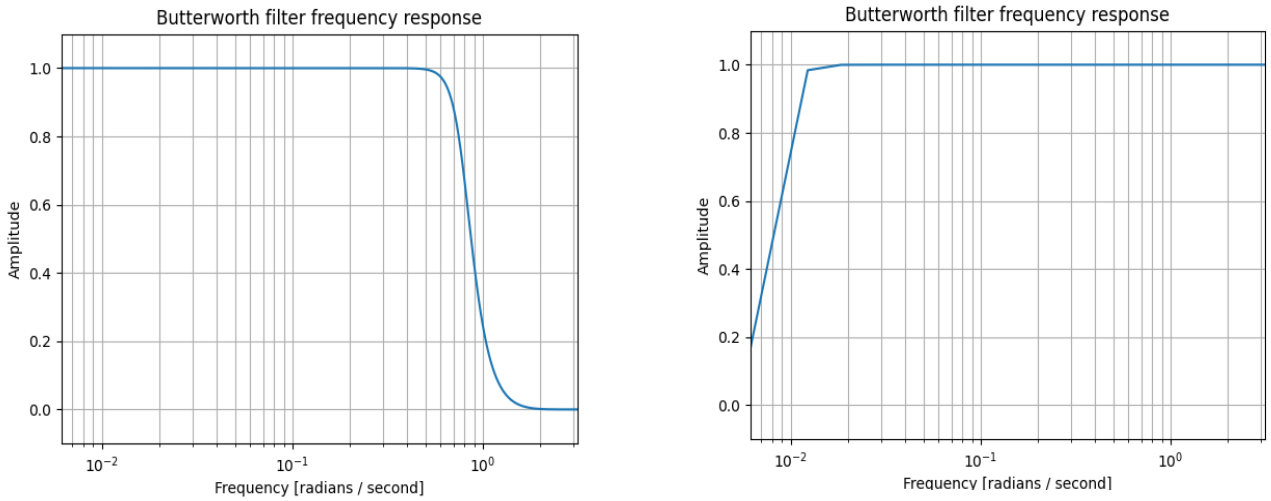


Figure 2.3: Frequency responses of the chosen filters.

The filtering was applied using zero phase filtering technique and the results are shown below for both frequency and time domain. As we can see, the PLI noise has been removed and the baseline is corrected:

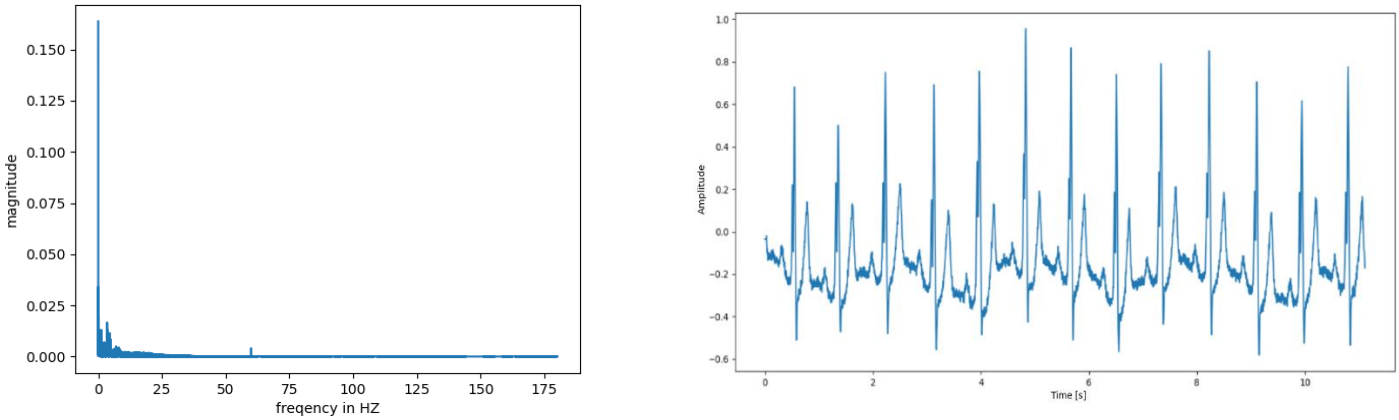


Figure 2.4: Unfiltered signal in frequency and time domain of record 111.

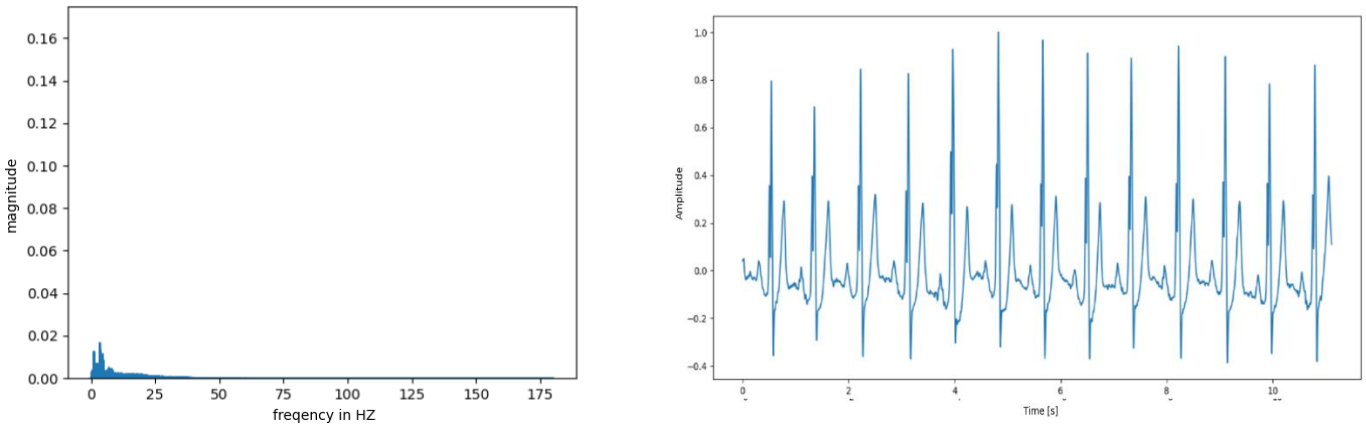


Figure 2.5: Filtered signal in frequency and time domain of record 111.

2.2.4 Pan Tompkins algorithm

Training a neural network to classify heartbeats into corresponding arrhythmias will require proper segmentation of the ECG signal into individual heartbeats, for that we need to leverage the position information of the R peaks. The method that we chose for detecting R locations was the Pan Tompkins algorithm, for the standard 24 h MIT BIH arrhythmia database, this algorithm correctly detects 99.3 percent of the QRS complexes [22].

The pan Tompkins algorithm uses three different steps of processing, digital filtering, nonlinear transformation and decision rule. These steps are based on applying a series of filters to highlight the frequency components of the QRS complex and to amplify its contribution; Then applying adaptive thresholding to detect the peaks of the filtered signal.

2.2.4.1 Filtering and nonlinear transformation

Noise filtering: noise cancellation is not only beneficial for accuracy of the deep learning model but also increases the detection sensitivity of the QRS complex in the pan Tompkins approach, as it reduces false detections due to interference present in the signal. While in Pan Tompkins algorithm a cascaded highpass and lowpass filters of cutoff frequencies 5Hz and 11Hz respectively were used, we chose to run the algorithm with the previously proposed filter. Noise elimination of record 100 of the MIT-BIH arrhythmia database is shown below.

