People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronics Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of

The requirement for the degree of

Master

In **Electrical and Electronics Engineering**

Option: **Computer Engineering**

Title:

Cardiovascular Diseases Detection from Phonocardiograms Using Deep Learning

Presented by:

Aymen Abderraouf RAHMANI

Supervisor:

Dr. Elhocine BOUTELLAA

Co-supervisor:

Dr. Rachid NAMANE

Registration Number: 2023

Dedication

I wholeheartedly dedicate this work to my parents, whose unwavering support and guidance have been a constant source of strength throughout my entire life. From teaching me as a child to shaping the person I am today; their love and dedication have been immeasurable.

I also extend my dedication to my brothers, friends, and family members who have stood by my side, offering encouragement and motivation along this journey.

Lastly, I would like to pay tribute to my aunt, Rahmani Nacera, who played a significant role in my childhood and education. Though she is no longer with us after her untimely passing in 2022, her teachings and influence will forever be cherished in my heart.

Acknowledgment

I would like to extend my heartfelt gratitude to my supervisors, Dr. NAMANE Rachid and Dr. BOUTELLAA Elhocine, for their invaluable guidance and support throughout the completion of my final year project titled "Cardiovascular Diseases Detection from Phonocardiograms using Deep Learning" and for their contributions in enhancing this report.

I would also like to express my appreciation to all my friends and colleagues who generously shared their time, insights, and assistance, which proved helpful in the successful completion of this project.

Abstract

Cardiovascular diseases are a significant public health concern, responsible for a high number of global deaths. Manual diagnosis of CVDs using heart sound signals requires extensive clinical expertise. In recent years, researchers have explored signal processing and machine learning techniques to automate the early detection of cardiovascular diseases from Phonocardiograms. However, the majority of these approaches depend on traditional features and classifiers, which may experience difficulties capturing the complexity of heart sounds.

This study aims to develop a deep learning model capable of accurately classifying heart sounds as normal or abnormal. Making use of the publicly available PhysioNet 2016 dataset to train a hybrid CNN-LSTM model, a comprehensive comparison between different sound segmentation (windowed segments and heart cycle segments) and feature extraction techniques (Mel Spectrograms and Mel Frequency Cepstrum Coefficients) are conducted. The goal of this comparative study is to identify the optimal combination of segmentation and feature extraction methods to effectively represent heart sounds for efficient training of the adopted deep neural network architecture. We achieved an overall final score of 93% and an accuracy of 92% using the heart cycle segments and spectrogram features setting. Performance comparisons with the existing literature indicate the efficiency of this approach. This research aims to contribute to the advancement of automated CVD detection from Phonocardiograms, potentially aiding in early diagnosis and intervention.

Contents

List of Figures

List of Tables

List of Acronyms

CVD	Cardiovascular Disease
WHO	World Health Organization
PCG	Phonocardiogram
AV	Atrioventricular
SL	Semilunar
SA	Sinoatrial
ECG	Electrocardiogram
MRI	Magnetic Resonance Imaging
CM	Continuous Murmur
MSM	Mid-Systolic Murmur
EDM	Early-Diastolic Murmur
MDM	Mid-Diastolic Murmur
PSM	Pre-Systolic Murmur
EDM	Early-Diastolic Murmur
LSM	Late-Systolic Murmur
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GPU	Graphical Processing Unit
GD	Gradient Descent
SGD	Stochastic Gradient Descent
MGD	Minibatch Gradient Descent
ADAM	Adaptive Moment Estimation
MFCC	Mel Frequency Cepstrum Coefficients
FFT	Fast Fourier Transform
STFT	Short Time Fourier Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
RR	R-R Interval
MLP	Multilayer Perceptron
CRNN	Convolutional Recurrent Neural Network
PRCNN	Parallel Recurrent Convolutional Neural Network
GRU	Gated Recurrent Units
BN	Batch Normalization
SMOTE	Synthetic Minority Oversampling Technique
DCNN	Deep Convolutional Neural Network
RPS	Reconstructed Phase Space
SVM	Support Vector Machine
ECOC	Error Correcting Output Codes
TPR	True Positive Ratio
CVT	Convolutional Vision Transformer
UAR	Unweighted Average Recall
API	Application Programming Interface
TP	True Positive
TN	True Negative

FP	False Positive
FN	False Negative
MAcc	Mean Accuracy

Introduction

Cardiovascular diseases (CVDs) continue to be the leading cause of global death, tragically claiming approximately 17.9 million lives annually, as reported by the World Health Organization (WHO) [1]. Heart attacks and strokes account for over four-fifths of all CVD-related deaths, with a significant portion occurring prematurely in individuals under the age of 70 [1]. However, the impact of CVDs extends beyond the loss of life. It also burdens individuals in low- and middle-income countries with significant financial difficulties. Therefore, early detection and diagnosis of CVDs are crucial in order to effectively reduce the mortality rate [2].

Cardiac auscultation, a simple yet essential method, proves to be an efficient approach for examining cardiovascular diseases [2]. The heart sound signals obtained through this method carry valuable early pathological information, facilitating early detection of CVDs [3]. These heart sounds, commonly referred to as Phonocardiograms (PCG), are acquired in a noninvasive and easily accessible manner [4]. Their noninvasiveness and ability to accurately reflect the mechanical motion of the heart and cardiovascular system makes them important in the early diagnosis of CVDs [5].

However, Cardiac auscultation requires significant clinical experience and expertise, besides it is limited by the human ear's ability to perceive different frequencies in the heart sounds. This limitation makes it challenging to diagnose certain conditions. Consequently, there is great potential for computer-based automatic analysis and classification of heart sound signals to advance the field of human health management [5]. Particularly, with the recent progress in artificial intelligence algorithms, deep learning algorithms have been investigated for heart sound classification. Unlike traditional algorithms, deep learning algorithms possess the capability to automatically extract features from complex heart sound signals [3]. Making them a preferred choice in enhancing the accuracy and efficiency of heart sound classification.

The primary objective of this research is to develop an effective detection model for cardiovascular diseases using PCG signals. This will be achieved by conducting a comprehensive comparative study on heart sound segmentation techniques and audio feature extraction methods. The structure of this report is as follows: the first chapter provides a theoretical background, covering topics such as cardiovascular diseases, diagnosis techniques, deep learning theory, and the PCG classification process. In the second chapter, various related works are presented, and their findings are discussed. The third chapter outlines the methodology and the proposed approach employed in this research. The final chapter presents the results obtained, compares them, and provides a thorough discussion. The report concludes with a general conclusion and visions for future research works.

Chapter 1: Theoretical Background

1.1 Introduction

This chapter serves as an introduction to the report, providing an overview of Cardiovascular Diseases, Phonocardiography, and Deep learning theory. The chapter begins by briefly introducing the physiology of the heart and proceeds to discuss the different types of cardiovascular diseases along with their symptoms and statistics. The chapter also gives a brief overview of Phonocardiography techniques used in the diagnosis of heart diseases. Lastly, the chapter provides a brief introduction to the principles underlying deep learning, including the concepts of neural networks, their learning process, and the steps of audio classification.

1.2 Cardiovascular Diseases Overview

1.2.1 Heart Anatomy

The heart is the primary organ of the human cardiovascular system, a network of blood vessels that pumps blood throughout the body. It works with other body systems to control the heart rate and blood pressure [6]. The heart contains Walls, Chambers, Valves, Blood Vessels, and an Electrical Conduction System as can be seen in Fig.1.

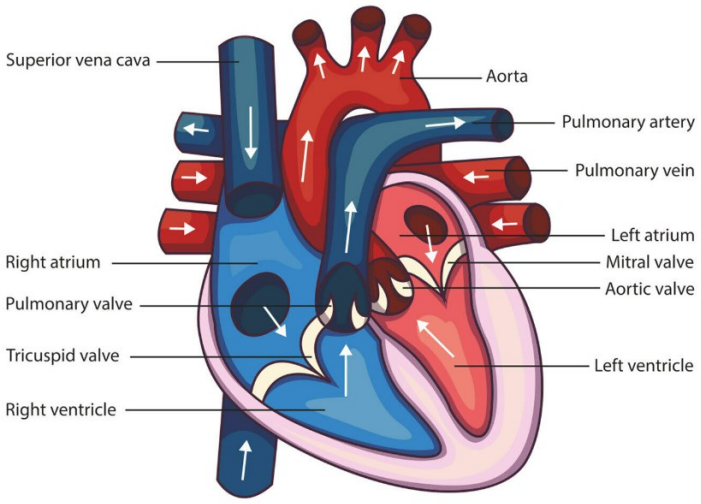


Figure 1: Chambers and Vessels of the heart [7].

The Heart Walls: The heart's function of regulating blood flow is carried out by the contraction and relaxation of its walls. The septum, a layer of muscular tissue, divides the heart walls into left and right sides. The walls of the heart are composed of three layers: the innermost layer called the Endocardium, the middle muscular layer called the Myocardium, and the outermost layer known as the Epicardium. The Epicardium is part of the Pericardium, a protective sac that encases the heart, and serves to prevent friction with surrounding organs by producing lubricating fluids.

Heart Chambers: The heart is comprised of four chambers, two on top and two on the bottom, divided into two sides. These chambers function to both receive and pump blood out of the heart. Initially, deoxygenated blood travels from the Superior and Inferior vena cava and accumulates in the right atrium. It is then transmitted to the right ventricle, which pumps it

through the Pulmonary artery directly to the lungs where the blood is oxygenated. Following this, oxygenated blood returns to the heart from the Pulmonary vein, entering the Left atrium. The Left atrium then propels the blood to the Left ventricle, which subsequently pumps the oxygenated blood throughout the body via the aorta.

Heart Valves: The heart has valves located between its chambers (as shown in Fig.1) that regulate the flow of blood. There are two main types of valves: the Atrioventricular (AV) valves and the Semilunar (SL) valves. The AV valves are positioned between the upper and lower chambers of the heart and consist of the Tricuspid and Mitral valves. The SL valves open when blood flows out of the ventricles and consist of the Aortic valve located between the Left Ventricle and Aorta, and the Pulmonary valve located between the pulmonary artery and the right ventricle.

Blood Vessels: The great vessels of the heart are the aorta, pulmonary artery, pulmonary veins, and the superior and inferior vena cava, which are directly connected to the heart. These arteries and veins play a vital role in the circulatory system, they send blood between the heart and the lungs (Pulmonary circuit) and between the heart and the body (Systemic circuit) [8].

Heart's Electrical Conduction System: The heart's pumping action is regulated by a network of nodes, cells and signals that coordinate the contraction of the various chambers of the heart [9]. The Electrical Conduction System also known as the Cardiac Conduction System consists of:

- **Sinoatrial node:** the SA node acts as a natural pacemaker of the heart; it creates an excitation signal which travels to the atria causing it to contract.
- **Atrioventricular node:** the AV node delays the signal until the atria is fully contracted and empty of blood before the impulse is transmitted to the ventricles.
- **Bundle of His:** It carries the impulse signal from the AV node to the Purkinje fibers. It also divides into right and left pathways to stimulate the right and left ventricles [9].
- **Purkinje fibers:** A network of fibers that distribute the impulse to the muscle cells of the ventricles, causing them to contract and pump blood out of the heart [10].

1.2.2 Cardiovascular Diseases

Cardiovascular diseases commonly referred to as heart disease and stroke are a group of disorders that impact the heart and blood vessels, leading to a range of complications. These diseases are a significant public health concern and are responsible for a large number of deaths globally each year. There are various forms of cardiovascular diseases, including coronary heart disease, stroke, heart failure, and arrhythmias. Many of these diseases are linked to lifestyle factors such as poor diet, physical inactivity, and smoking. Certain conditions are also associated to high blood pressure, elevated cholesterol levels, diabetes, or chronic kidney disease [11]. There is also a possibility that a person may develop CVDs due to a family history of heart disease. It is important to detect cardiovascular disease as early as possible so that management with counselling and medicines can begin [1].

1.2.3 Cardiovascular Diseases Statistics

Cardiovascular diseases (CVDs) are the leading cause of death globally, taking an estimated 17.9 million lives each year according to the World Health Organization (WHO). More than four out of five CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in people under 70 years of age [1]. The Centers for Disease Control and Prevention says that one person dies every 34 seconds in the United States from Cardiovascular disease. About 697,000 people in the US died from heart disease in 2020 which is 1 in every 5 deaths [12]. The most recent WHO data released in 2020 reveals that in Algeria, there were 54,547 deaths caused by coronary heart disease, which accounts for 29.46% of all deaths, and 23,393 deaths caused by stroke, which accounts for 12.63% of all deaths.

1.2.4 Cardiovascular Diseases Symptoms

Cardiovascular disease symptoms can vary depending on the type of the disease and its severity. These symptoms can be chest pain (angina), pressure, heaviness, or discomfort, sometimes described as a 'belt around the chest'. Many people with cardiovascular diseases also experience [13]:

- Shortness of breath.
- Dizziness or fainting.
- Fatigue or exhaustion.
- Pain in the back or left arm.
- A fast heartbeat.
- Vomiting.

Women often have different symptoms like [14]:

- Nausea
- Stomach aches after eating.
- Sweating.
- Feeling weak.

However, it is important to note that some people do not feel any symptoms until they have a heart attack or a stroke [14]. That is what makes cardiovascular diseases challenging. Therefore, Regular health checkups are essential in order to identify them at an early stage.

1.2.5 Types of Cardiovascular Diseases

There are many heart diseases with different causes. Atherosclerosis is often involved in the development of these diseases, where a buildup of a fatty substance called plaque occurs on the inner walls of arteries. This can make it difficult for blood to flow through these tubes that transport blood from the heart to different parts of the body. As a result, symptoms such

as chest pain or heart attacks may occur [9]. There are many different types of cardiovascular diseases including:

- **Coronary Artery Disease (CAD):** It is a condition that results from the buildup of plaque in the coronary arteries due to atherosclerosis as can be seen in Fig.2, causing the arteries to narrow over time. This narrowing can result in partial or complete blockage of blood flow through the arteries [15].

- **Heart Valve Disease:** occurs when one or more of the heart valves fail to open or close properly, disrupting blood flow through the heart. The three types of heart valve disease are regurgitation, stenosis, and atresia. Regurgitation occurs when the valve flaps fail to close properly, allowing blood to leak backward in the heart. Stenosis occurs when the valve flaps become thick or stiff and fuse together, reducing blood flow through the valve. Atresia occurs when the valve is not formed, and a solid sheet of tissue blocks the blood flow between heart chambers [16].

- **Congestive Heart Failure:** happens when the heart muscle does not pump blood efficiently. Blood can accumulate and cause fluid buildup in the lungs, leading to shortness of breath. The heart becomes weaker and stiffer over time due to certain heart conditions like high blood pressure or narrowed arteries in the heart [17].

- **Cardiomyopathy:** a form of heart disease that weakens the heart muscles and makes it difficult to pump blood properly. This can result in heart valve problems or heart failure. The three main types of cardiomyopathies are dilated, hypertrophic, and restrictive [18].

- **Heart Arrhythmia:** it is when the heart beats irregularly, either too fast or too slow. This can be a harmless fluttering or racing sensation, but in some cases, it can cause life-threatening symptoms. It can be congenital or develop over time, and untreated arrhythmia can lead to cardiac arrest and stroke [19].

- **Pericarditis:** it is the swelling and irritation of the Pericardium as depicted in Fig.2. It can cause sharp chest pain when the layers of the pericardium rub against each other. In some cases, pericarditis can be severe and may require medication or surgery. Early diagnosis and treatment can reduce the risk of long-term complications and potential heart failure [20].