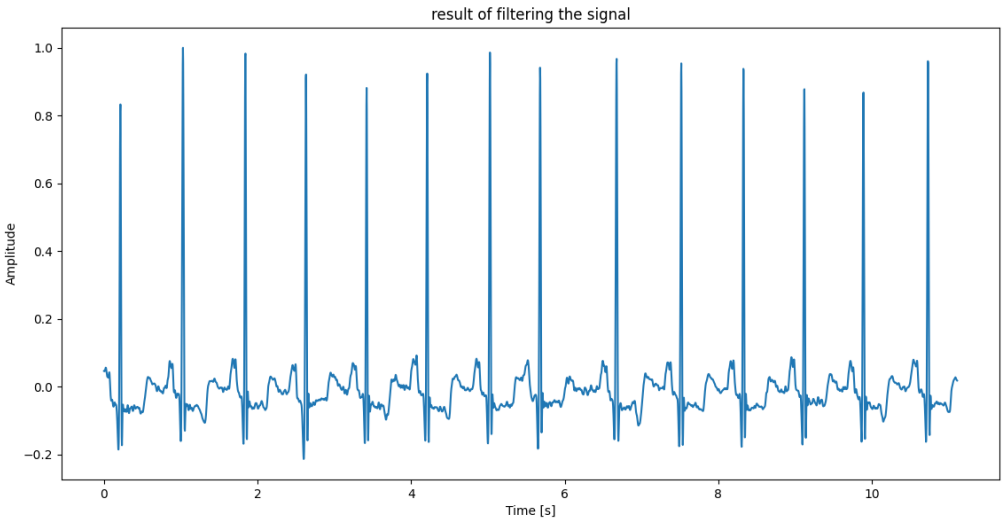


Figure 2.6: Filtered signal of record 100.

Derivative: this step is useful to enlarge the R peak of the QRS complex and reduce the P and T waves appearance. A five-point derivative with transfer function H(z) was applied and the result is shown below.

$$\boldsymbol{H(z) = \frac{1}{8T}(-2z^{-2} - z^{-1} + z^{1} + 2z^{2})} \qquad \textbf{Equation 2.1}$$

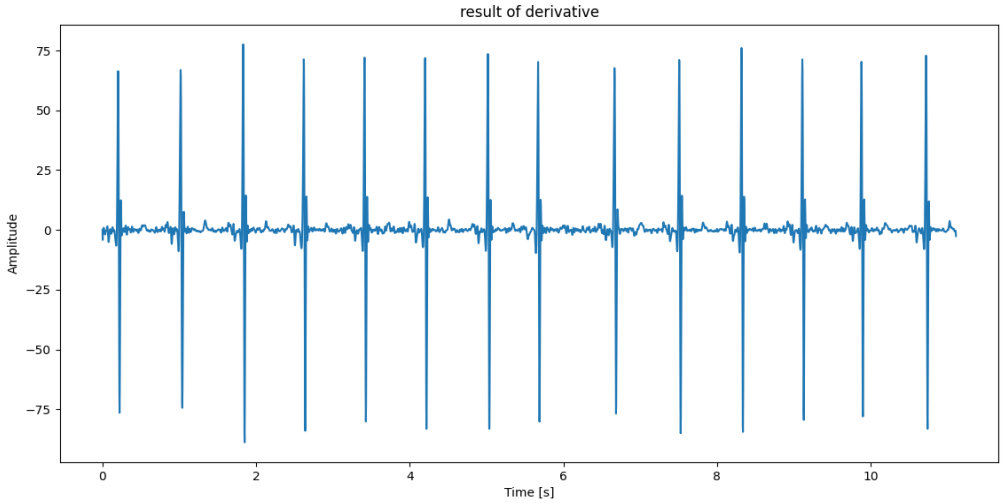


Figure 2.7: Results of applying derivative (record 100).

Squaring: squaring is used to further intensify the slope of QRS complex and eliminate the slopes of T and P waves. The result is shown in figure 2.13.

$$\boldsymbol{y(nT) = [x(nT)]^2} \qquad \textbf{Equation 2.2}$$

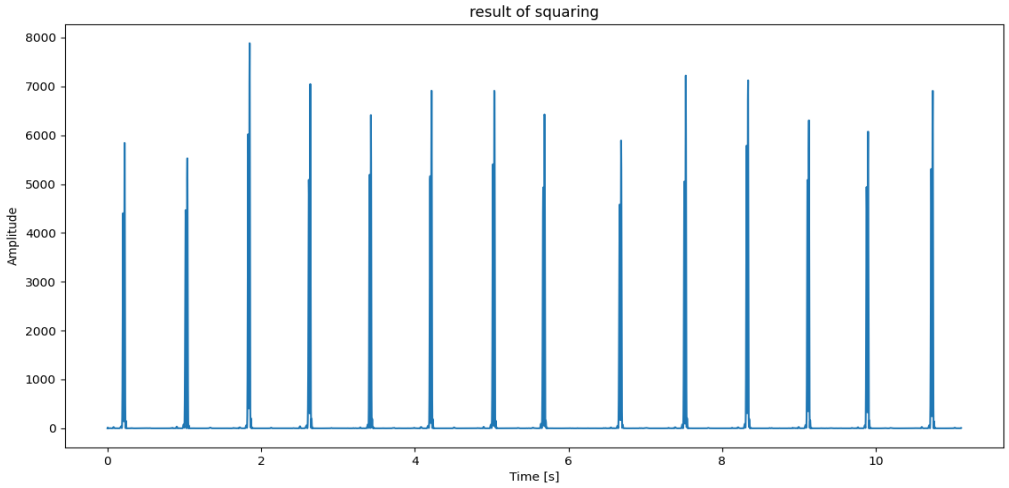


Figure 2.8: Result of squaring (record 100).

Moving window integrator: sometimes, information provided by the slope is not sufficient, for that we need more information about the QRS complexes. This can be achieved using a moving window integrator as shown below.

$$\boldsymbol{y(nT) = \frac{1}{N}[x(nT - (N-1)T) + x(nT - (N-2)T) + \cdots + x(nT)]} \qquad \textbf{\textit{Equation 2.3}}$$

Where N is the number of samples in the width of the integration window, for a sampling frequency of 360samples/second N is chosen to be 54.

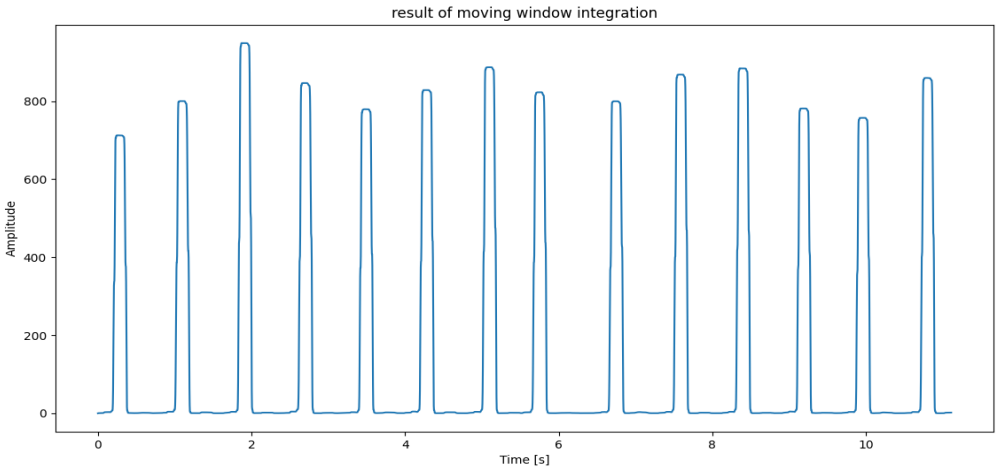


Figure 2.9: Result of moving window integrator (record 100).

2.2.4.2 Decision rule

Previous steps of the algorithm have generated a pulse-shaped waveform, the determination of whether or not a pulse corresponds to a QRS complex is done by the decision rule.

Fiducial mark

The local peaks of the integrated signal are found. A peak is defined as the point in which the signal changes direction within a predefined interval (from an increasing direction to a decreasing direction). After each peak, no peak can be detected in the next 200ms [23].

Adaptive thresholding

Fiducial mark peaks are considered potential QRS, to confirm whether a peak is a QRS peak, adaptive thresholding is performed on both the integrated and filtered signals.

Two thresholds are used for each signal, a signal threshold THRESHOLD I1, and a noise threshold THRESHOLD I2, that are based upon running estimates of the noise and signal levels NPKI, SPKI respectively.

Each potential QRS peak is compared with the thresholds, if PEAK > THRESHOLD I1 then PEAK is considered a candidate QRS peak and the signal level SPKI is updated. If THRESHOLD I2 < PEAK < THRESHOLD I1 then PEAK is considered a noise peak and the noise level NPKI is updated. After updating the noise and signal levels the thresholds are updated as well.

If PEAKI is a signal peak:

$$\boldsymbol{SPKI = 0.125\ PEAKI + 0.875\ SPKI} \qquad \textit{Equation 2.4}$$

If PEAKI is a noise peak:

$$\boldsymbol{NPKI = 0.125\ PEAKI + 0.875\ NPKI} \qquad \textit{Equation 2.5}$$

Updating thresholds:

$$\boldsymbol{THRESHOLD\ I1 = NPKI + 0.25(SPKI - NPKI)} \qquad \textit{Equation 2.6}$$

$$\boldsymbol{THRESHOLD\ I2 = 0.5\ THRESHOLD\ I1} \qquad \textit{Equation 2.7}$$

Where:

SPKI: Signal level in integrated signal.

NPKI: Noise level in integrated signal.

The algorithm implements a further check to confirm whether the candidate QRS peak is an actual QRS peak or not by taking into consideration the information provided by the filtered signal; The corresponding peak of the integrated signal is found in the filtered signal and compared to a threshold in the exact same manner as we did for the integrated signal. If the peak is found to be a QRS peak in the filtered signal then this peak is saved as an actual QRS peak location.

If PEAKF is a signal peak:

$$\boldsymbol{SPKF = 0.125\ PEAKF + 0.875\ SPKF} \qquad \textit{Equation 2.8}$$

If PEAKF is a noise peak:

$$\boldsymbol{NPKF = 0.125\ PEAKF + 0.875\ NPKF} \qquad \textit{Equation 2.9}$$

Updating thresholds:

$$\boldsymbol{THRESHOLD\ F1 = NPKF + 0.25(SPKF - NPKF)} \qquad \textit{Equation 2.10}$$

$$\boldsymbol{THRESHOLD\ F2 = 0.5\ THRESHOLD\ F1} \qquad \textit{Equation 2.11}$$

where:

SPKF: Signal level in filtered signal.

NPKF: Noise level in filtered signal.

Search back for missed QRS complexes

Pan Tompkins algorithm takes into consideration the possibility of missing a QRS complex peak. If a long period has passed without detecting any QRS peak, it is assumed that one has been missed, this reduces the number of false negatives, therefore a search back window is performed. The search back is based upon the assessment of RR intervals; Two RR intervals are calculated in this case considering both regular and irregular heart rhythms, one is the average of the eight most-recent beats. The other is the average of the eight most-recent beats having RR intervals that fall within certain limits.

$$\boldsymbol{RR\ AVERAGE1 = 0.125\ (RR_{n-7} + RR_{n-6} + \cdots + RR_n)} \qquad \textit{Equation 2.12}$$

Where RR_n is the most recent RR interval.

$$\boldsymbol{RR\ AVERGAE2 = 0.125(RR'_{n-7} + RR'_{n-6} + \cdots + RR'_n)} \qquad \textit{Equation 2.13}$$

where RR'_n is the most recent RR interval that fell between the acceptable low and high RR-interval limits.

The RR-interval limits are:

$$\boldsymbol{RR\ LOW\ LIMIT = 92\%\ RR\ AVERAGE2} \qquad \textit{Equation 2.14}$$

$$\boldsymbol{RR\ HIGH\ LIMIT = 116\%\ RR\ AVERAGE2} \qquad \textit{Equation 2.15}$$

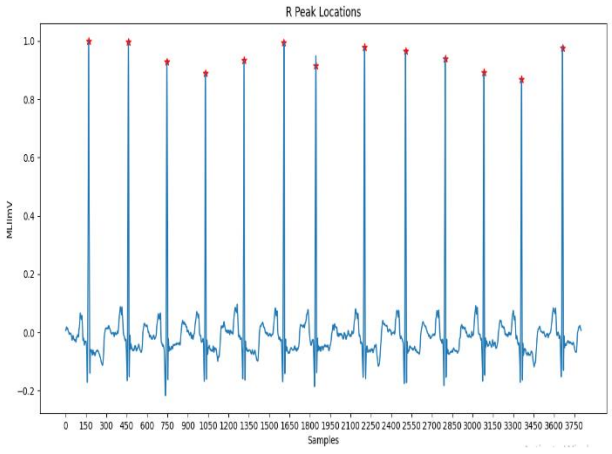
$$\boldsymbol{RR\ MISSED\ LIMIT = 166\%\ RR\ AVERAGE} \qquad \textit{Equation 2.16}$$

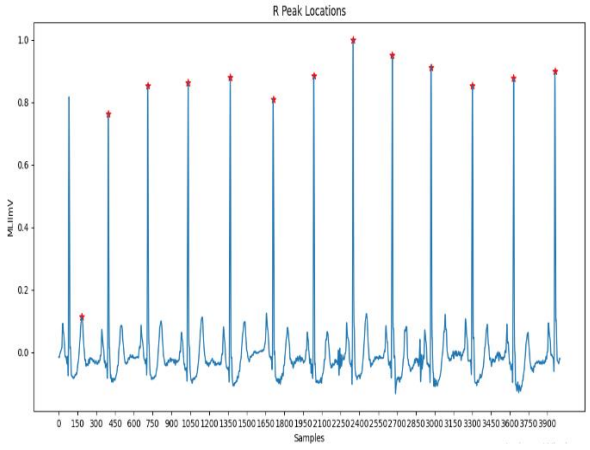
If a QRS complex is not found during the interval specified by the RR MISSED LIMIT, the maximal peak reserved between the two established thresholds is considered to be a QRS candidate. Also, in case of irregular heart rate the threshold is reduced by half.

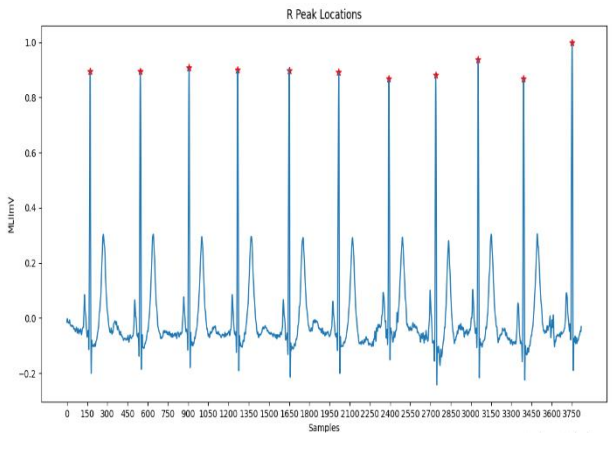
T-Wave Identification

If a QRS detection occurs within 360ms of the previous one i.e., RR_n< 360ms, one must check if it is a QRS or a T wave. If the maximal slope that occurs during this waveform is less than half that of the QRS waveform that preceded it, it is identified to be a T wave; otherwise, it is a QRS complex.

The results of the applied algorithm [24] are shown below for different records of the MIT-BIH arrhythmia database where the R locations are represented by a red cross on the signal.