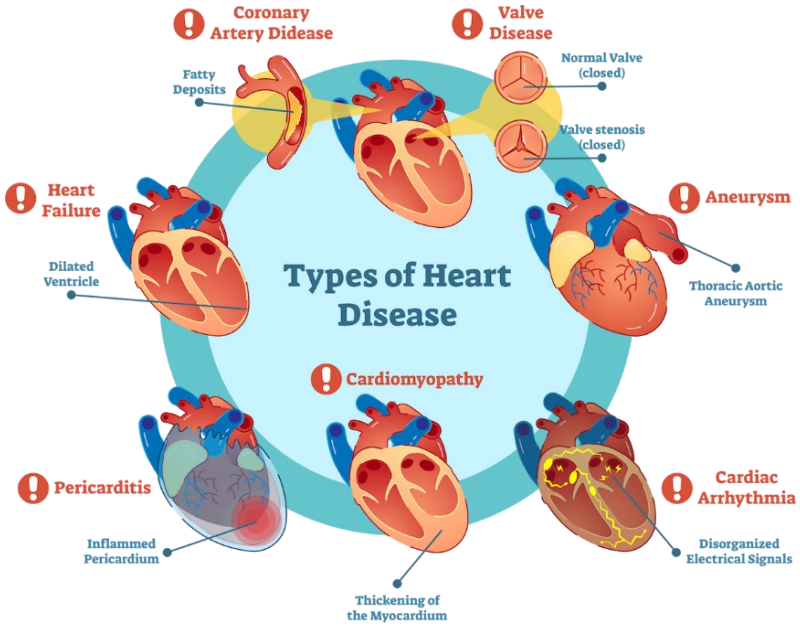


Figure 2: Types of Heart Disease [14].

1.3 Diagnostic Methods for Cardiovascular Diseases

1.3.1 Overview of the current procedures

There are many different tests and diagnostic methods that can be used to diagnose heart diseases, these methods can be classified as invasive and non-invasive techniques.

Non-invasive procedures are those that do not require inserting an instrument through the skin or into a body opening [21]. They include [22]:

- **Physical Examination:** The process of physical examination involves a thorough evaluation of the patient's cardiovascular system, which includes examining and feeling the heart and blood vessels, as well as listening to their sounds.
- **Electrocardiogram (ECG or EKG):** a non-invasive test that records electrical signals in the heart.
- **Echocardiogram:** Medical imaging technique that uses sound waves to create detailed images of the heart in motion.
- **Stress tests:** A diagnostic method that monitors the heart's response to physical activity, which can reveal the presence of heart disease and associated symptoms.
- **Heart CT scan:** A diagnostic imaging test that uses X-rays and computer technology to generate detailed images of the heart and chest.
- **Heart MRI scan:** Cardiac Magnetic Resonance Imaging, a diagnostic imaging technique that uses magnetic fields and radio waves to create detailed images of the heart and its structures.

Invasive procedures are those that invade the body, usually by cutting or puncturing the skin or by inserting instruments into the body [23]. They include:

- **Cardiac Catheterization:** A diagnostic method that involves inserting a tube through a blood vessel and guiding it to the heart to measure pressures and identify blockages in the heart arteries or irregular heartbeats.
- **Coronary Angiogram:** A procedure where a special dye visible by X-ray is injected into the blood vessels of the heart to produce a series of images (angiograms), which can reveal any blockages or abnormalities in the blood vessels.

These procedures are more invasive and carry higher risks of complications than non-invasive diagnostic methods. In some cases, they may be necessary for a definitive diagnosis or to guide treatment decisions. However, non-invasive methods are generally preferred whenever possible due to their lower risk and greater convenience for patients.

1.3.2 The Need for Improved Diagnostic tools

While there are available diagnostic methods for cardiovascular diseases, developing more sensitive and specific diagnostic tools is necessary. Current methods requiring specialized equipment and personnel may not be available in some settings and unable to detect early stages of the disease, limiting the effectiveness of treatments. Hence, improved diagnostic techniques that are more accessible and accurate are needed for early detection and better management of cardiovascular diseases.

1.4 Heart Sounds

1.4.1 Overview

Many pathologic cardiac conditions can be diagnosed by auscultation of the heart sounds [24]. Heart sounds are the noises generated by the beating heart and the resultant flow of blood through it [25]. These sounds reflect the turbulence created when the heart valves close during the cardiac cycle.

1.4.2 Normal Heart Sounds

The normal heart sounds usually consist of two main sounds S1 and S2 as can be seen in Fig.3. They are produced by the closing of atrioventricular valves (AV) and semilunar valves (SL), respectively [25]. The first heart sound S1 is followed by a short silence called Systole while the ventricles contract. Diastole is the second time interval between S2 and S1 while the ventricles relax and fill with blood which is usually longer than the Systole.

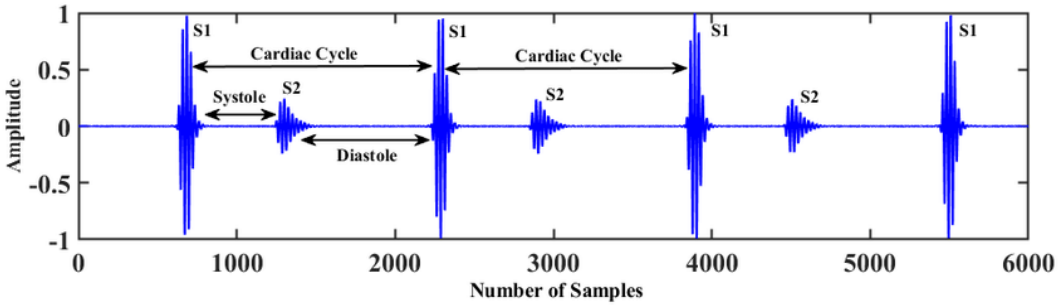


Figure 3: Normal heart sound signal showing S1, S2, systolic and diastolic periods [26].

Examiners utilize a stethoscope to listen to heart sounds, which are auscultated at four different sites on the chest wall that are illustrated in Fig.4. These sites correspond to the locations of blood flow passing through the aortic, pulmonic, tricuspid, and mitral valves, respectively. This makes it possible to differentiate similar defects associated with different valves [27].

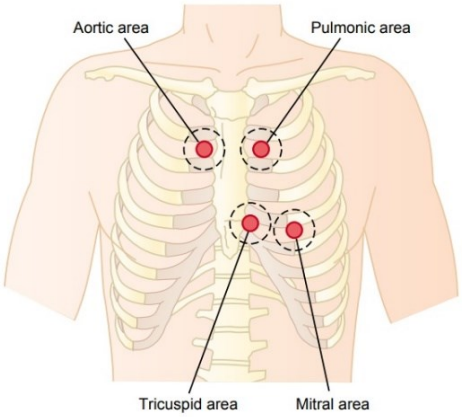


Figure 4: Chest areas from which sound from each valve is best heard [28].

1.4.3 Abnormal Heart Sounds

Heart sounds can have extra sounds which are referred to as Gallop Rhythms. A 3rd heart sound S3 can be normal sometimes but may be pathologic. However, a fourth sound S4 is almost always pathologic [24]. Heart sounds can also include heart murmurs which are unique whooshing sounds produced when blood flows across a heart valve or blood vessel [29]. These murmurs may be physiological or pathological. The causes of abnormal murmurs include stenosis restricting the opening of a heart valve or valvular regurgitation allowing backflow of blood.

Heart murmurs are diagnosed based on the time they occur in the cardiac cycle (Systolic or Diastolic), their changes in intensity (Shape), and the auscultation site where they are best heard (location). We can classify murmurs into Systolic and Diastolic murmurs. For example, systolic murmurs include:

- **Mitral and Tricuspid regurgitation:** The murmurs of both mitral and Tricuspid regurgitation start at S1 and have a consistent intensity throughout the systole period

which can be seen in Fig.5. While Mitral regurgitation is best heard at the mitral region, Tricuspid regurgitation is best heard at the tricuspid area.

- **Aortic and Pulmonic stenosis:** When the aortic or pulmonic valves fail to open properly, they create pressure on the blood through a narrow opening. The blood flow starts out small, rises to a maximum in mid-systole, and then decreases towards the end of systole as illustrated in Fig.5. The opening of the stenotic valve often precedes an ejection click. Aortic stenosis is most audible in the aortic area, while Pulmonic stenosis is best heard in the pulmonic area.

Other systolic murmurs include ventricular septal defect and mitral valve prolapse. Meanwhile, diastolic murmurs consist of aortic regurgitation, pulmonic regurgitation, mitral stenosis, and tricuspid stenosis.

- **Aortic regurgitation:** when the aortic valve does not close properly, the blood flows back to the left ventricle during diastole (the filling phase). This type of murmur is best heard in the left sternal border between the aortic and tricuspid areas and has a triangular shape, with a peak at the beginning of diastole and a rapid decrease thereafter as depicted in Fig.5.

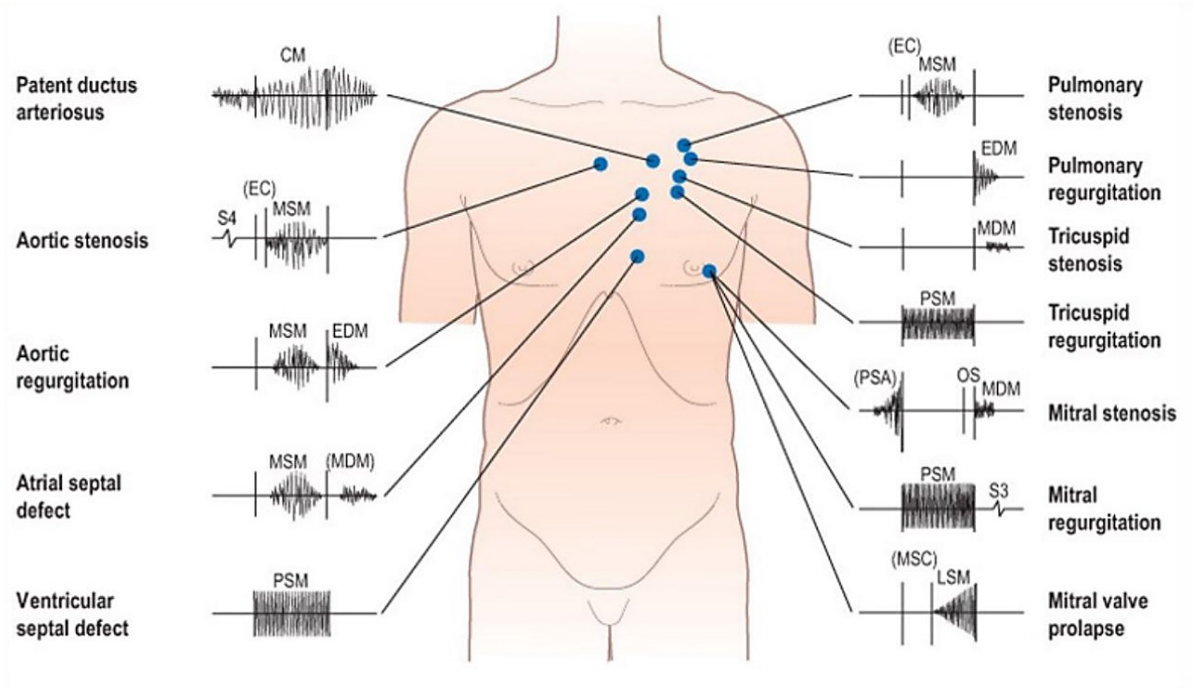


Figure 5: Heart murmurs time, shape, and location [30].

1.4.4 Phonocardiogram (PCG)

A phonocardiogram (PCG) is a precise record of the heart's sounds and their duration over a period of time. It is a valuable tool for evaluating heart health by detecting inaudible murmurs and sounds. PCGs provide quantifiable data on the characteristics of heart sounds, such as intensity, timing, and duration, and are a useful way to monitor disease progression [31]. The acoustical stethoscope is the primary device used to auscultate heart sounds. However, an enhanced alternative to the acoustical stethoscope is the electronic stethoscope, which consists of a microphone, amplifier, and headset [32].

1.5 Deep Learning Theory

1.5.1 Overview

Deep learning is a subset of machine learning that employs artificial neural networks to learn and make decisions based on given data, similar to how the human brain works. These networks consist of multiple layers of interconnected nodes, which enable the algorithm to learn complex patterns and relationships in the data. Deep learning has achieved remarkable success in diverse applications such as image and speech recognition, natural language processing, and predictive analytics. Unlike other machine learning methods, deep learning can automatically learn and extract features from raw data, making it particularly suitable for image and speech recognition tasks.

1.5.2 Artificial Neural Networks

Artificial neural networks (ANNs) are computational systems modeled after the biological neural networks present in animal brains. These networks consist of interconnected artificial neurons that receive and process signals. The connections between neurons (edges) are assigned weights and biases that adjust during the learning process. These weights modify the strength of the signal transmitted between neurons. Typically, neurons are organized into layers, with each layer performing specific transformations on the inputs it receives [33].

By stacking multiple layers of artificial neurons, a deep neural network can be formed. This process allows signals to propagate through the layers, starting with the input layer and passing through intermediate hidden layers before ultimately reaching the output layer. A visual representation of this process is shown in the Fig.6 below:

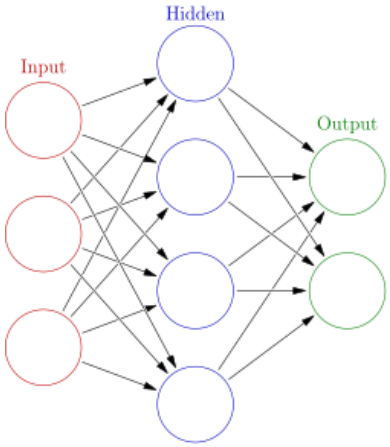


Figure 6: Artificial Neural Network [34]

1.5.3 Convolutional Neural Networks

Convolutional neural networks commonly referred to as CNNs or ConvNets is a class of Artificial Neural Networks that is commonly used for image and video processing tasks [35]. It works by automatically and adaptively learning to assign importance to various aspects and

objects in the input image [36], making them highly effective at tasks such as image classification, object detection, and segmentation.

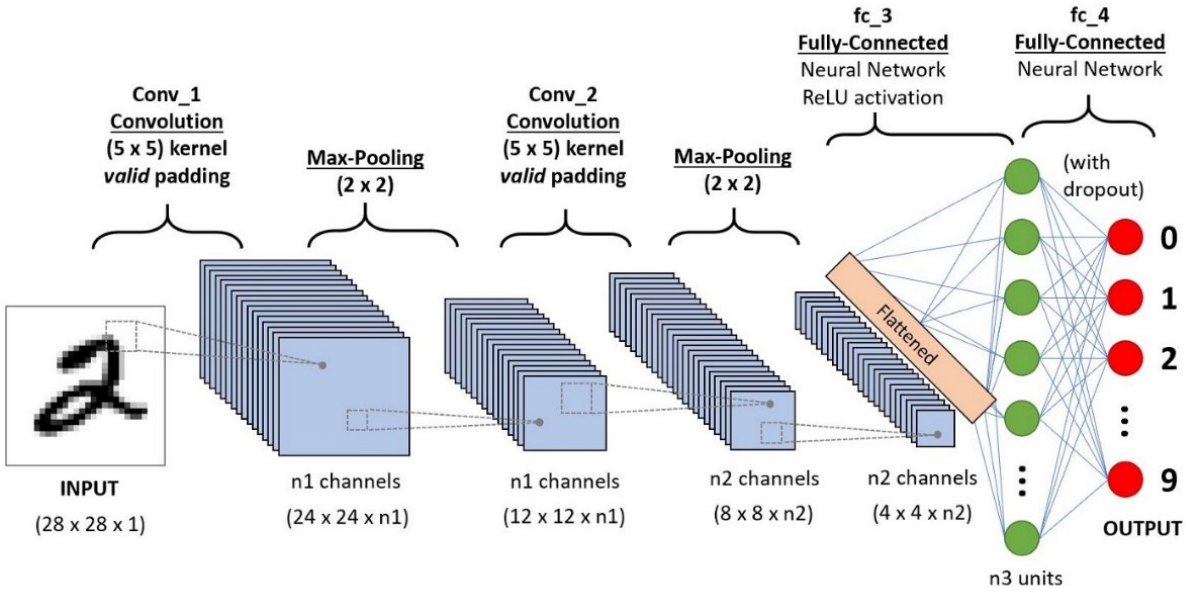


Figure 7: Convolution neural network

In CNNs, a variety of layers are utilized, including Convolutional Layers, Pooling Layers, Non-Linear Layers, Flatten Layers, and Fully Connected Layers as depicted in Fig.7.

Convolutional Layer: The Convolutional Layer is a crucial element of Convolutional Neural Networks (CNNs) that plays a significant role in processing images. This layer operates by applying a set of learnable filters to the input image. These filters, also referred to as kernels, are small matrices that are convolved with the input data to produce corresponding feature maps. The number of filters can be adjusted based on the complexity of the input image to extract more features. During the training phase of the CNN, the weights in each filter are optimized to improve the accuracy of the network.

The Convolutional Layer offers adjustable parameters such as kernel size, stride, and padding. The kernel size determines the size of the matrix used for the convolution and can significantly impact the feature extraction capabilities of the CNN. Stride, on the other hand, defines the step size of the kernel during convolution, while padding adds additional border pixels around the input image to maintain spatial dimensions.

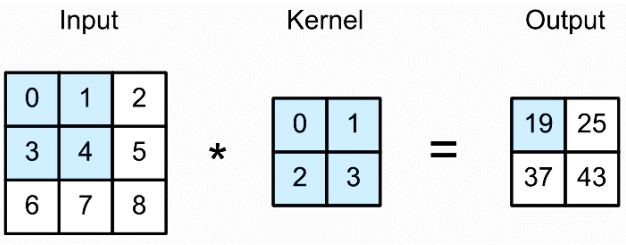


Figure 8: Convolutional layer operation.

In the case of images with multiple channels, the kernel has the same depth as that of the input image. After the matrix multiplication is performed, the output of each channel is summed with the bias to give us a squashed one-depth channel convoluted feature output [36].

Pooling Layer: In Convolutional Neural Networks (CNNs), the pooling layer is a critical component used to reduce the dimensionality of feature maps generated by the convolutional layer. The pooling layer works by aggregating neighboring pixel values within a small region of the feature map. This helps to reduce the size of the feature map while preserving the essential features, thereby reducing computational complexity.

There are several types of pooling layers available, but max pooling and average pooling are two commonly used ones. Max pooling is used to highlight the most significant features in the input data by selecting the maximum value within the pooling window as the output. On the other hand, average pooling computes the average of the values in the pooling window and is often used to reduce the effect of noise in the input.

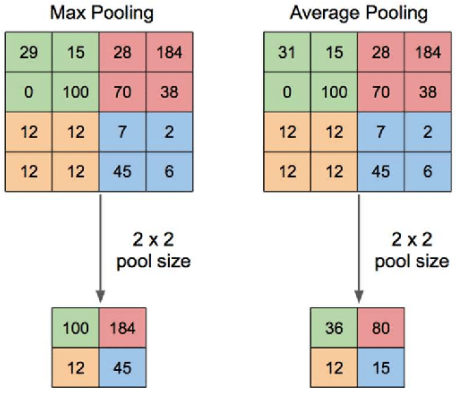


Figure 9: Max-pool and Average-pool layer operation [37].

Non-Linear Layer: A non-linear layer is an important component of neural networks that introduces non-linearity to the output of the layer. This is achieved by applying a non-linear activation function to the layer's input, allowing the network to learn complex non-linear patterns in the input data.

Two commonly used non-linear activation functions are the Rectified Linear Unit (ReLU) and the sigmoid function. ReLU returns the input if it is positive and zero otherwise, which makes it simple and efficient, and it has been shown to be effective in deep learning models. On the other hand, sigmoid maps any input to a value between 0 and 1, making it useful for binary classification problems where the output is interpreted as the probability of the input belonging to a certain class. However, sigmoid can suffer from vanishing gradients, which can make it challenging to train deep networks.

Flatten Layer: The flatten layer is used to convert the output of the previous layers from a multidimensional format into a one-dimensional vector. This allows the output to be processed by a fully connected layer.

Fully Connected Layer: in CNNs a fully connected layer also called a dense layer, is a layer where every neuron in the input is connected to all the neurons in the output.

The fully connected layer generates a vector of fixed size, which is usually determined by the number of classes. It is usually the last layer of the CNN architecture, taking the flattened output of the previous layers to produce the final output. The fully connected layer is necessary for the CNN to learn complex non-linear relationships between the input data and the final output.

Processing large images with traditional neural networks can be computationally expensive due to the high number of neurons and weights needed to handle the image size. For instance, an 8k image of size 7680 by 4320 would require 33,177,600 neurons. However, Convolutional Neural Networks can efficiently process images with a small number of neurons and weights. This is achieved by using small kernel sizes, such as 3 by 3, resulting in only 9 neurons per layer. As a result, CNNs are more suitable for image processing tasks and provide better efficiency compared to traditional neural networks.

1.5.4 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) is another class of Artificial Neural Networks where connections between nodes can create a cycle [38]. They are commonly used to analyze sequential data, such as time series and natural language. RNNs are able to maintain an internal state or memory which allows them to capture information from previous inputs that affects the subsequent inputs. This makes RNNs well-suited for tasks where the meaning of a sequence depends on the context of preceding elements. Another distinguishing characteristic of recurrent networks is that they share parameters across each layer of the network, while feedforward networks have different weights across each node [39].

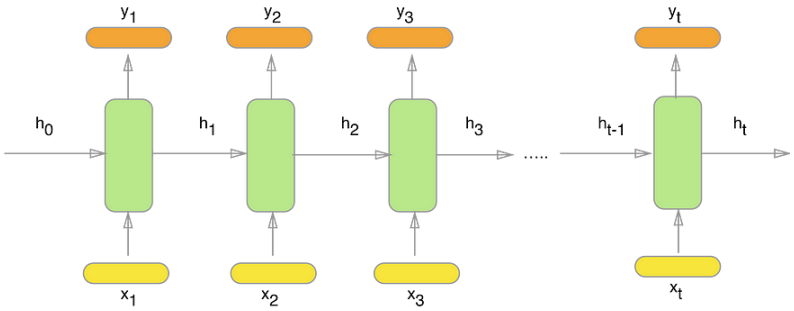


Figure 10: Recurrent neural network

The architecture of Recurrent Neural Networks involves a feedback mechanism, as depicted in Fig.10 such that:

$$h_t = f(W_2.h_{t-1} + W_1.x_t) \quad \text{where} \quad h_0 = 0$$
$$y_t = softmax(W_3.h_t) \qquad (1.5.1)$$

This feedback mechanism utilizes the information from the previous input h_{t-1} to calculate that of the current input h_t which is then passed through a non-linear activation function f (usually tanh or sigmoid). The output of this activation function is then used to calculate the predicted output y_t for the current input using a Softmax function.

Despite the success of training, using the simplest RNN model for predicting outputs can yield disappointing results due to the vanishing gradient and exploding gradient problems[1], which significantly reduce its accuracy [40]. To address these issues, a better alternative is the Long Short-Term Memory (LSTM) Networks.

1.5.5 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) Networks are built on the same principles as RNNs, but with a more complex architecture that is able to address the issues of vanishing and exploding gradients. LSTMs are designed to maintain information over longer periods of time by using a memory cell (cell state) that remembers values over arbitrary time intervals and three gates: a forget gate, an input gate, and an output gate. These gates are responsible for controlling the flow of information into and out of the memory cell, which allows LSTMs to selectively remember or forget information as needed.

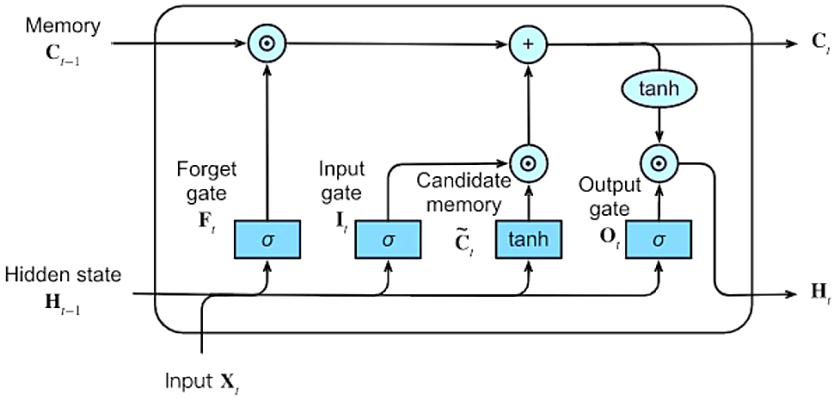


Figure 11: LSTM Architecture

Fig.11 illustrates the architecture of an LSTM network. The input data is processed through the three stages: the forget gate, input gate, and output gate.

Forget gate: The forget gate produces an output between 0 and 1 for each memory cell, indicating how much of the previous memory cell state should be retained based on the previous hidden state and the current input. A value of 1 means to keep all the information, and a value of 0 means to discard it [41]. This gate formula is given by:

$$F_t = \sigma\left(h_{t-1}.U_f + X_t.W_f + b_f\right) \qquad 0 \leq F_t \leq 1 \tag{1.5.2}$$

where U and W are the learnable weights corresponding to the hidden state h_t and the input X_t, respectively.

Input gate: The input gate determines which parts of the new input should be used to update the memory cell state. It uses the current input and the previous hidden state to

[1] Gradients becoming extremely small or large during backpropagation

produce two outputs, one for which parts of the memory cell state should be updated and another for the potential new values known as the candidate memory. They are given by:

$$I_t = \sigma(h_{t-1}.U_i + X_t.W_i + b_i) \qquad 0 \leq I_t \leq 1$$
$$\tilde{C}_t = tanh(h_{t-1}.U_c + X_t.W_c + b_c) \qquad -1 \leq \tilde{C}_t \leq 1 \tag{1.5.3}$$

The new memory cell state can now be computed as follows:

$$C_t = C_{t-1}.F_t + I_t.\tilde{C}_t \tag{1.5.4}$$

Output gate: The output gate determines how much of the memory cell state should be exposed to the output. It uses the current input and the previous hidden state as well as the new memory cell state to produce an output vector which is also the hidden state of the current time step *t*. It is given by:

$$O_t = \sigma(h_{t-1}.U_o + X_t.W_o + b_o)$$
$$h_t = \tanh(C_t)\,.O_t \tag{1.5.5}$$

In an LSTM layer, the size of the hidden state matrix h_t is determined by the hidden size, which represents the number of features within an LSTM cell.

The weight matrices U and W have specific shapes, with U having a matrix of dimensions *hidden_size x hidden_size,* and W having a matrix of dimensions *hidden_size x input_variables*, where *input_variables* is the shape of the input at the current time step [42].

The hidden size is a critical hyperparameter that significantly impacts the LSTM model's capacity to learn complex patterns in the data. Selecting an appropriate hidden size is essential to ensure optimal performance of the model, considering the dataset's size, complexity, and available computational resources.

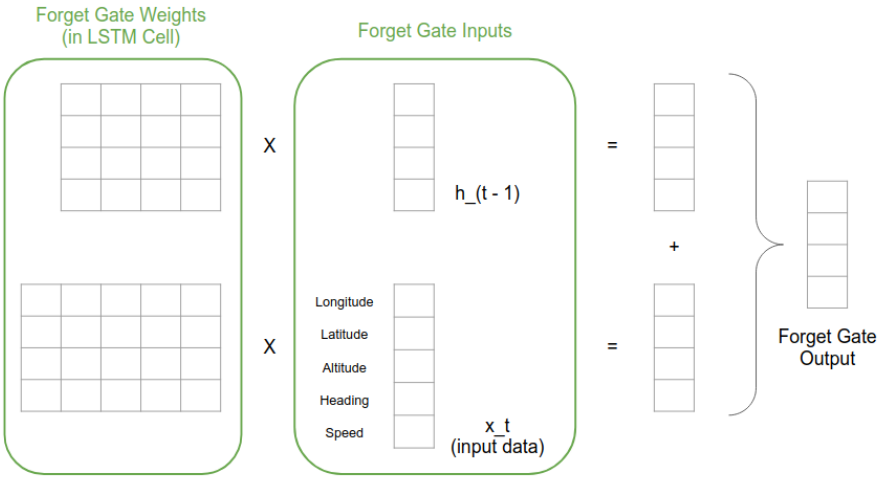


Figure 12: Forget Gate Operations and Dimensions example [42].

The depicted example in Fig.12 illustrates a forget gate operation on time series data with 5 variables (Longitude, Latitude, Altitude, Heading, Speed) and a hidden size of 4. This results in the weights having shapes of 4×4 and 4×5 for U and W, respectively, And an output of 4×1.

1.5.6 Deep Neural Network Learning Process

Training a deep neural network involves determining the optimal weights for each neuron to produce the most accurate results and minimize the error between the predicted output and the actual output based on the provided training data as demonstrated in Fig.13. The process comprises several interrelated steps, including:

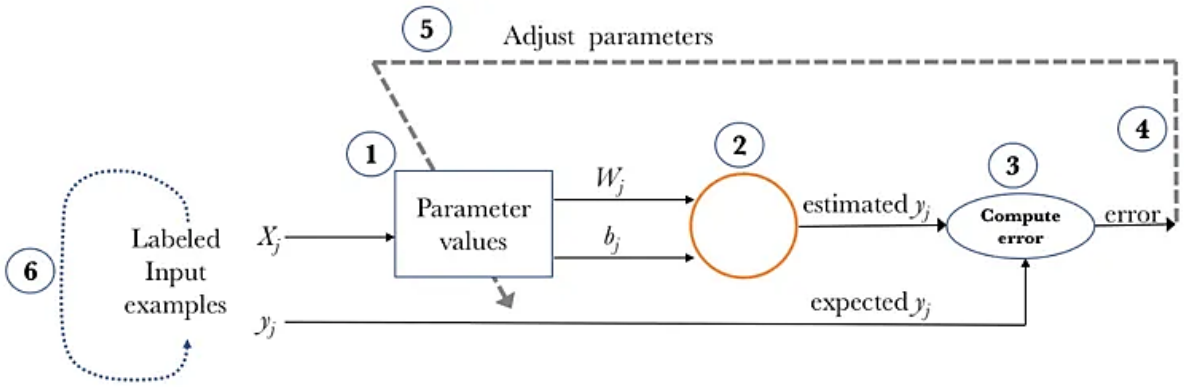


Figure 13: Neural Networks Learning Algorithm [43].

1. Data Preparation

In most cases, the input data must be suitably preprocessed to fit the desired sizes and shapes before it can be fed into the neural network. This preprocessing step may include operations such as tensor[2] reshaping, trimming, or padding, and ensuring that the data resides on the same computing device as the model.