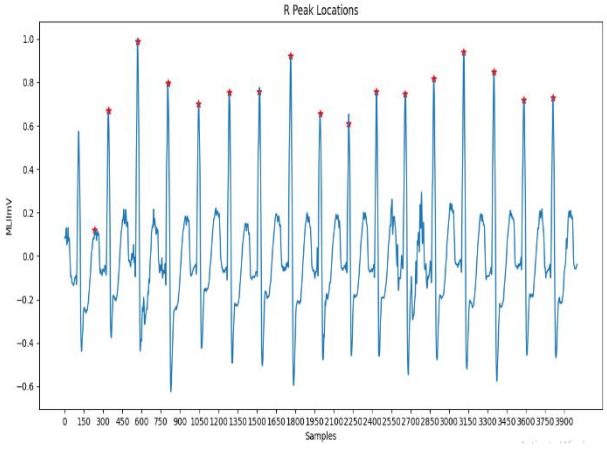


Figure 2.10: R locations on different ECG records.

After localizing the R peaks of the QRS complexes, it is time now to segment the whole ECG signals into individual heartbeats. We defined a single ECG beat by taking 130 samples before the R peak, and 170 samples after the R peak. And then converted each heartbeat into a grayscale image of dimension 200×200, some of the results are illustrated below.

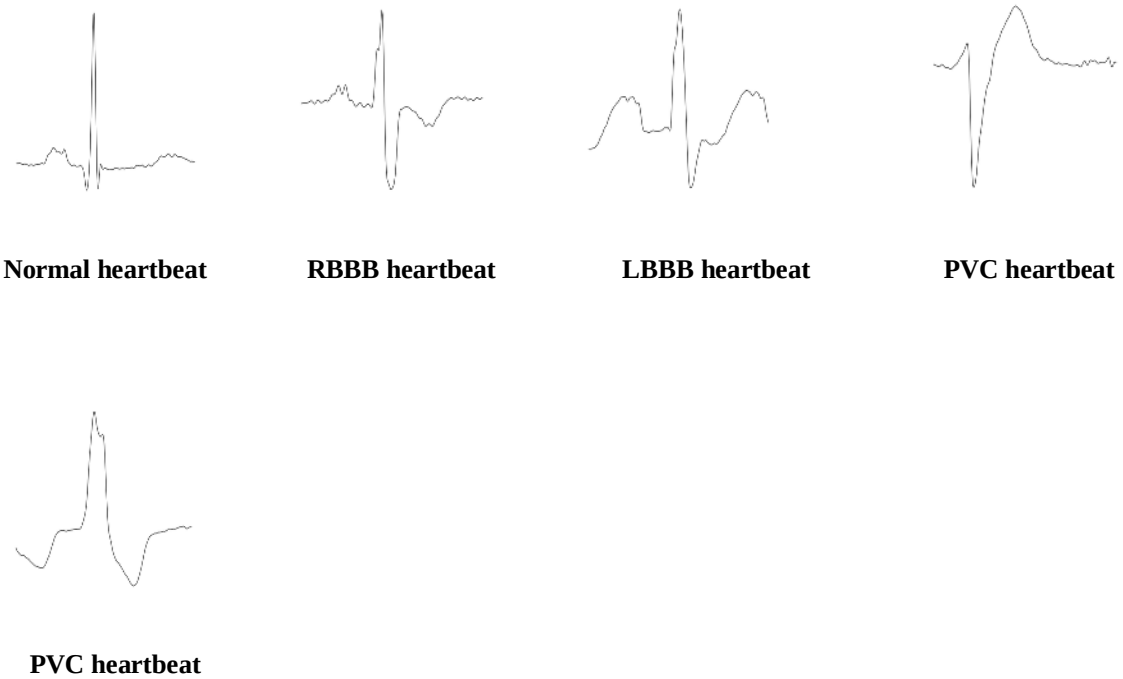


Figure 2.11: Grayscale extracted ECG beats.

2.3 Classification algorithm

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups [25].

Classification is of two types [26]:

- **Binary classification:** When we have to categorize given data into 2 distinct classes. Example On the basis of given health conditions of a person, we have to

determine whether the person has a certain disease or not.

- **Multi class classification:** The number of classes is more than 2. For Example, On the basis of data about different species of flowers, we have to determine which specie does our observation belongs to.

2.4 Convolutional neural networks

CNN is one of the most popular deep learning architectures, it usually deals with image data and is used in variety of applications. The word convolutional in CNN arise from the fact that CNNs use convolution operations in its layers, these convolutional layers help in determining more complex features. depending on the convolution direction CNN can come in 3 different dimensions 1D, 2D, 3D, but when we usually speak of CNNs we are referring to the 2D CNN.

2.4.1 CNN Architecture and working

Similar to ANNs, a CNN is composed of input layer, a set of hidden layers and an output layer; The hidden layers are what make CNNs special, they comprise of a set of convolutional layers+ Rectified linear unit (ReLU) activation functions, pooling layers and finally fully connected layers.

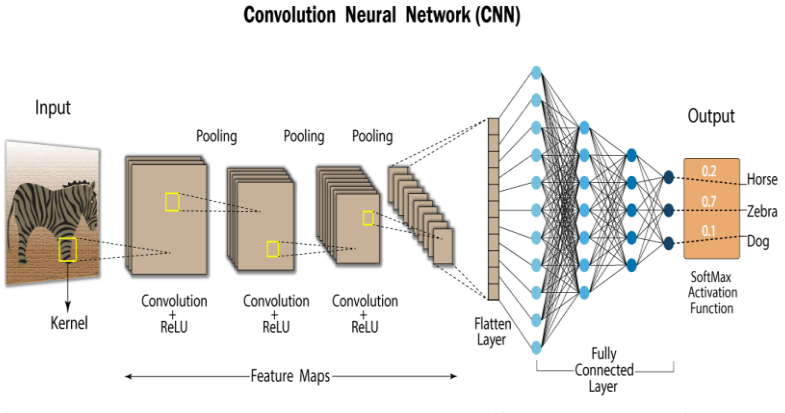


Figure 2.12: CNN architecture for a classification problem [27].

2.4.1.1 Convolutional layers

Convolutional layers are the building block of a CNN model, it is where the features are detected. This layer comes with a number of filters called kernels which represent 2-dimensional array of weights that are initialized and then updated by backpropagation using gradient descent. They are typically smaller than the input image size; Each kernel slide across the image with a certain stride, and in every slide, each value of the kernel gets multiplied by the corresponding pixel in the image and then adds everything together producing one value, this process is repeatedly performed all across the image resulting in a 2-dimensional array called feature map, which represents the detected features from the convolutional layer. Applying a set of filters on a single image result in a set of feature maps called a tensor.

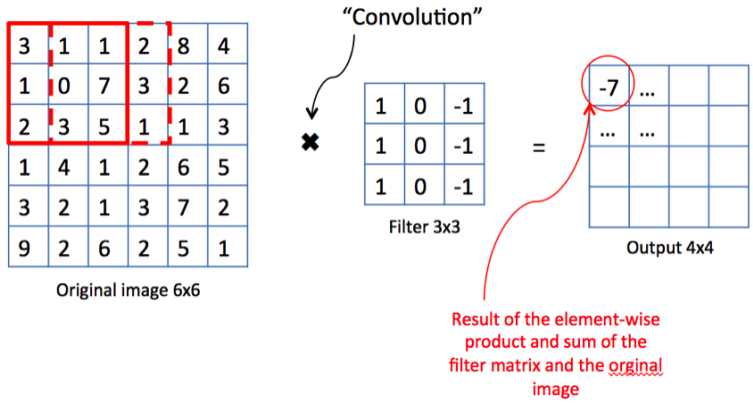


Figure 2.13: Convolution process.

For a given image of dimension $n_h \times n_w$ the convolution operation is given by the equation below:

$$\boldsymbol{Conv(image, kernel)} = \sum_{j=1}^{n_h} \sum_{i=1}^{n_w} \boldsymbol{kernel_{j,i} \times image_{x+j-1,y+i-1}} \qquad \textit{Equation 2.17}$$

Where:

$\boldsymbol{n_h}$: The height of the image.

$\boldsymbol{n_w}$: The width of the image.