The first step of preparing the data involves collecting and organizing the training data that will be used to train the neural network. Typically, this data is partitioned into distinct subsets for training, validation, and testing purposes. The training set is used to optimize the model's parameters, while the validation set is used to monitor the model's performance and prevent overfitting[3]. Finally, the testing set is used to evaluate the model's ability to generalize to new, unseen data.

Once the data is split into training, validation, and testing sets, a DataLoader can be used to load the data into the memory. The DataLoader samples the dataset into minibatches based on the specified batch size and also provides the option to shuffle the data [44], which can help mitigate the effects of data order on the model's training. Additionally, the DataLoader can be used to resample the data to address any class imbalance issues.

Training the model using minibatches typically leads to faster convergence and better results compared to training on the entire dataset at once. This is because training on a minibatch allows the model to update its parameters more frequently, which can lead to quicker convergence. Moreover, using minibatches enables the model to handle larger

[2] A tensor is a multidimensional array or matrix.
[3] The model performs well on the training data but poorly on unseen data.

datasets that may not fit into memory, as only a subset of the data is processed at each iteration.

2. Model Definition and Hyperparameters

Defining the neural network architecture involves determining the type and number of layers, the number of neurons and the activation functions applied in each layer. In addition, to accelerate the training process, deep learning models are often prepared at this level to be processed on GPUs, for faster performance.

Once the architecture is set, the parameters of the neural network are initialized randomly. In addition to defining the architecture, training hyperparameters such as the number of epochs, loss functions, optimizers, learning rates, and schedulers need to be specified. These hyperparameters can have a significant impact on the final performance of the model and must be carefully selected based on the specific task at hand.

3. Forward propagation

During the training process, the neural network receives the training data as input and computes its output based on the current parameter values. This involves passing the input through each layer of the network, where each neuron applies its transformation to the information it receives from the neurons in the previous layer before passing it on to the neurons in the next layer. This process continues until the output of the last layer, which represents the final prediction of the network.

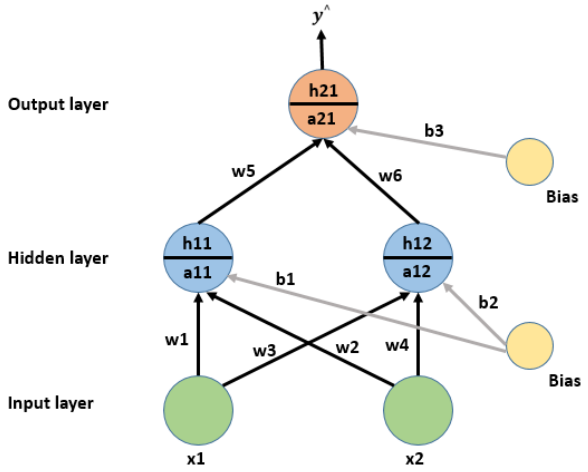


Figure 14: Forward Propagation Example [45].

Neurons in a neural network process information in two steps: Preactivation and activation. In the Preactivation step, the neuron computes the weighted sum of its input. In the activation step, the result of the Preactivation is passed through an activation function [45]. Fig.14 shows an example of a neural network with 2 inputs, 1 hidden layer of 2 neurons, and 1 output. During forward propagation, preactivation and activation occur at each node of the hidden and output layers. The preactivation output of the first hidden neuron is denoted as ***a11***, and its corresponding activation output is denoted as ***h11***, such that:

$$a_{11} = w_1 * x_1 + w_2 * x_2 + b_1$$
$$h_{11} = \sigma(a_{11}) \qquad (1.5.6)$$

Similarly, they are denoted for the second hidden neuron as:

$$a_{12} = w_3 * x_1 + w_4 * x_2 + b_2$$
$$h_{12} = \sigma(a_{12}) \quad (1.5.7)$$

Finally, the final result which is the output neuron activation can be calculated by:

$$a_{21} = w_5 * h_{11} + w_6 * h_{12} + b_3$$
$$h_{21} = \sigma(a_{21}) \quad (1.5.8)$$

4. Loss Computation

The loss computation step involves evaluating how well the deep learning algorithm models the dataset by calculating the difference (error) between the predicted and actual output using a loss function. The lower is the loss, the better will be the model performance [46]. There are various loss functions designed for specific tasks, with Binary Cross Entropy and Categorical Cross Entropy being the most commonly used for classification tasks.

Binary Cross Entropy measures the distance between predicted probabilities and actual labels in binary classification problems. Its formula is given by:

$$Cost = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)], \quad (1.5.9)$$

where $\hat{y}_i$ is the neural network predicted value and y_i is the actual value. N is the number of samples.

Categorical Cross Entropy is similar, but it is used for multiclass classification tasks. It is given by:

$$Cost = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k}[y_{ij} \log(\hat{y}_{ij})], \quad (1.5.10)$$

where k is the number of classes.

5. Backward Propagation

Back-propagation is a widely used algorithm in deep learning that calculates the gradient of the loss function with respect to the weights for a single input-output example. It is based on the chain rule in calculus and computes the gradient one layer at a time while iterating backwards [47]. This allows the information from the cost to flow backwards through the network, enabling fine-tuning of the weights [48].

Assume we have the following simplified neural network:

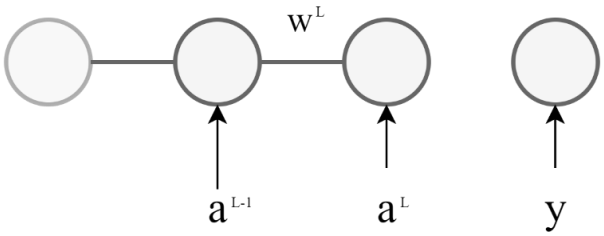


Figure 15: Simplified neural network.

Here y is the actual output and a^L is the predicted output. We have:

$$z^L = w^L * a^{L-1} + b^L, \tag{1.5.11}$$

where w^L is the weight between a^{L-1} and a^L, and b^L is the bias.

$$a^L = \sigma(z^L) = \frac{1}{1 - e^{-z}} \tag{1.5.12}$$

As we can see in the Equation 1.5.10 the cost for one sample is related to a^L or $\hat{y}_{ij}$ which is affected by w^L and b^L. Our first goal is to understand how sensitive the cost function is to small changes in our weight w^L. meaning, what is the derivative of the cost C_0 with respect to w^L [49].

Using the chain rule, we find that [50]:

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial C_0}{\partial a^L} \frac{\partial a^L}{\partial z^L} \frac{\partial z^L}{\partial w^L} \tag{1.5.13}$$

However, all these calculations are for one training sample only since the actual cost function is the average of all the costs in a minibatch, the actual gradient is the average of all the derivatives for each sample:

$$\frac{\partial C}{\partial w^L} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial C_i}{\partial w^L} \tag{1.5.14}$$

Also, the gradient for the bias term is similar and can be given by:

$$\frac{\partial C_0}{\partial b^L} = \frac{\partial C_0}{\partial a^L} \frac{\partial a^L}{\partial z^L} \tag{1.5.15}$$

Since

$$\frac{\partial z^L}{\partial b^L} = 1 \tag{1.5.16}$$

Now that we know how to calculate the gradient for the last layer, we can propagate backwards by calculating the derivative of the cost with respect to the activation of the

previous layer a^{L-1} which allows us to find the derivates of its weights and biases. And we can repeat the same process until we reach the input layer.

$$\frac{\partial C_0}{\partial a^{L-1}} = \frac{\partial C_0}{\partial a^L} \frac{\partial a^L}{\partial z^L} \frac{\partial z^L}{\partial a^{L-1}} \tag{1.5.17}$$

The given example is oversimplified since it has one neuron per layer, in the case of multiple neurons, we add subscripts to each neuron as follows:

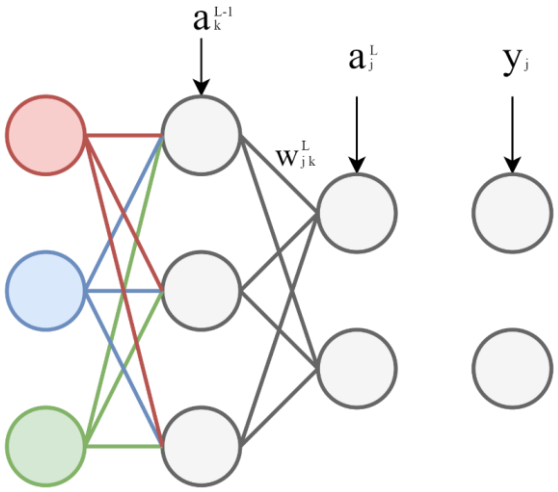


Figure 16: Simple neural network with multiple neurons per layer.

We index the neurons in the L-1'th layer as *k* and those in the L'th layer as *j*, now we get:

$$\begin{gathered} z_j^L = w_{j1}^L * a_1^{L-1} + w_{j2}^L * a_2^{L-1} + w_{j3}^L * a_3^{L-1} + b_j^L \\ a_j^L = \sigma(z_j^L) \\ \frac{\partial C_0}{\partial w_{jk}^L} = \frac{\partial C_0}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{jk}^L}, \end{gathered} \tag{1.5.18}$$

where C_0 is the Cost for one training sample, w_{jk}^L is the weight between the *j*'th and *k*'th neurons in the layer L and L -1 respectively, and a_j^L is the activation of the *j*'th neuron.

One thing that changes is the derivative of the cost C_0 with respect to the activation of a neuron in the previous layer a_k^{L-1}. As depicted in Fig.16 the neurons in the layer L-1 can affect the final result through 2 different paths (which is the equivalent to the number of neurons in the layer L), so we have to sum them up as follows:

$$\frac{\partial C_0}{\partial a_k^{L-1}} = \sum_{j=0}^{n(L)} \frac{\partial C_0}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial a_k^{L-1}} \tag{1.5.19}$$

where n(L) is the number of neurons in layer L.

6. Optimization

In deep learning, once the gradient has been computed, the next step is to update the parameters of the neural network accordingly. This optimization process involves using optimization algorithms such as [51]:

- **Gradient Descent:** It is the most basic optimization algorithm that updates the parameters using the gradient of the cost and a learning rate α. It is given by:

$$W_{j,t} = W_{j,t-1} - \alpha \frac{dC}{dW_{j,t}} \quad (1.5.20)$$

The size of the steps taken towards the local minimum in Gradient Descent is determined by the learning rate α. A smaller learning rate leads to slower convergence but more accurate results, while a larger learning rate may cause overshooting as seen in Fig.17 below:

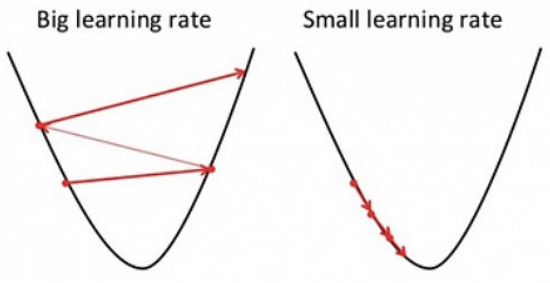


Figure 17: the difference between a big and a small learning rate.

- **Stochastic Gradient Descent:** Gradient Descent updates the parameters only after calculating the gradient on the entire dataset, which can be slow for large datasets. In contrast, Stochastic Gradient Descent updates the parameters after each training example, resulting in a faster but noisy path, which also may not converge to a minimum.
- **Mini-batch Gradient Descent:** To achieve the benefits of both GD and SGD, we use Mini-batch Gradient Descent (MGD), which updates the parameters after each mini-batch instead of after each individual sample. This reduces time complexity while still resulting in some noise.
- **SGD with Momentum:** it is a technique that reduces the convergence time of Stochastic Gradient Descent. It accomplishes this by denoising the gradient using exponential weighting averaging, which gives more weight to recent updates compared to previous ones. It is given by:

$$\begin{aligned} V_{j,t} &= \gamma * V_{j,t-1} + \alpha \frac{dC}{dW_j} \\ W_{j,t} &= W_{j,t-1} - V_{j,t} \end{aligned} \quad (1.5.21)$$

Such that γ is the momentum term and is usually set to 0.9 or similar values.

- **Adam:** Adaptive Moment Estimation (ADAM) is a popular optimization algorithm that combines ideas from both SGD with Momentum and RMSprop (Another optimization algorithm). It employs exponential moving averages to calculate the first and second

moment of the gradient, which allows it to adaptively adjust the learning rate for each weight in the neural network. Its rule is given by [52]:

$$\begin{aligned} m_{j,t} &= \beta_1 m_{j,t-1} + (1-\beta_1) g_{j,t} \\ v_{j,t} &= \beta_2 v_{t-1} + (1-\beta_2) g_{j,t}^2 \\ \hat{m}_{j,t} &= \frac{m_{j,t}}{1-\beta_1^t} \\ \hat{v}_{j,t} &= \frac{v_{j,t}}{1-\beta_2^t} \\ W_{j,t} &= W_{j,t-1} - \alpha \frac{\hat{m}_{j,t}}{\sqrt{\hat{v}_{j,t}} + \varepsilon} \end{aligned} \qquad (1.5.22)$$

Where β_1 and β_2 are hyperparameters that control the decay rates they are mostly set by default to 0.9 and 0.999 respectively, and g is the gradient. m and v are the moving averages however they are biased towards zero at the beginning of the training so in order to correct them we compute the bias-corrected estimates $\hat{m}$ and $\hat{v}$.

7. Validation

During the neural network learning process, the validation step involves assessing the network's performance on a separate validation set. This evaluation includes making predictions on the validation set and computing relevant metrics to monitor the training progress and prevent overfitting. To make predictions on the validation set, the network undergoes a forward pass, and the validation loss and other metrics are computed based on the predicted outputs and the actual values.

8. Testing

After the completion of the training process, the neural network undergoes the evaluation phase, also known as testing. This step is similar to validation, but instead of using the validation set, the testing set is used to assess the trained network's generalization capability and its performance on unseen and independent data.

The main difference between the validation and testing steps is that validation evaluates the model while tuning hyperparameters which can be biased, as the model configuration is influenced by the validation dataset, while testing provides an unbiased evaluation of the final model.

1.5.7 Some Recurrent Problems in Deep Learning

Deep Neural Networks may face several issues during the training process that need to be addressed to achieve optimal results. These issues include:

- **Vanishing Gradient:** When backpropagating the error through many layers, the gradients may become very small, causing the weights to update very slowly or not at all. This can make the training process slow or even stop it entirely [53].
- **Exploding Gradient:** When backpropagating the error through many layers, the gradients may become very large, causing the weights to update too much, and leading to numerical instability [53].

- **Overfitting:** Overfitting occurs when the model performs well on the training data but poorly on new data. This happens when the model has too many parameters, and it starts to memorize the training data instead of learning the underlying patterns [54].
- **Underfitting:** Underfitting occurs when the model is too simple to capture the complexity of the data. This happens when the model has too few parameters or when it is not trained for long enough [54].
- **Dataset Bias:** The dataset used for training may not be representative of the actual distribution of data, leading to a biased model [54].