Two results can be obtained from this process. One in which features are reduced in dimensionality compared to the input, which is known as "valid padding", the other is increased or remains the same which is known as "same padding"; Padding refers to the number of pixels added to the input image to allow the kernel to cover more space in the image. Valid padding refers to no padding at all, so when the kernel slides along the image, the unreached pixels are dropped resulting in a shrank output size, whereas same padding refers to adding pixels to the borders of the image so that the kernel can fit and cover all the pixels of the image.

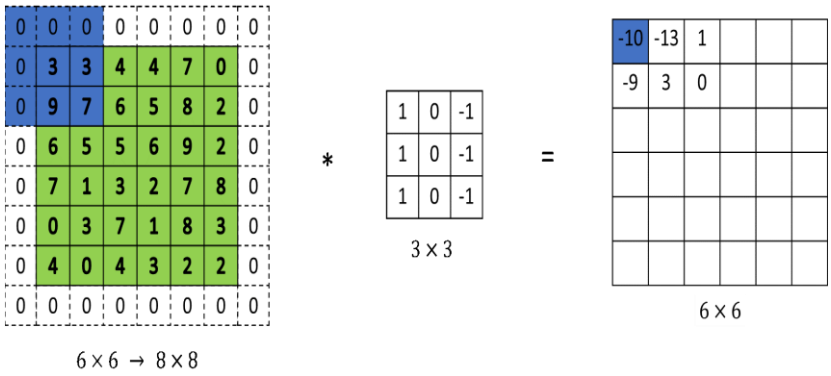


Figure 2.14: Zero padding.

The feature map dimension can be obtained using this formula:

$$\boldsymbol{dim(featuremap)} = \frac{\boldsymbol{n_h - k + 2p}}{\boldsymbol{S}} + \mathbf{1} \times \frac{\boldsymbol{n_w - k + 2p}}{\boldsymbol{S}} + \mathbf{1} \qquad \textit{Equation 2.18}$$

Where:

n_h: the height of the input image.

n_w: the width of the input image.

k: the kernel dimensionality.

p: the amount of padding.

s: the stride, which is the step by which the kernel is sliding across the image.

After the feature maps are created, the next step is to pass these latter through a non-linear activation function usually ReLU function. Considering images are naturally non-linear the goal here is to enhance the non-linearity even further to compensate the linearity caused by the convolution operation [28], making the feature maps more adaptable to real world data.

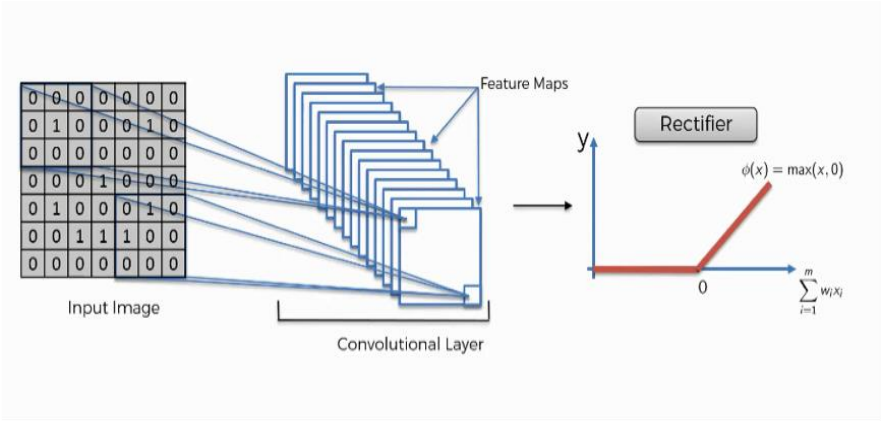


Figure 2.15: Rectification process.

2.4.1.2 Pooling layer

Pooling layer is responsible for dimensionality reduction of the feature map, hence decreasing the computational power required to process the input data. Moreover, it results in extracting dominant features from the images thus, training the model effectively.

Pooling comprises of different types, the most commonly used are max pooling and average pooling. Max pooling returns the maximum pixel value from the portion selected by the corresponding kernel, average pooling on the other hand returns the average from the portion selected by the kernel. The same process is repeated as the kernel slides through the entire feature map resulting in a reduced image dimension which is calculated as follows:

$$\boldsymbol{dim(pool(feature map)) = \frac{n_h - k}{S} + 1 \times \frac{n_w - k}{S} + 1} \qquad \textit{Equation 2.19}$$

Where:

n_h: the height of the input feature map.

n_w: the width of the input feature map.

k:the kernel dimensionality.

s: the stride.

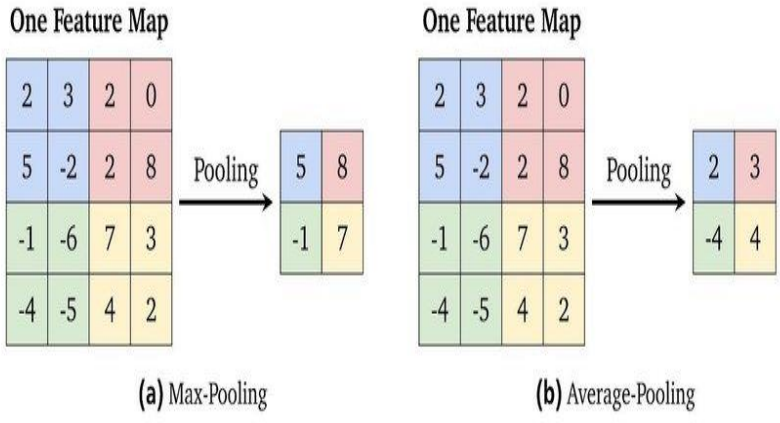


Figure 2.16: Pooling operation.

Combining the convolutional layer + ReLU and the pooling layer creates a single hidden layer. Generally, the first hidden layer is designed to capture low quality features in the image,

nevertheless as we pass through multiple hidden layers, the model becomes sophisticated to detect more complex features. This whole process is known as feature extraction.

2.4.1.3 Fully connected layer

After the feature extraction is done, the next step is learning the non-linear patterns of our data and map the input to the output by means of a classical neural network.

The output of the last pooling layer is flattened and converted into a column vector, and then fed to the neural network, forward and backward passes are done after each iteration over a series of epochs until the model achieves high accuracy.

2.4.2 Batch normalization

The input's distribution of a hidden layer varies as the weights of previous layers are being updated after each iteration. This causes instability in the network as the weights are trying to be adjusted to different distributions each time. This slows down the training and results in an unstable convergence. This phenomenon is known *as Internal Covariate Shift* [29].

Batch normalization was introduced to reduce this problem, it considers normalizing, scaling and shifting the inputs for each chosen layer by normalizing each activation input to have the same distribution over each mini-batch. even if the input values change, their mean and variance will remain same. This reduces the effect of previous layer's changes on the current layer, therefore, makes it learn independently.

Batch Normalizing Transform, applied to activation x over a mini-batch [29]:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 2.17: Batch Normalization process [29].

Where *x* represents a single activation of a certain layer and *B* represents the values of this activation in the mini-batch. In the first two steps, the mean and variance of inputs are calculated over a mini batch, then each input sample gets normalized to have mean of 0 and variance of 1, scaled by γ and shifted by β to set a desired distribution. γ and β are learnable parameters that help to set the distribution.

2.4.2.1 Batch normalization in CNN

In CNN, we use batch normalization directly after the convolutional layer; We calculate the mean and variance for all feature maps across the mini batch.