Adjusting hyperparameters and model complexity can alleviate these problems. For example, adding more parameters and layers can help to prevent underfitting, but it can also lead to overfitting if the model is too complex. Training the model for too long would increase the risk of overfitting, while training for too few epochs might result in underfitting. A smaller learning rate is able to help prevent exploding gradients, but it may also slow down training. In addition to adjusting these factors, several popular techniques can be used to tackle these problems, including:

- **L1 and L2 Regularization:** Also known as weight decay, these techniques add a penalty term to the loss function to limit the magnitude of the weights. L1 regularization promotes sparse weight values while L2 regularization promotes small but non-zero weights [55].
- **Dropout:** it is a layer that randomly drops out a fraction of the neurons in each layer during training [56], forcing the remaining neurons to learn more robust features to prevent overfitting.
- **Data augmentation:** Data augmentation techniques create additional training data by applying random transformations such as rotations, translations, and scaling to the existing data. Sequential data can be augmented through time shifts.
- **Early Stopping:** Early stopping stops the training process when the validation accuracy stops improving, preventing overfitting to the training data.
- **Batch Normalization:** it is a technique that normalizes the inputs of each layer by adjusting the mean and standard deviation of the batch to 0 and 1, respectively. It results in faster and more stable training.
- **Scheduler:** it helps to improve the performance of deep learning models by adjusting the learning rate during training to ensure that the model converges to the optimal solution more effectively.

1.6 Audio Classification Steps

1.6.1 Overview

Audio classification is the process of automatically assigning audio data into predetermined categories. This can be a difficult task due to the complexity and high dimensionality of audio data. In order to build an accurate deep learning model for audio classification, several key steps must be taken. These steps involve identifying and sourcing high-quality data, performing thorough data analysis, as well as preprocessing and feature extraction, to ensure the data is suitable for the model's input. The detail of each step of the audio classification is disused in the following subsections.

1.6.2 Preprocessing

Although deep learning algorithms work well with raw data, it is usually more practical to apply some pre-processing steps to convert it into a format that the model can handle. In the case of audio inputs, these pre-processing steps can be performed dynamically at runtime while reading and loading the audio files. The following pre-processing steps can be applied to audio inputs [57]:

Resampling: changing the sample rate of the audio files to a standard one.

Rechanneling: Converting the audio to stereo or mono, which can reduce dimensionality and improve performance.

Resizing: for consistency, all the audio files are resized to the same fixed time length by trimming or padding.

Time-Shifting: it is a data augmentation technique that shifts the time of the audio data.

Filtering: Filters, such as bandpass and noise reduction filters, are commonly applied to audio data to remove unwanted noise and artifacts.

1.6.3 Segmentation

Audio segmentation involves dividing an audio signal into smaller subgroups or segments. For instance, in phonocardiogram (PCG) signals, we can segment the input into individual heart cycles or even into specific cycle states (such as S1, S2, systole, and diastole) [4] [5]. We can also use windowing techniques to split the signals into smaller segments of fixed lengths.

1.6.4 Feature Extraction

Feature extraction is a crucial step in preparing input data for deep learning models, as it involves identifying and extracting key features that can help the model better understand the data. In the context of audio classification, there are several popular techniques for feature extraction [4] [5]., including:

Time domain features: These features are based on properties of the waveform in the time domain, such as the amplitude, duration, and shape of the signal. These features can include time intervals between events, amplitude measurements, and skewness of the amplitude distribution.

Frequency domain features: These features are based on properties of the signal in the frequency domain, such as the power spectrum of the signal.

Time-Frequency domain features: These features are based on properties of the signal in both the time and frequency domains and can capture information about both the temporal and spectral characteristics of the signal. Common time-frequency domain features include mel-frequency cepstral coefficients (MFCCs), spectrograms, and wavelet transforms.

By extracting these key features from the input audio data, we can create a more compact and meaningful representation of the data that can be used to train and improve the accuracy of the deep learning model.

In our research, we will be using two different Time-Frequency domain features namely MFCCs, and Spectrograms. These features are widely used in related works [3] [58] [59] [60] and have been useful in audio applications.

1. Mel Spectrogram

Spectrograms provide a visual representation of audio signals that incorporates both time and frequency domain features. They are constructed by plotting the spectrum of the sound at different time intervals. This creates a three-dimensional graph where the X-axis represents time, the Y-axis represents frequency, and the Z-axis represents the amplitudes of the signal [61]. For a better visualization of the spectrogram, the amplitudes are represented using colors. An Example of a Mel Spectrogram of a heart sound is shown in Fig.18.

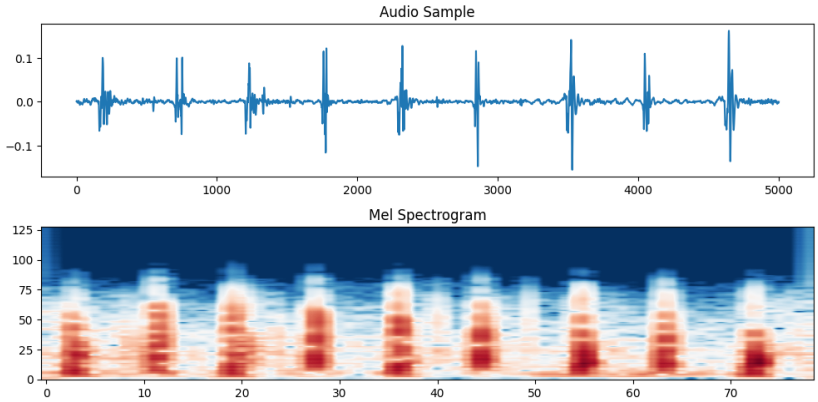


Figure 18: An audio sample of a heart sound and its equivalent Mel Spectrogram.

Audio spectrograms are generated by decomposing the audio signal into its fundamental frequencies using Fourier transforms [61]. While the Fast Fourier Transform (FFT) algorithm is commonly employed, it provides a spectrum of the entire signal without any temporal information or insight into how frequencies change over time [62]. To address this, the Short Time Fourier Transform (STFT) algorithm is utilized. The STFT algorithm partitions the audio into smaller segments using a sliding time window, computes the FFT for each segment, and subsequently merges the outcomes to construct a spectrogram.

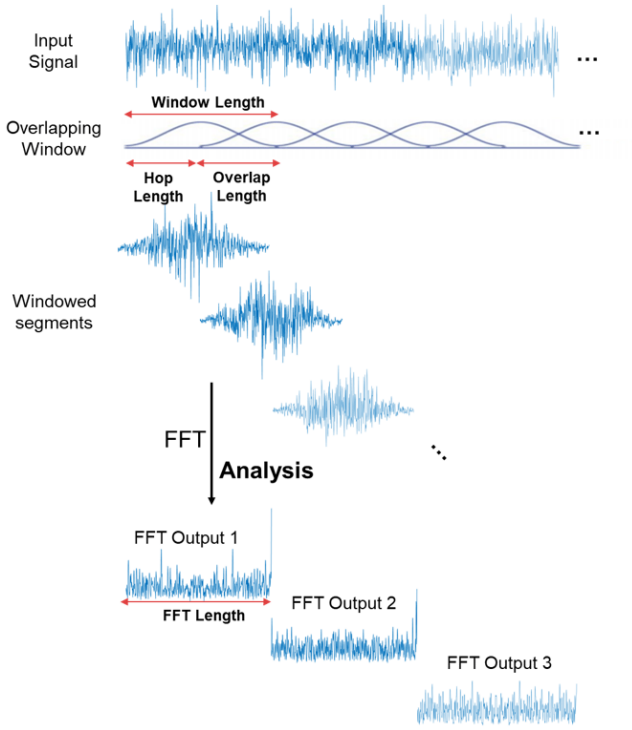


Figure 19: Visualization of the STFT process [63].

In Fig.19, we can observe the process of the STFT, which involves several important parameters:

Window length: it refers to the length of the windows used when computing the STFT. It determines the time resolution of the resulting spectrogram.

FFT size: This parameter defines the number of points used in the FFT when computing the STFT. It determines the frequency resolution of the resulting spectrogram.

Hop length: The hop length parameter determines the number of samples the window is shifted for each subsequent calculation.

Regular spectrograms often lack vibrant colors due to the linear representation of frequencies. Since human perception of frequencies follows a logarithmic scale, distinguishing differences in large frequencies becomes challenging compared to smaller frequencies [61]. To address this limitation, the frequencies are converted to the mel scale using the following equation:

$$Mel(f) = 2595\log\left(1 + \frac{\mathrm{f}}{700}\right) \qquad (1.6.1)$$

Similarly, amplitudes of a sound are also perceived as its loudness which is heard logarithmically rather than linearly. So, the decibel scale is used for the amplitudes instead.

To convert the spectrogram to the mel scale, mel filterbanks are utilized. A Mel filterbank is a set of triangular filters each corresponding to a specific frequency band on the mel scale [64]. The output of each filter is the sum of its weighted spectral components, which represent the energy in that frequency band.

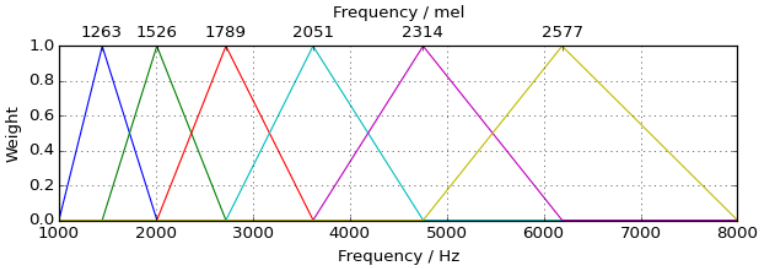


Figure 20: Mel filterbank.

2. Mel Frequency Cepstrum Coefficients:

Mel Frequency Cepstrum Coefficients (MFCCs) provide a concise representation of the spectrum by capturing information about rate changes in different spectrum bands [65]. The term "Cepstrum" refers to the spectrum of a spectrum, which is in the time domain.

The extraction of MFCCs involves the computation of the log mel spectrogram, as shown earlier. However, the coefficients in this spectrogram are highly correlated [65]. To address this, a Discrete Cosine Transform (DCT) is applied to decorrelate the coefficients, resulting in the computation of Mel Frequency Cepstrum Coefficients.

DCT is a Fourier-related transform that shares similarities with the Discrete Fourier Transform (DFT). However, DCT focuses on real-valued signals and employs cosine functions only [66].

In summary, MFCCs can be viewed as the DCT of the log mel spectrogram, serving as an effective method to capture essential characteristics of the audio signal while reducing correlation among the coefficients. An example of an MFCC representation of a heart sound is given in Fig.21 below:

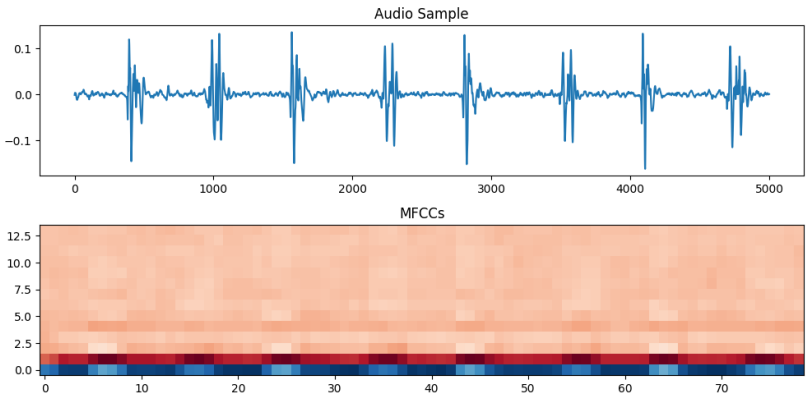


Figure 21: An audio sample of a heart sound and its equivalent MFCCs.

1.7 Summary

In this chapter, we covered various aspects related to cardiovascular diseases and the diagnosis of heart conditions. We discussed the structure and functioning of the heart, along with an overview of different cardiovascular diseases and their impact on human health. We explored the current diagnostic methods and highlighted the need for improved automated diagnosis techniques. Then, we delved into heart sounds and phonocardiogram (PCG) signals, emphasizing their relevance in diagnosing heart diseases.

Furthermore, we introduced the deep learning theory, including artificial neural networks (ANN), convolutional neural networks (CNN), and long short-term memory networks (LSTM). We then discussed the training process and limitations of deep learning models.

Finally, we outlined the steps involved in PCG classification and introduced two feature extraction techniques: Mel Frequency Cepstrum Coefficients (MFCCs) and Mel spectrograms. Overall, this chapter provided a comprehensive foundation for understanding the context, theoretical background, and key concepts related to the subsequent chapters of the report.

Chapter 2: Related Works

2.1 Introduction

This chapter provides an overview of existing research on Phonocardiography (PCG) classification techniques. PCG signals contain important information about the functioning of the heart and are often used in the diagnosis of various cardiovascular diseases. However, analyzing PCG signals can be challenging due to their complex nature and variability across patients. As a result, researchers have developed various classification techniques to accurately identify different heart sounds and diagnose related conditions.

In this chapter, we will review a few research papers that propose different approaches for PCG classification. Each paper will be summarized, and its methodology and findings will be discussed. We will also compare and analyze the different techniques and identify their respective strengths and limitations.

2.2 Existing PCG Classification Techniques

2.2.1 Ensemble of Feature-based and Deep learning-based Classifiers for Detection of Abnormal Heart Sounds

1. Overview:

The paper [58] introduces an algorithm to classify normal/abnormal heart sounds in the 2016 PhysioNet/CinC Challenge. The algorithm involves the extraction of 124 time-frequency features from PCGs, which are then fed into a variant of the AdaBoost Classifier. In addition, a CNN classifier is trained using PCG cardiac cycles that have been decomposed into four frequency bands. The final decision is made based on the output of both classifiers, resulting in an ensemble approach.

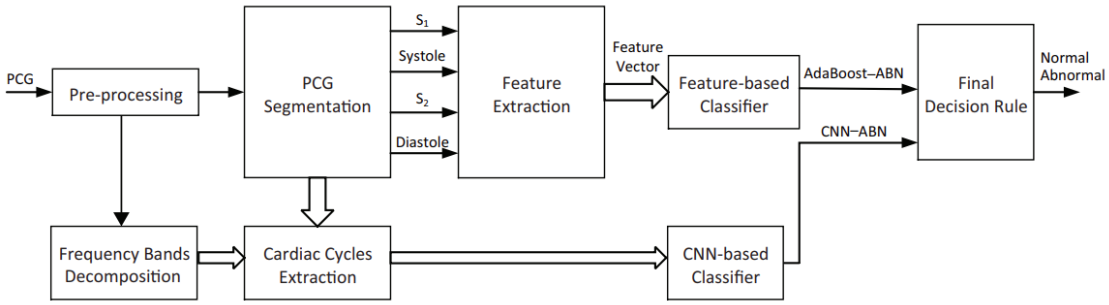


Figure 22: Block diagram of the approach of [58] for classification of normal/abnormal heart sounds.

2. Pre-processing:

To prepare the PCGs for analysis, they were resampled to 1000 Hz and bandpass filtered between 25 Hz and 400 Hz. Next, any spikes were removed from the preprocessed PCGs. The PCGs were then segmented into four heart sound states (S1, Systole, S2, and Diastole) using the *Springer segmentation method*.

3. Feature Extraction:

Different features were extracted from the PCG signals including time domain features, frequency domain features, and time-frequency domain features. For the time domain features, 36 were extracted such that the mean and standard deviation of each parameter was used:

PCG Intervals: RR, S1, S2, Systolic, Diastolic, Ratio of Systolic to RR during each heartbeat. Ratio of Diastolic to RR during each heartbeat, Ratio of Systolic to Diastolic during each heartbeat.

PCG Amplitudes: mean absolute amplitude during Systole to S1 in each heartbeat, mean absolute amplitude during Diastole to S2 in each heartbeat, Skewness of amplitude during (S1, S2, Systole, Diastole) in each heartbeat. Kurtosis of amplitude in (S1, Systole, S2, and Diastole) in each heartbeat

The authors extracted frequency domain features by computing the power spectrum of each heart sound state (S1, Systole, S2, and Diastole) using a hamming window and DTFT. They calculated the median power of 9 frequency bands ranging from 25 Hz to 400 Hz that corresponded to the different states. Then, they took the mean of the median power of the 9 frequency bands across all cycles.