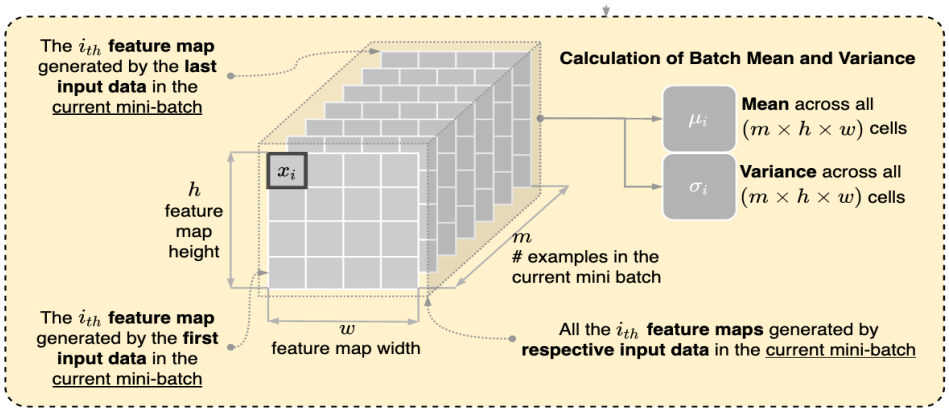


Figure 2.18: Mean and variance of feature map.

the procedure of calculating the mean and variance is shown below:

$$\boldsymbol{mean} = \frac{\mathbf{1}}{\boldsymbol{m \times w \times h}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{h}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{w}} \boldsymbol{featureMap}[\boldsymbol{m, j, k}] \qquad \textit{Equation 2.20}$$

$$\boldsymbol{variance} = \frac{\mathbf{1}}{\boldsymbol{m \times w \times h}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{h}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{w}} (\boldsymbol{featureMap}[\boldsymbol{m, j, k}] - \boldsymbol{mean})^2 \qquad \textit{Equation 2.21}$$

Where:

m: number of examples in the current minibatch.

w: feature map width.

h: feature map height.

2.4.3 Dropout

As the number of layers increase in a neural network, we will be having a large number of weights and biases to learn which causes overfitting, which means that the model learns so many details about our training data that it performs poorly on unseen data, this can be observed when the difference between testing and training accuracies is high. One way to solve this issue is by using dropout. The term "dropout" refers to randomly dropping out units in a neural network by a probability of p called the dropout rate, resulting in a network that contains only a subset of the original network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections [30]. A dropout rate of 0.1 means that each neuron has 10% chance to be dropped, similarly a dropout rate of 1 means that all units are dropped and a rate of 0 means that all units are kept. Figure 2.19 illustrates the dropout layers.

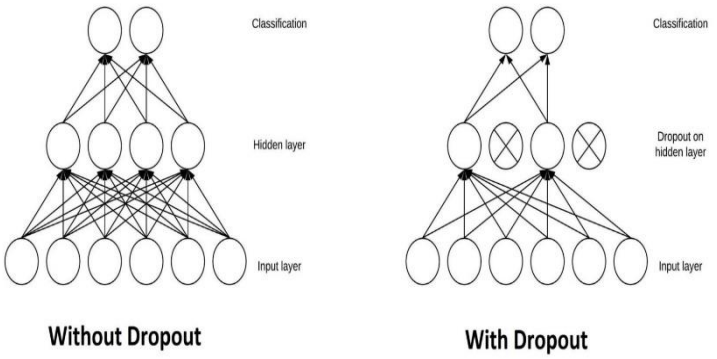


Figure 2.19: Dropout [30].

Dropout is only used for training data; in test data all neurons are activated and the weights of the layer we applied dropout to are multiplied by the dropout rate *p*. The reason for that is that when neurons are all activated in test time, it puts the output neurons in an unusual regime producing too large values, so the trick of reducing the weight values by p is used to prevent this overexcitement from happening.

2.4.4 Evaluation metrics

Different evaluation metrics exist to assess a deep learning model, one of them being the confusion matrix. Confusion matrix is an N×N matrix representing how well the model is performing, mainly used for classification problems where N is the number of classes. It compares the predicted values of the model with the actual target values and gives an overview

about the correct predictions and false predictions. Figure 2.20 represents a confusion matrix for binary classification.

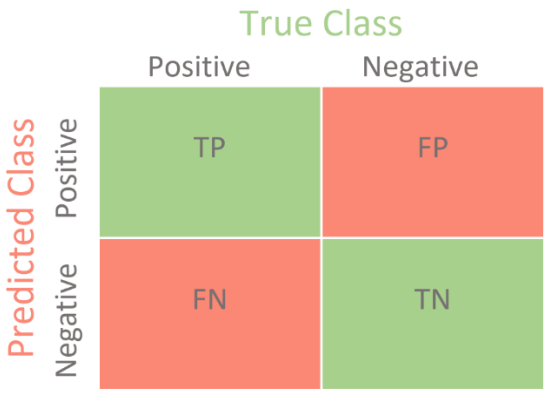


Figure 2.20: Confusion matrix for binary classification.

To better understand this let's consider a binary classification of whether a patient has a heart disease or not:

True positive (TP): is when the model has predicted that the patient has a heart disease and he actually does have a heart disease.

True negative (TN): is when the model has predicted that the patient does not have a heart disease and he actually does not have a heart disease.

False positive (FP): is when the model has predicted that the patient has a heart disease when he actually does not. Also known as type I error.

False negative (FN): is when the model has predicted that the patient does not have a heart disease when in fact, he does have a heart disease. Also known as type II error.

The performance of the model is assessed using what we call *key performance indicators (KPI):*

Classification accuracy: (TP+TN)/(TP+TN+FP+FN).

Misclassification rate: (FP+FN)/(TP+TN+FP+FN).

Precision: TP/(TP+FP), it measures the model's accuracy in classifying a sample as positive.

Recall (sensitivity): TP/(TP+FN), it measures the model's ability to detect Positive samples.

The above example was for the case of binary classification, but what about multiclass classification? How are TP TN FP FN calculated?

Let's consider the confusion matrix for multi class classification shown in figure 2.21:

		Predicted			
		C_1	C_2	⋯	C_n
Actual	C_1	N_{11}	N_{12}	⋯	N_{1n}
	C_2	N_{21}	N_{22}	⋯	N_{2n}
	⋮	⋮	⋮	⋮	⋮
	C_n	N_{n1}	N_{n2}	⋯	N_{nn}

Figure 2.21: Confusion matrix for multi class classification.

In this case we are trying to classify n classes, C_1, C_2…. C_n, hence, we have n×n matrix.

N_{11}represents actually C_1classified as C_1, N_{22} represents actually C_2classified as C_2, N_{nn} represents actually $C_n$classified as C_n, therefore, we can conclude that the diagonal values represent the true positives TP for each class.

Similarly, N_{12} represents actually C_1classified as C_2, N_{21} represents actually C_2 classified as C_1, and so on. Hence, the values off the diagonal are the errors.

The total number of text samples for each class is calculated by summing all values of the corresponding row i.e., total number of samples of class $C_1 = N_{11} + N_{12} + \cdots + N_{1n}$.

The total number of false negatives FN of a class is the sum of all values in the corresponding row excluding the true positive.

The total number of false positives FP of a class is the sum of all values in the corresponding column excluding the TP.

The total number of true negatives TN of a class is the sum of values of all rows and columns excluding values of corresponding row and column.

After calculating TP, FP, TN, FN of each class, we can continue to calculate the key performance indicators for each class in the same way we did with binary classification.

2.5 Conclusion

In this chapter, we went through the different preprocessing steps to get our ECG data to be ready for further processing by a neural network; we introduced the concept of convolutional neural networks and the convolution operation, along with different deep learning techniques used to enhance the performance of a CNN, finally we explained the theory behind confusion matrices and how this latter is used to evaluate a deep learning model.

Chapter 3

Results and discussion

3.1 Data collection

Different records from the MIT BIH arrhythmia database were selected to extract the heartbeats of the four different classes (N, LBBB, RBBB, PVC). We ensured that for each class the same number of heartbeats was extracted, which is around 5000 samples per class. This is in order to create a balanced dataset for our models. Table below shows the records used and the number of data samples acquired per each class. Table 3.1 demonstrates the records used to retrieve samples of each class.

Classes	Records	№ of samples
N	100,101,103,105,106,108,112,114,115, 116,119,121,123,200,201,202,203,205, 209,210,219,221,222,223,231.	5000
LBBB	109,111,214.	4795
RBBB	118,124,212.	5000
PVC	109,105,106,116,118,119,124,200,201, 203,204,205,208,210,215,219,221,223, 228,233.	5000

Table 3.1:Records used to retrieve samples of each class.