As time-frequency features, the authors utilized 13 Mel-frequency cepstral coefficients (MFCCs) from each cardiac cycle and state, as well as the average of the MFCCs across various cardiac cycles.

4. Classification:

The authors used a boosted classifier, denoted by H(x), which is defined as:

$$H(x) = b + \sum_{t} \alpha_t h(x; \theta_t) \qquad (2.2.1)$$

In this equation, b is a constant bias, $h(x; \theta_t)$ is a base classifier with a vector of parameters θ_t, and α_t is the weight assigned to each base classifier. The function $H(x)$ produces a classification output (+1 or -1) based on the input x. Each base classifier is a simple decision stump over one of the features mentioned earlier, and the modified version of AdaBoost (AdaBoost-abstain) allows each base classifier to abstain from voting by outputting 0. The final decision is made by taking the sign of $H(x)$, which represents a weighted majority vote.

For the CNN approach, each PCG was decomposed into 4 frequency bands from 25 Hz to 400 Hz and segmented into S1, S2, Systole and Diastole then input to the CNN as shown in Fig.23. The input tensor for the CNN corresponds to a specific frequency band of the cardiac cycle, with a length of 2500 samples (1000 Hz * 2.5s). The convolution layers involve a convolution operation, a non-linear transformation, and a max-pooling operation. The first convolution layer consists of 8 filters of length 5, followed by a ReLU and a max-pooling of 2. The second convolution layer consists of 4 filters of length 5, followed by a ReLU and a max-pooling of 2.

The output of the four CNNs is flattened into a single long continuous linear vector and input to a multilayer perceptron (MLP) with an input layer of 4 neurons, a hidden layer with

20 neurons, and an output layer with 1 node. ReLU is used for the hidden layer and Sigmoid for the output layer. Dropout of 25% is applied after the second convolution layer, and dropout of 50% and L2 regularization are applied at the hidden layer of the MLP. Adam optimizer is used for stochastic optimization. Finally, a decision rule is used to combine the two classifiers such that if one of them outputs abnormal then the final decision is abnormal.

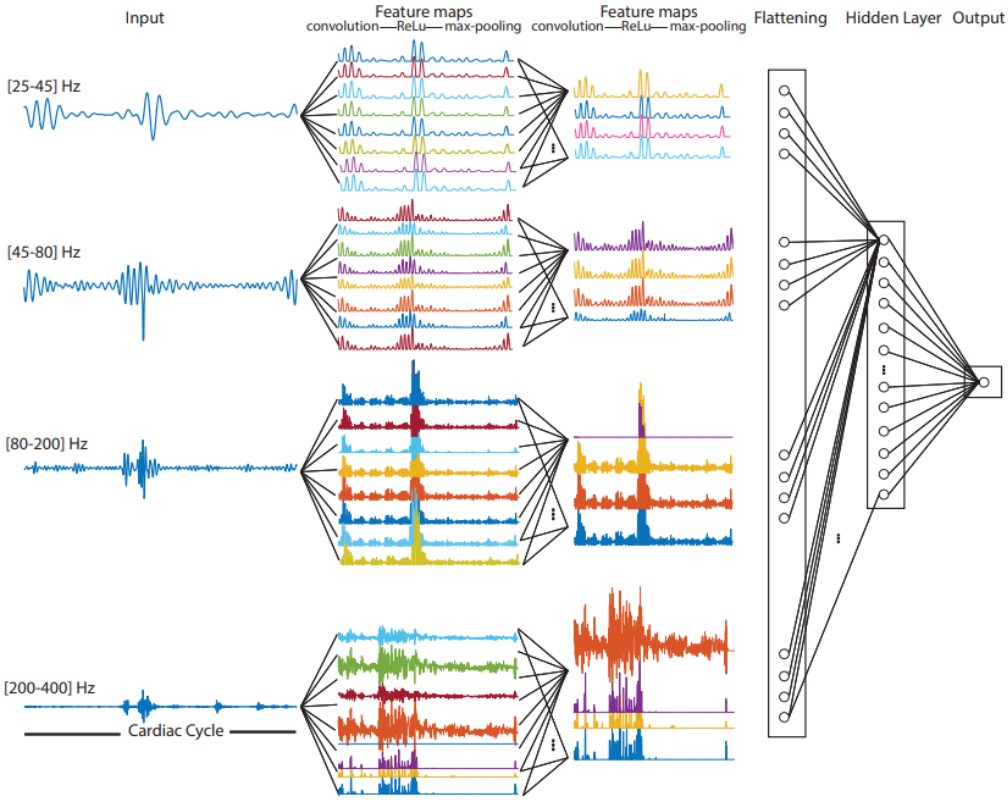


Figure 23: CNN architecture for classification of normal/abnormal heart sounds [58].

5. Hyperparameters:

The authors used the in-house training set to tune the hyperparameters. Resulting in a batch size of 1024, a learning rate of 0.0007 and training over 200 epochs. Training is stopped when the loss function stops decreasing.

6. Results:

The standalone AdaBoost-Abstain classifier obtained an overall score of 0.85, while the CNN approach achieved an overall score of 0.84. However, the classifier ensemble algorithm outperformed both approaches and obtained the highest score in the PhysioNet 2016 challenge. It achieved a sensitivity, specificity, and overall score of 0.94, 0.78, and 0.86, respectively. The results demonstrate the effectiveness of the proposed method in detecting abnormal heart sounds, and it provides a strong foundation for further development in this field.

2.2.2 Heart sound classification based on improved MFCC features and Convolutional Recurrent Neural Networks

1. Overview:

The objective of this study [3] was to develop a more accurate heart condition identification algorithm using an improved version of Mel-frequency cepstral coefficients (MFCCs) features and a Convolutional Recurrent Neural Networks (CRNN) based classifier. The algorithm utilized standard, 1st, and 2nd order MFCCs. Various combinations of CNN and RNN models were tested to compare their performance in classifying heart sounds.

2. Pre-processing:

To prepare the heart sound for analysis, a 5th order Butterworth bandpass filter is applied, which limits the frequency range to 25-400 Hz; and each sample is intercepted at 5 seconds. While pre-processing plays a crucial role, the focus of the paper is primarily on the feature extraction and pattern recognition techniques.

3. Feature Extraction:

The authors improved the Mel-frequency cepstral coefficients (MFCC) by applying pre-emphasis at the beginning to amplify high-frequency components. They also extracted first and second-order MFCCs to incorporate dynamic features.

4. Classification:

In this research paper, the authors compared the performance of two classification architectures, namely CRNN (Convolutional Recurrent Neural Network) and PRCNN (Parallel RCNN). The CRNN architecture consisted of three convolutional blocks, each comprising a Conv layer, a ReLU activation function, and a Max-pool layer, with Batchnorm and Dropout layers included. This was followed by an LSTM (or GRU which is another RNN variant) layer and a Fully Connected Layer with 64 neurons, and finally a Softmax function. The PRCNN architecture, on the other hand, used three convolutional blocks for the CNN part and a Max-pool layer followed by an LSTM (or GRU) layer for the RNN part. The outputs of the two blocks were concatenated into one feature vector and input to a Fully Connected layer with 32 neurons, followed by a Softmax layer.

5. Hyperparameters:

The authors partitioned the dataset into three subsets, with 75% used for training, 15% for validation, and 10% for testing. To address oversampling, they employed the k-means SMOTE algorithm to resample the training set. A dropout rate of 0.5 was found to achieve the highest accuracy, with a value of 0.97. The Adam optimizer was utilized with an initial learning rate of 0.01 and exponential decay rates of 0.9 and 0.999 for the first and second moment estimates, respectively.

6. Results:

The performance of the model is evaluated using four metrics: accuracy, recall, precision, and F1 score. According to the experimental results presented in Fig.24, the model that includes three convolutional layers and one LSTM layer yields the best results. Among the tested architectures, CRNN-a and PRCNN-a achieve the highest accuracy and F1 scores. These

findings suggest that combining CNNs with LSTMs is an effective approach for audio classification, and that the MFCC features used in this study are efficient.

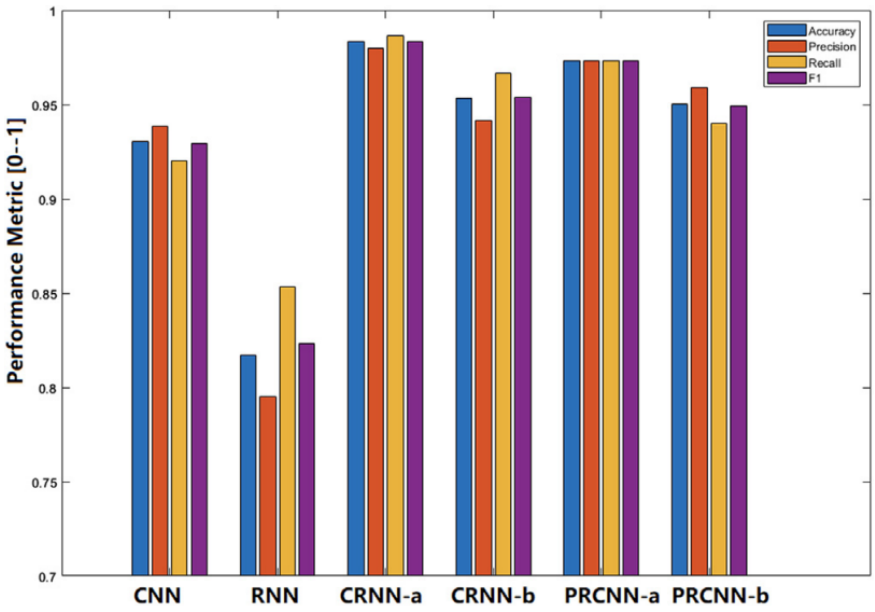


Figure 24: Classification performance of different network architectures of [3].

2.2.3 Classification of Heart Sounds Using Chaogram Transform and Deep Convolutional Neural Network Transfer Learning

1. Overview:

This study [67] aims to accurately classify PCG signals as normal or abnormal. The proposed approach involves applying a Chaogram transformation to the original PCG signal and using a pretrained DCNN (Deep CNN) model for classification.

2. Pre-processing:

The authors of this paper proposed a two-stage noise cancellation technique for improving PCG signal quality. In the first stage, a third-order Butterworth bandpass filter with a frequency range of 15 to 800 Hz is applied. In the second stage, a Spectral Subtraction denoising scheme with adaptive filters is used based on the noise power outside the expected range of the heart sound spectrum. This approach results in a final denoised PCG signal, obtained by subtracting the predicted noise power from the PCG spectrum.

3. Feature Extraction:

In this research, the authors focused on extracting a Chaogram image from the PCG signals using a Reconstructed Phase Space (RPS) which can be given by:

$$\overline{S_n} = \left[S_n, S_{n+\tau}, S_{n+2\tau}, \dots, S_{n+(d-1)\tau}\right], \quad (2.2.2)$$

where S_n is the signal, n is the sample index, d is the embedding dimension, and τ is the time delay. The authors determined that the optimal values for τ and d are 18 and 3, respectively.

The observations confirmed that the patterns formed in the RPS of a PCG are strongly correlated with the heart's functioning condition. The space of RPS was partitioned into 224×224×224 dimensions, and the frequency of points inside each cell was calculated to form a 3D tensor T. To obtain a more intuitive representation, three images (224×224) were then extracted by projecting T on the XY, XZ, and YZ planes.

Finally, these images act as the color channels of an RGB image to build the Chaogram. The Chaogram image is then enhanced to emphasize the weak details and provide a more comprehensive representation of the PCG signals.

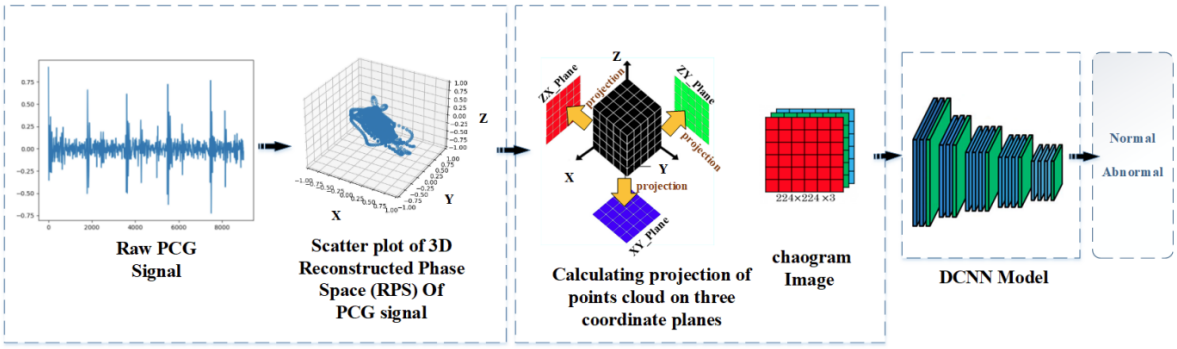


Figure 25: Flow diagram of the [67] approach for PCG signal classification.

To further improve the accuracy, the authors enriched the training set by transforming and deforming existing samples. The Chaogram image was rotated by 5 to 30 degrees, scaled with a factor of 1.05 to 1.15, and width and height shifted by 5 to 20 pixels. This process generated 20 new versions of each sample, increasing the size of the training set from 2,868 to 60,228 samples.

4. Classification:

To avoid overfitting on the small PhysioNet dataset, the authors suggest using pretrained networks with transfer learning as it can enhance the generalization of the model and accelerate the learning process. Four pretrained models (AlexNet, VGG16, InceptionV3, and ResNet50) were tested by fine-tuning only the last two layers on the dataset while keeping the weights of the other layers unchanged from their pretraining on the large-scale ImageNet dataset. To prevent overfitting, a dropout layer of 0.5 was also used in the model.

5. Hyperparameters:

The authors used different optimizer functions such as Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, and Ftrl, and varied the learning rate from 1×10^{-5} to 1×10^{-1} with steps of 1×10^{-5}. The batch size was varied from 50 to 500 with steps of 2 and the number of epochs varied from 50 to 400 with steps of 20. The optimal values for these hyperparameters were determined using the Bayesian optimization algorithm. The training process was stopped when there was no improvement in 20 consecutive epochs.

6. Results:

The algorithm's performance was evaluated using multiple metrics, including sensitivity, specificity, accuracy, and a score value that represents the sum of sensitivity and specificity divided by the total number of samples.

Network	Sensitivity	Specificity	Accuracy	Score
AlexNet	82.55	91.21	89.68	86.88
VGG16	83.36	91.49	90.05	87.43
InceptionV3	**84.49**	**91.63**	**90.36**	**88.06**
ResNet50	83.68	91.39	90.02	87.54

Table 1: Results of the Classification of the PCG signals with the method of [67].

The proposed approach demonstrated good performance in classifying PCG signals with the InceptionV3 model achieving a sensitivity of 84.49, specificity of 91.63, accuracy of 90.36, and a score of 88.06.

2.2.4 PCG classification through spectrogram using transfer learning:

1. Overview:

This study [60] proposes an approach for classifying PCG signals that combines signal processing and deep learning techniques. It involves several key processing steps and can perform binary or multiclass classification. The study highlights that multi-class PCG signal classification is possible with only 2-3 seconds of data and a hybrid classifier composed of a pretrained CNN and an SVM can significantly reduce training time. Finally, the hybrid classifier is complemented with a voting-based system for final classification.