As mentioned in the previous unit, the heartbeats were converted into grayscale images; The images and the corresponding labels were saved in separate HDF5 files **data.hdf5** and **label.hdf5**.

3.2 Data preparation

Our data can't be fed directly to the neural network, some preparation needs to be done including splitting, reshaping, and normalization.

We first read our data, then using the **train_test_split** method from the **sklearn** library we split it into a training and a testing set with a ratio of 80/20. Table 3.2 shows the number of samples per class.

Classes	Trainset (№ of samples)	Testset (№ of samples)
N	4005	995
LBBB	3828	972
RBBB	3996	1004
PVC	4011	989

Table 3.2: Training and testing datasets.

3.3 Experimental results

A 2D CNN model was proposed to classify the ECG heartbeat images into four different categories. After trying many models with different layering structure, we found that the architecture demonstrated in table 3.3 below works best for our classification.

	Type	**kernel size**	**stride**	**number of kernels**	**Input size**
Layer 1	Conv2D+ReLU	3×3	1	16	200×200×1
Layer 2	Conv2D+ReLU	3×3	1	16	200×200×16
Layer 3	AveragePooling2D	2×2	2		200×200×16
Layer 4	Conv2D+ReLU	3×3	1	16	100×100×16
Layer 5	Conv2D+ReLU	3×3	1	16	100×100×16
Layer 6	AveragePooling2D	2×2	2		100×100×16
Layer 7	Conv2D+ReLU	3×3	1	16	50×50×16
Layer 8	Conv2D+ReLU	3×3	1	16	50×50×16
Layer 9	AveragePooling2D	2×2	2		50×50×16
Layer 10	Fully connected (ReLU)			512	25×25×16
Layer 11	Fully connected (ReLU)			512	512
Layer 12	Out (SoftMax)			4	512

Table 3.3: Architecture of proposed model.

The performance of the above model was assessed for three different approaches, one being CNN without Batch normalization (BN) and dropout; Second being CNN with BN, in which we placed BN layers after each convolutional layer, lastly CNN with dropout where we used

two dropout layers after each fully connected layer and trained the model for different dropout rates.

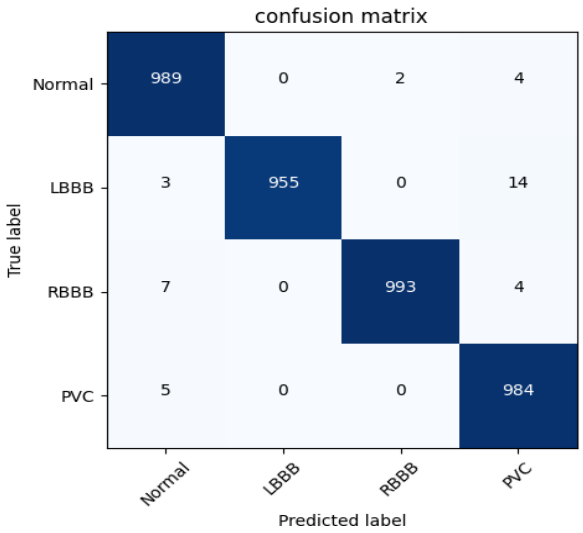
As hyperparameters the model was compiled with cross entropy loss function, Adam optimizer with learning rate of 0.0001 and a batch size of 32. The assessment was based on training and testing accuracies as well as training time. The results obtained for each approach are reported in tables 3.4 and 3.5, and confusion matrices in figure 3.1.

Model	**Epochs**	**Training time(s)**	**Training accuracy (%)**	**Testing accuracy (%)**
Without BN And dropout	10	3940	99.97	99.02
With BN	5	2698	99.98	99.32

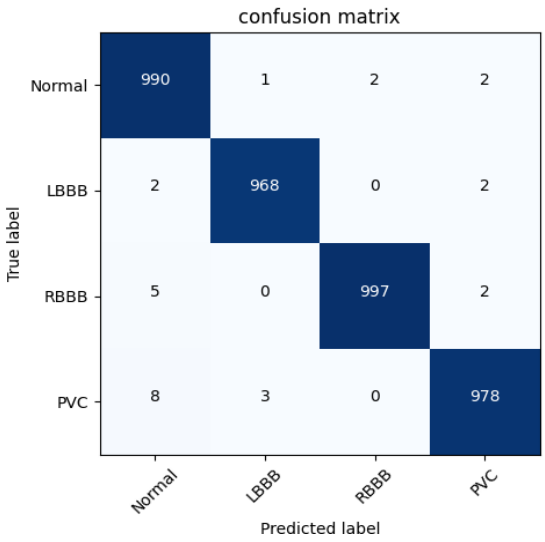
Table 3.4: Results of CNN with and without BN.

Dropout rate	**Epochs**	**Training accuracy (%)**	**Testing accuracy (%)**
0.1	10	99,82	99,22
0.2	10	99,81	99,29
0.3	**10**	**99,85**	**99,47**
0.4	10	99,79	99,34
0.5	10	99,71	99,29
0.6	10	99,60	99,24
0.7	10	99,36	99,17
0.8	10	98,74	99,09
0.9	10	98,62	98.68

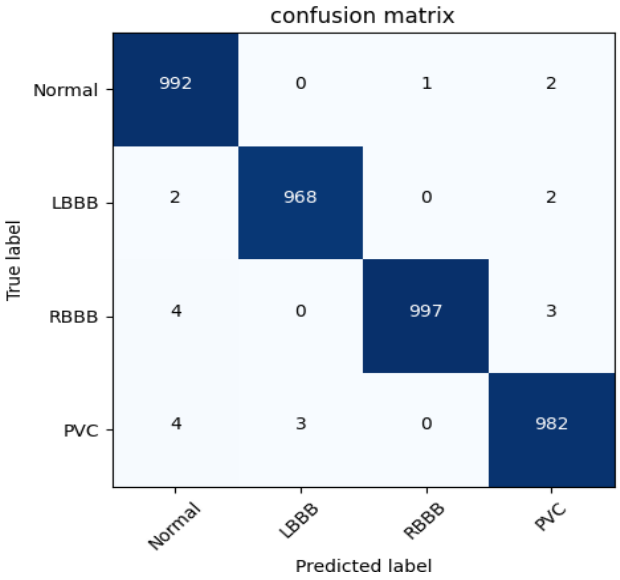
Table 3.5: Results of CNN with dropout.



Confusion matrix of CNN without BN and dropout.



Confusion matrix of CNN with BN.



Confusion matrix of CNN with dropout.

Figure 3.1: Confusion matrices of the different models.

3.4 Discussion of the results

A 2D CNN model was suggested in this study for ECG heartbeat arrythmias classification, three approaches were evaluated, a CNN without BN and dropout, CNN with BN and CNN with dropout. In this section of the chapter, we discuss the results obtained from the three different techniques.

The results recorded in table 3.4 demonstrated the performance of the CNN model and CNN with BN model, where BN layers placed in between the conv2D and ReLU layers. The CNN model has achieved a training accuracy of 99.97% and a testing accuracy of 99.02% after 10 epochs of training, meanwhile for only 5 epochs of training on the CNN with BN model attained the same training accuracy of 99.98%. This highlights the effect of BN layers in accelerating the learning process and leading to faster convergence; As the training times of both architectures indicate, introducing the BN layers decreased from 3940 seconds(1h05min) to 2698 seconds(44min), in this way, we saved time of almost 21min in our training.

Another promising finding is that a substantially better testing accuracy has been observed with a percentage of 99.32%, and that BN offers some regularization effect, meaning it improves the generalization of our model, while the difference between training and testing accuracies of the CNN model was around 1%, with BN it got reduced down to 0.66%.