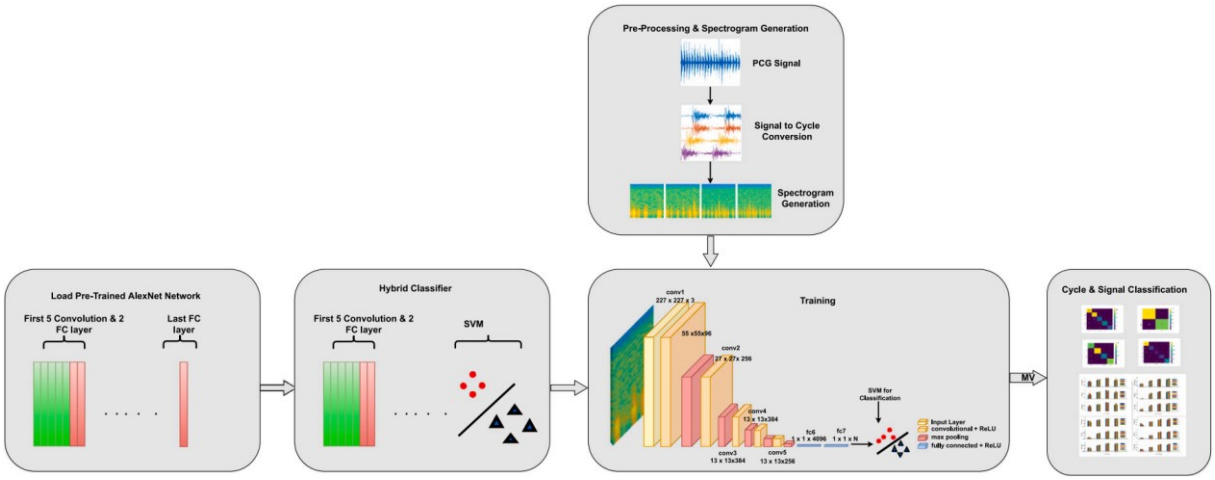


Figure 26: Flow diagram of the proposed CNN-SVM based classifier [60].

2. Pre-processing:

To prepare PCG signals for classification, a low-pass filter is used to decimate the signals by a factor of 21, resulting in a spectral range of 0-800 Hz. The signals are then segmented into fixed-duration cycles with an overlapping factor of $0.1\times fs$. To classify the signals effectively, at least 2-3 seconds of data are required to observe multiple heartbeats. However, to cover two or more cycles without knowledge of the person's age, at least 1 second of data is needed. Hence, Signals in the dataset with lengths less than 1 second are not considered. So, the authors decided to experiment with 4 different durations (1s, 2s, 3s, and 4s).

3. Feature Extraction:

The authors extracted the spectrogram representation from each cycle. The cycles are smoothed using a Kaiser window of length 256 samples with parameter $\beta = 5$. Overlapping samples of 220 and 1024 frequency points are used during spectrogram generation. The frequency used for the spectrogram is also decimated, resulting in spectrogram resolutions of 52×513, 110×513, 168×513, and 227×513 for durations of 1-4 seconds respectively.

4. Classification:

The researchers used the AlexNet CNN network for feature extraction for its deep architecture. However, as it requires a 227×227 input shape, the spectrograms are resized to that, but some details may be lost. The output of AlexNet is a 4096-dimensional feature vector, which is then used for classification.

The features are fed into an Error Correcting Output Codes (ECOC) model that relies on Support Vector Machines (SVM) which is a type of supervised machine learning algorithms.

The authors used a majority voting scheme to classify the signals based on the labels of their segments. The threshold for classification is set at $N_c \geq 0.5 \times N_t$, except for extra systole which has a lower threshold (0.3) due to its rarity compared to other cyclo-stationary heart sounds like S1 and S2.

5. Results:

The authors evaluated the dataset using different protocols for binary and multi-class classification. Binary classification included grouping all pathological classes as abnormal. The Performance was measured using Sensitivity. Average Sensitivity for multiclass classification was 96.32% for 2-second segments, while binary classification achieved 91.04% for 1-second segments. For signal classification (after the voting scheme), normal signals achieved 99.9% Sensitivity for 3-second segments, artifact signals got 100% across all segments, and extra heart sounds received 100% for 2 and 4-second segments. Murmurs achieved 99.9% Sensitivity for the 3-second segments. Results were averaged over 5-fold cross-validation.

2.3 Comparison and Analysis of PCG Classification Techniques

2.3.1 Overview

The field of PCG classification has gained significant attention due to the availability of good datasets through challenges like PhysioNet2016 and PASCAL. As a result, numerous research papers have been published proposing novel techniques and methods for various stages of the classification pipeline including preprocessing, segmentation, feature extraction, and classification. While there may be differences in parameter settings or model architectures, many of these papers share similarities such as denoising techniques, Segmentation methods, time-frequency features, and the use of CNNs and RNNs for classification. Table 2 provides a summary of the related works along with their performance results.

Year	Authors	Model	Segmentation	Features	Dataset	Results		
2016	Potes et al. [58]	AdaBoost & CNN	Springer	Time and Frequency Features, MFCC	PhysioNet	Sensitivity 94.24%	Specificity 77.81%	MAcc 86.02%
2020	Deng et al. [3]	CRNN	-	MFCCs	PhysioNet	Sensitivity 98.66%	Specificity 98.01%	MAcc 98.34%
2022	Harimi et al. [67]	DCNN	-	Chaogram	PhysioNet	Sensitivity 84.49%	Specificity 91.63%	MAcc 88.06%
2022	Abbas et al. [68]	CVT	-	CWTS	the phonocardiogram database	Sensitivity 99%	Specificity 99.5%	Acc 100%
2021	Ren et al. [59]	CNN	-	Log Mel Spectrogram	HSS		UAR 51.2%	
2022	S. Ismail et al. [60]	CNN-SVM	Fixed-duration cycles	Spectrogram	PASCAL		Sensitivity 99.9%	

Table 2: Summary of Related Works Techniques and Results

2.3.2 Discussion

Upon analyzing the results, it can be seen that Paper 2 by Deng et al. [3] achieved impressive accuracy results compared to Papers 1 and 3 despite not utilizing segmentation, indicating the effectiveness of MFCC features and RNN models in audio classification. In contrast, Paper 3 utilized a novel feature extraction technique with pre-trained CNN models to accelerate training, resulting in promising results.

Although Paper 4 used a distinct dataset, it demonstrated the effectiveness of spectrograms in audio data analysis as they provide time and frequency features. The study produced a remarkable average sensitivity of 0.999.

Based on the analysis, CNN models are the most common techniques employed for PCG classification, with RNN models used in certain cases. Bandpass filtering is frequently applied in the preprocessing stage to denoise the PCG signals. The Springer method is also a popular segmentation algorithm utilized in multiple papers (even in those not discussed here). MFCCs and Spectrograms are the most popular feature extraction techniques in audio classification.

However, determining the most effective techniques is challenging as each paper employs various approaches and datasets. Nonetheless, pre-trained CNN models and hybrid CNN-RNN models have shown potential in achieving high accuracy for PCG classification. Future research can explore more advanced deep learning techniques and larger datasets to improve the accuracy of PCG classification.

2.4 Summary

Chapter 2 provides an overview of previous works in the field of heart sound classification and the prediction of cardiovascular diseases using PCG signals. The chapter summarizes each paper, highlighting the techniques employed and the corresponding results obtained. By examining these works, we gain valuable insights into different approaches and methodologies used by researchers in addressing this problem. This chapter serves as a foundation for understanding the existing methods and identifying the most promising techniques, thus guiding us in choosing the techniques to implement for our project.

Chapter 3: Methodology

3.1 Introduction

In this chapter, we will present the various methodologies adopted for classifying phonocardiogram (PCG) signals in this project, including the preprocessing steps applied to the PCG signals, the design and implementation of the models, and the dataset used for training and testing these models. Specifically, we experiment with different feature extraction techniques and segmentation methods of the heart sounds data and discuss the different parameters and the decisions taken in detail.

3.2 Data Collection

To obtain a stable and generalized deep learning model for this research, a substantial amount of data samples is necessary. Fortunately, various publicly available heart sounds datasets exist, each with its advantages and disadvantages. Notably, the PASCAL dataset [69] and the PhysioNet 2016 dataset are the most widely used in related works. The PCG data is collected by multiple researchers, and challenges are organized with the aim of finding the most effective approach for detecting CVDs.

The PASCAL dataset [69] includes two sets: A which includes 176 files in .wav format and are categorized into four categories (Normal, Murmur, Extra Heart Sound, Artifact), and B which includes 656 files in wav format which are categorized into (Normal, Murmur, and Extrasystole).

The PhysioNet 2016 dataset [70] consisted of eight heart sound databases collected independently over a period of more than a decade. It contains a total of 3,126 heart sound recordings, lasting from 5 seconds to just over 120 seconds. The heart sounds are categorized into Normal and Abnormal sounds, the abnormal samples included cases of heart valve defects and coronary artery diseases.

In this study, the PhysioNet dataset was selected for model training due to its larger number of samples. However, it is highly unbalanced since it includes 2,575 samples labeled as normal and only 665 abnormal samples.

3.3 Proposed Methodology

3.3.1 Segmentation

Segmentation is a significant aspect in this project, primarily due to the periodic nature of heart sounds and the variations in recording lengths within the utilized dataset. As a result, we decided to experiment with two different techniques for segmentation.

Windowing: The first segmentation technique employed in this study is windowing, which involves dividing each audio file into multiple 5-second-long sound signals with a 0.5-second overlap. By adopting this approach, we aimed to increase the number of data samples and reduce the input size, ultimately enhancing the overall performance of the classification model. Fig.27 below illustrates the distribution of samples after performing the windowing segmentation.

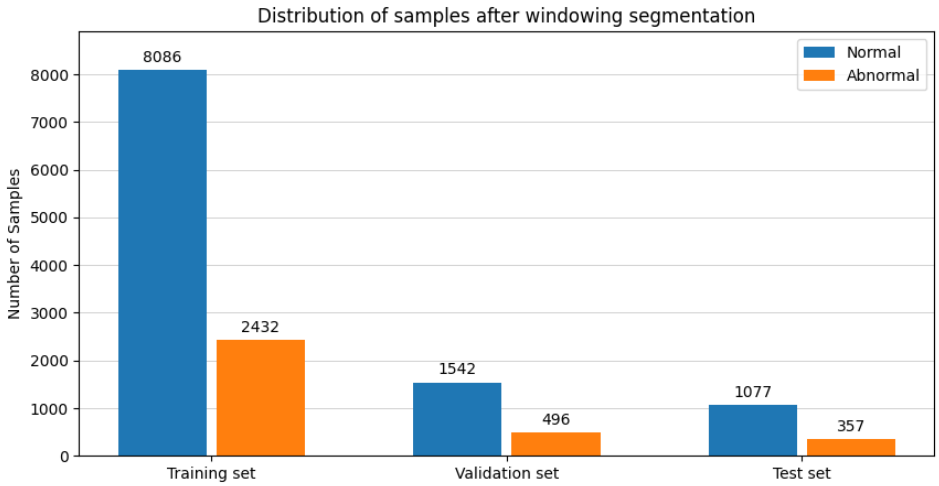


Figure 27: Distribution of samples after windowing segmentation.

Cycles: The second segmentation method employed in this study is based on heart cycle segmentation. We utilized the *Springer Segmentation Method* [71], which leverages machine learning techniques that employ Logistic Regression and Hidden Semi-Markov Model (HSMM) to accurately extract each individual heart cycle from the audio signals. This method identifies the different states of each cycle, including S1, Systole, S2, and Diastole. By precisely extracting heart cycles, we enable the model to analyze each cycle independently, which can provide meaningful insights for classification. Notably, in the PhysioNet Dataset, the longest heart cycle duration is approximately 2.5 seconds. Consequently, this segmentation approach generates more samples compared to the windowing segmentation method as can be seen in Fig.28. To ensure consistency in the dataset, cycles shorter than 2.5 seconds are zero-padded.

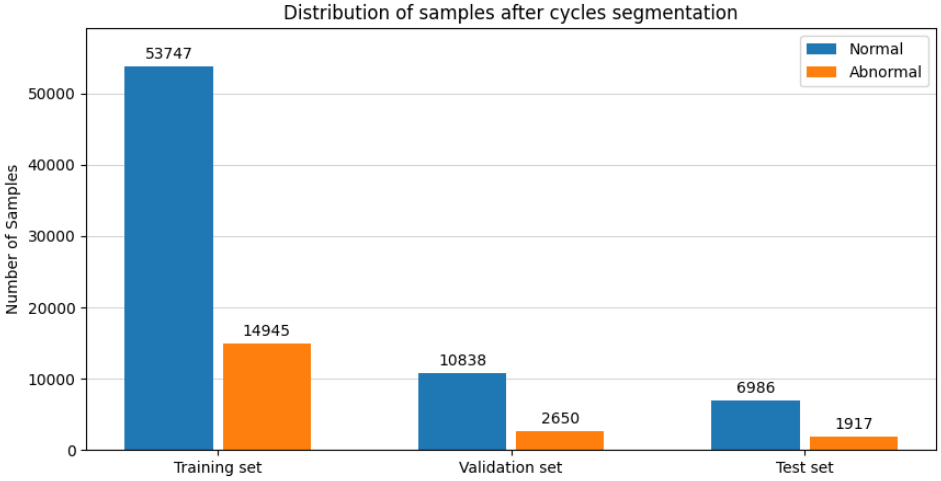


Figure 28: Distribution of samples after cycles segmentation.

3.3.2 Preprocessing

Data preprocessing is an important part of the PCG Classification task. It involves denoising the signal and performing different operations to the raw input data.

To begin, the input audio is converted into mono by merging the two channels of the audio into a single channel to ensure consistency in the data representation.

As the next step of the preprocessing, we address the sampling rate of the signals. For the windowed signals, we resampled the data to a sampling rate of 2000 Hz. This resampling helps to standardize the data and ensures compatibility with subsequent processing steps. However, for the cycle-segmented signals, the sampling rate is set to 1000 Hz. This adjustment is made to align with the Springer segmentation algorithm, which operates on signals sampled at 1000 Hz. By maintaining a consistent sampling rate, we facilitate the accurate analysis and classification of the segmented heart cycles.

For denoising purposes, we employed a third-order Butterworth bandpass filter with a low cut-off frequency of 15 Hz and a high cut-off frequency of 400 Hz. This configuration effectively removes low and high-frequency artifacts from the PCG signals, enabling the preservation of the fundamental sounds of the heart. Although this denoising process does not entirely eliminate noise, it ensures that the essential heart sounds remain intact for precise analysis as can be seen in Fig.29 which illustrates a PCG signal before and after denoising.