The above observations are a proof that BN has multiple benefits in a deep learning model from improving learning time to improving generalization. By normalizing the inputs to each layer over the minibatch, we give equal importance to each input variable, thus, we reduce the internal covariate shift, which results in a more stable and fast training.

One thing to note is that our CNN without BN and dropout model does not overfit much (difference of 1% only between train and test accuracies), since we could provide an enough amount of data to the model, however, a further improvement could be done. One way of reducing this difference is by applying a regularization technique like dropout. Table 3.5 demonstrates the training and testing accuracies for different dropout rates for 10 epochs of training; The best accuracies were achieved with a dropout rate of 0.3 where training accuracy was 99.85% and 99.47% testing accuracy with a difference of 0.38%.

Dropping out random neurons by a certain fraction while training prevents the model from learning the noise and the details of the training data, which results in a more generalized model. As the dropout rate increases the accuracy decreases which is logical since as we increase the rates, we are decreasing the number of units that contribute in the learning thus resulting in a network that lacks sufficient neurons to model the input output relationship.

The resulting confusion matrices shown on figure 3.1, represent a summary of prediction results on our classification problem, where for each class it keeps count of the number of correct and incorrect predictions. The outcome of the matrices indicates that our models could perform very well on the prediction of the four classes equally. The misclassification rates of the different models are shown in table below.

Classes	CNN without BN And dropout	CNN with BN	CNN with dropout (0.3)
N	5.30e-3	5.05e-3	3.28e-3
LBBB	4.29e-3	2.02e-3	1.76e-3
RBBB	3.28e-3	2.27e-3	1.76e-3
PVC	6.81e-3	4.29e-3	3.78e-3

Table 3.6: Misclassification rates for the different models.

Table 3.6 above provides information about the percentage of observations that were incorrectly predicted by the models. In our study correctly predicting the arrhythmias is important, since patient's health depends on it, for that, we try to choose a model with the least misclassification rate, in our case the CNN with dropout model resulted in the least rate, therefore it was selected as the best model for the classification.

3.5 Conclusion

Dropout and batch normalization are two powerful deep learning techniques used for improving model performance and generalization, the CNN without BN and dropout resulted in a good accuracy of 99.02%, but further improvements could be achieved with BN and dropout. The CNN with dropout model is the well performing one achieving a testing accuracy of 99.47% and the lowest misclassification rate.

General conclusion

In this report, a CNN based model was implemented and trained for classification of ECG heartbeat arrythmias into four different classes (Normal beat, LBBB, RBBB, PVC). The model has been trained and tested using records from MIT BIH arrythmia database, that were filtered, segmented and converted into grayscale images of dimension 200×200 before being fed to the neural network. Also, the effect of dropout and batch normalization have been examined in terms of accuracy and training time.

We provided an introductory about electrocardiography, where we saw that any distortion in a heart's beating mechanism will lead to arrythmia, which can be detected by monitoring and diagnosing ECG signals. We also got acquainted with deep learning theory and how do neural networks learn with the help of backpropagation and gradient descent.

After that, Pan Tompkins algorithm was introduced which is the most widely used QRS detection method that applies a series of filters to highlight the characteristics of an ECG signal followed with applying adaptive thresholding technique to detect the R peaks, these latter are used as a reference to segment our signals into individual heartbeats. Moreover, 2D CNNs architecture and its consisting layers were explained along with two deep learning techniques which are batch normalization and dropout.

The experimental results are finally obtained by implementing, training and evaluating our CNN architecture, at first the model trained for 3940 seconds to finally achieve a testing accuracy of 99.02%; By adding batch normalization after each Conv2D layer, the resulting model trained for 2698 seconds resulting in a testing accuracy of 99.32%, this highlights the effect of BN in improving the training time as well as model generalization. We trained again the same original model and added dropout layers in between the fully connected layers, for different dropout rates, it was found that the rate of 0.3 works better in our case, as it achieved 99.47% testing accuracy along with the best regularization effect and lowest misclassification rate.

References

[1] A. Singh, «ECG arrhythmia classification using a 2-D,» DataDrivenInvestor, 2018.

[2] Nahom Ghebremeskel1 ,Vahid Emamian2, «CLASSIFICATIONOF CARDIAC ARRHYTHMIA USING A 2 D CONVOLUTIONAL,» International Journal of Recent Scientific Research, 2019.

[3] XUEXIANG XU , HONGXING LIU, «ECG Heartbeat Classification Using,» IEEE Access, 2020.

[4] «Basics of the Heart,» [Online]. Available: elitecardiovascular.com. [Accessed 2022].

[5] Oxford Languages, "Electrocardiography".

[6] M. Cadogan, "PR Interval," [Online]. Available: litfl.com. [Accessed march 2022].

[7] "Segments vs. Intervals in an ECG," [Online]. Available: timeofcare.com. [Accessed march 2022].

[8] «Electrocardiography,» [Online]. [Acessed 2022].

[9] "Understanding the 12-Lead ECG System, with Animation," 26 september 2016. [Online]. Available: alilamedicalimages.org. [Accessed march 2022].

[10] Marschall S. Runge, Cam Patterson, and George Stouffer, Netter's Cardiology, 2006.

[11] "Arrhythmia," [Online]. Available: my.clevelandclinic.org. [Accessed march 2022].

[12] J. FRANKENFIELD, "Artificial Intelligence (AI)," 08 march 2021. [Online]. Available: investopedia.com. [Accessed april 2022].

[13] N. Rowe. [Online]. Available: morioh.com. [Accessed 2022].

[14] K. Suzuki, Artificial Neural Networks-Architectures and Applications, IntechOpen , 2013.

[15] [Online]. Available: en.wikipedia.org/wiki/Gradient_descent. [Accessed le 2022].

[16] S. Kansal, «Quick Guide to Gradient Descent and Its Variants,» 2020.

[17] «MIT-BIH Arrhythmia Database,» [Online]. Available: physionet.org.

[18] Unknown. [Online].

[19] Liangling Gu,Nanquan Zhou,Haotian Wu, «Application of Interference Canceller in Bioelectricity Signal Disposing,» 2011.

[20] J. S. Karnewar, V. K. Shandilya, «Preprocessing ECG signal by eliminating various noises using denoising methods,» AIP Conference Proceedings, 2022.

[21] Falco Strasser, Michael Muma, A. Zoubir, «Motion artifact removal in ECG signals using multi-resolution thresholding,» semantic scholar, 2012.

[22] JIAPU PAN , WILLIS J. TOMPKINS, «A Real-Time QRS Detection Algorithm,» IEEE, 1985.

[23] «Pan–Tompkins algorithm,» [Online]. Available: en.wikipedia.org. [Accessed 2021].

[24] antimattercorrade, «Pan Tompkins QRS Detection,» 2021.

[25] «Getting started with Classification,» [Online]. Available: geeksforgeeks.org. [Accessed 2022].

[26] «Classification Algorithm in Machine Learning,» [Online]. Available: javatpoint.com. [Accessed 2022].

[27] S. K. E, «Convolutional neural networks».

[28] S. team, «Convolutional Neural Networks (CNN): Step 1(b) - ReLU Layer,» 17 August 2018. [Online]. Available: superdatascience.com. [Accessed may 2022].

[29] Sergey Ioffe, Christian Szegedy, «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,» arxiv, 2015.

[30] Nitish Srivastava nitish,Geoffrey Hinton,Alex Krizhevsky,Ilya Sutskever,Ruslan Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting,» *Journal of Machine Learning Research,* 2014.

[31] «Difference between IIR and FIR filters: a practical design guide,» april 2020. [Online]. Available: advsolned.com. [Accessed 2022].

[32] Daniel Ho, Eric Liang, Richard Liaw, 07 june 2019. [Online]. Available: bair.berkeley.edu.

[33] J. S. Karnewar , V. K. Shandilya, «Preprocessing ECG signal by eliminating various noises using denoising methods,» AIP Conference Proceedings, 2022.