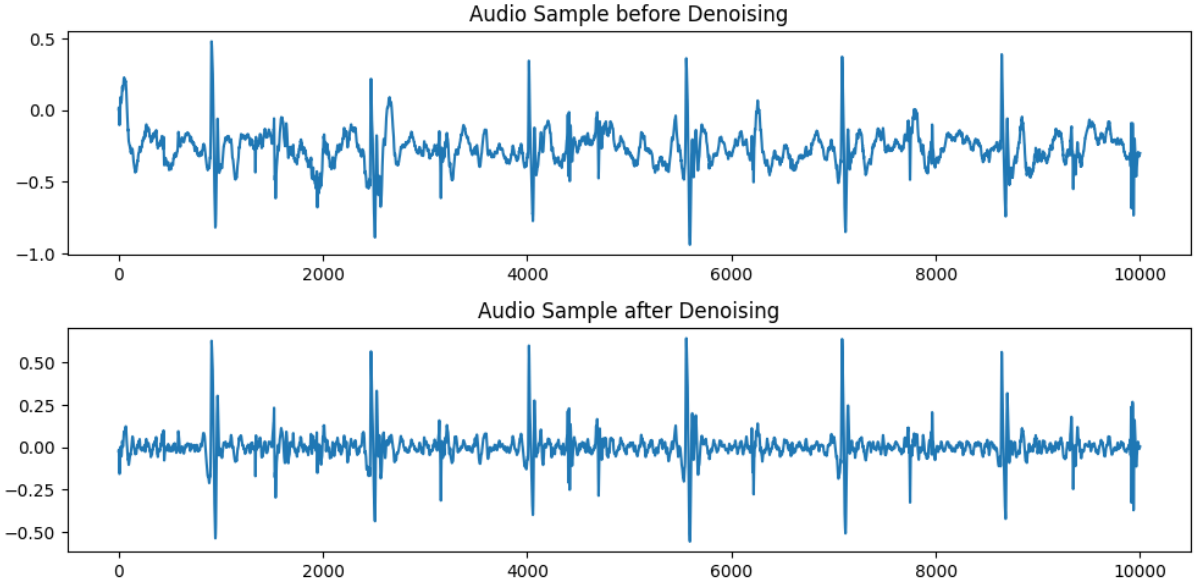


Figure 29: PCG sample before and after Denoising.

After the denoising process, the audio signals undergo pitch shifting. Pitch shifting is achieved by multiplying the amplitudes of the signal with a random value between 0 and 1. This step is implemented to enhance the generalization of the model to the data, allowing the model to better adapt to and handle varying amplitudes of the PCG signals.

3.3.3 Feature Extraction

Feature extraction is a crucial step in analyzing raw audio signals as it enhances the model's ability to understand the data and improves overall performance. In this project, we evaluated two popular feature extraction techniques commonly used in heart sound signal analysis: MFCCs (Mel-frequency cepstral coefficients) and Spectrogram features.

1. Mel-Frequency cepstral coefficients:

The first feature extraction technique employed in this project is the Mel-Frequency Cepstral Coefficients (MFCCs) inspired from [3]. MFCCs are chosen due to their ability to capture essential characteristics of the sound signal, which have proven valuable in various audio analysis tasks.

In our implementation, we derived standard MFCCs along with the delta and delta-delta MFCCs (1st and 2nd order derivatives). The inclusion of these derivatives allows us to highlight the changes in amplitudes over time within the signal.

To accommodate the different segmentation methods, we used different configurations for feature extraction. For the windowed signals, we employed a hop length of 64 samples, an FFT size of 256, 64 Mel filterbanks, and extracted 14 MFCCs. For the cycle-segmented signals, we utilized a hop length of 32 samples, an FFT size of 128, the same 64 Mel filterbanks, and extracted 14 MFCCs.

The three sets of features (standard MFCCs, delta MFCCs, and delta-delta MFCCs) are concatenated into a single tensor, with each set represented as a separate channel. Consequently, the resulting tensor has a shape of [3, 14, 157] for the windowed signals and [3, 14, 79] for the cycle-segmented signals. The Figures 30 and 31 visualize the extracted MFCC features for the windowed and cycle segments, respectively.

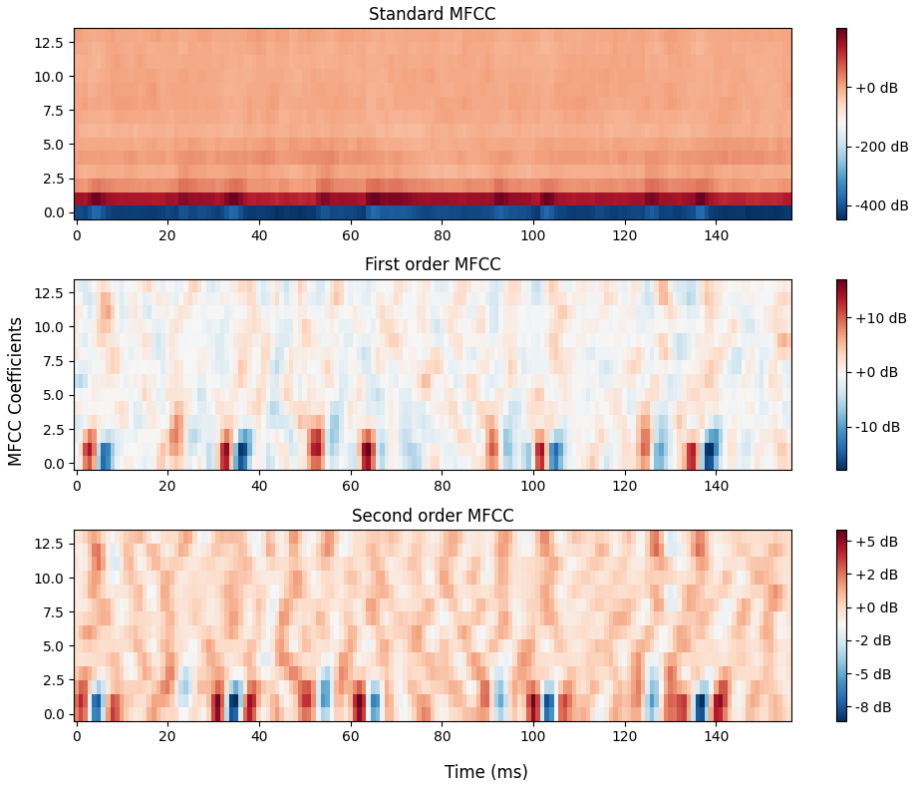
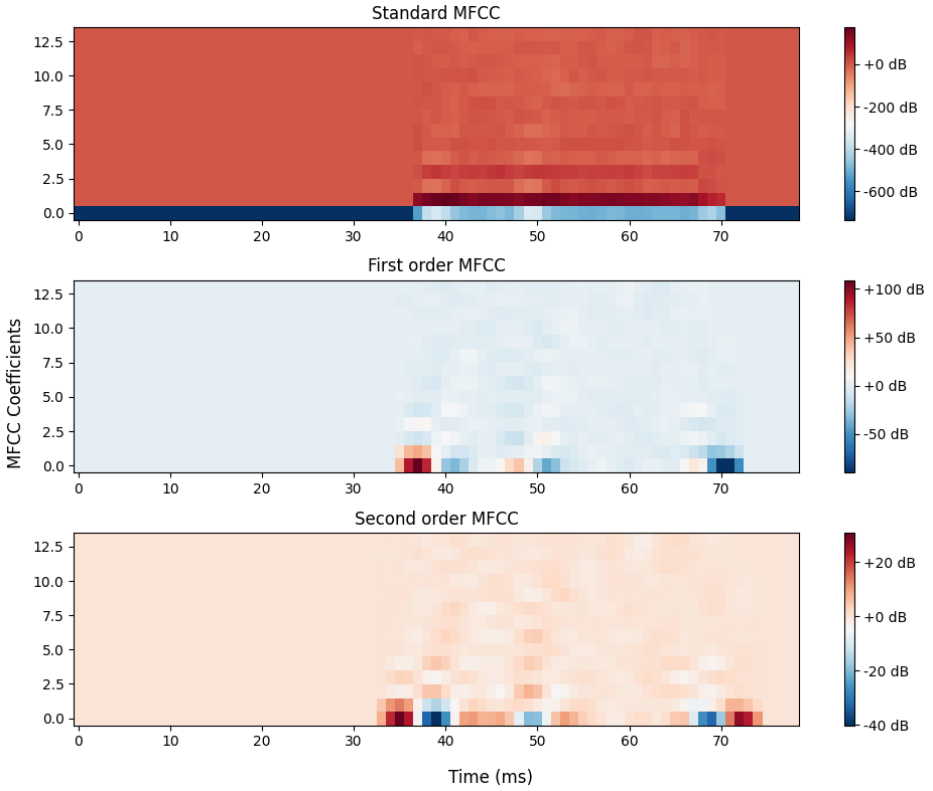


Figure 30: Visualization of the Standard, First order, and Second order MFCCs for the Windowed Signals.



q

Figure 31: Visualization of the Standard, First order, and Second order MFCCs for the Cycle-Segmented Signals.

2. Mel Spectrogram:

The second feature extraction method employed in this project is the Mel Spectrogram. Mel Spectrograms provide detailed information about the time-frequency relationship in the heart sounds, offering insights into the distribution of frequencies over time. To better represent the human perception of frequencies and amplitudes, the Mel frequency scale and Decibel scale are utilized.

Similar to the MFCC features, we also extracted the first and second-order derivatives of the spectrograms to capture the variations in the spectrogram features over time.

Different parameters were used for the spectrograms based on the segmentation methods. For the windowed signals, the parameters were set to a hop length of 64, an FFT size of 256, and 128 Mel filterbanks. Similarly, for the cycle-segmented signals, the parameters were adjusted to a hop length of 32, an FFT size of 256, and 128 Mel filterbanks.

In addition, we incorporated an effective augmentation technique specifically designed for spectrograms, known as Spectro-augment. This technique works by applying masking to a range of frequencies and time samples, adding horizontal and vertical bars to the spectrogram image. By introducing these masked regions, the model is encouraged to generalize well and avoid overfitting to specific frequencies and time stamps.

Following a similar approach to the previous method, the three sets of features derived from the Spectrogram, namely the Spectrogram, 1st order Spectrogram, and 2nd order Spectrogram, are combined into a single tensor. Each set is treated as a separate channel within the tensor, resulting in a tensor shape of [3, 128, 157] for the windowed signals and [3, 128, 79] for the cycle-segmented signals. The Figures 32 and 33 visualize the extracted Mel Spectrogram features for the windowed and cycle segments, respectively.

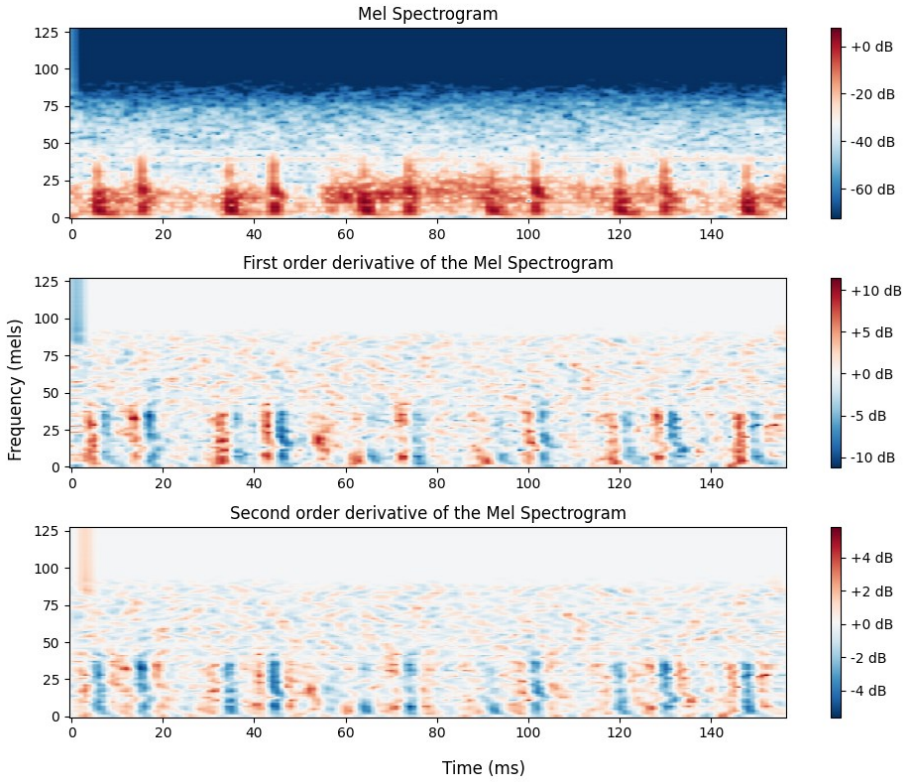


Figure 32: Visualization of the Mel Spectrogram representation for the Windowed Signals.

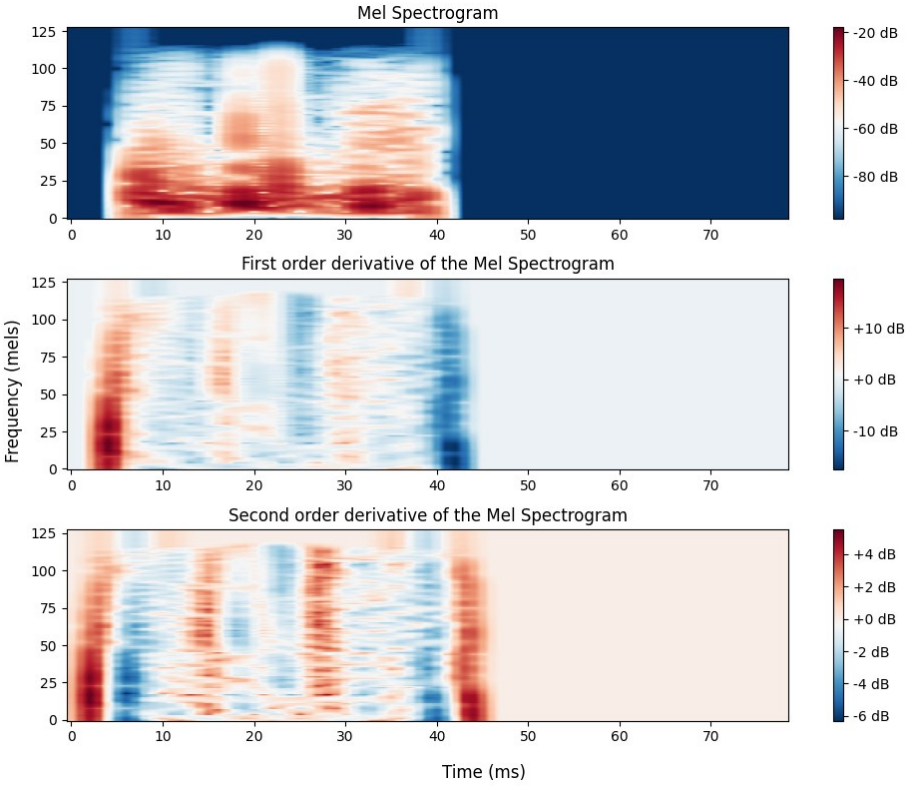


Figure 33: Visualization of the Mel Spectrogram representation for the Cycle-Segmented Signals.

3.3.4 Classification and Network Architecture

In this research, we adopted a hybrid CNN and LSTM model for classification, drawing inspiration from relevant works in the field [3]. The decision to combine these two architectures originated from the nature of the extracted features, which possess both sequential and image-like characteristics.

According to [5], a CNN is highly effective at extracting deep features from MFCCs or Spectrograms by performing convolutions in the time and frequency domains of heart sound signals. Meanwhile, an LSTM is proficient at learning temporal patterns across various frequencies in the PCG signals. Combining CNN and LSTM models creates a mutually beneficial fusion, complementing each other's strengths [5].

The proposed hybrid model integrates the CNN and LSTM networks in a parallel manner. Both networks receive the same input, and their outputs are concatenated. Subsequently, the concatenated output is processed through another feed-forward network for final classification.

To ensure a comprehensive analysis, the model takes the segmented windows or cycles from the main signal as input. Each segment is classified individually during training, and a voting scheme similar to the approach in [60] is employed. The majority vote, with a predetermined weighted threshold, is used to make the final decision regarding whether the signal should be classified as normal or abnormal. Fig.34 demonstrates the network architecture of the adopted model.

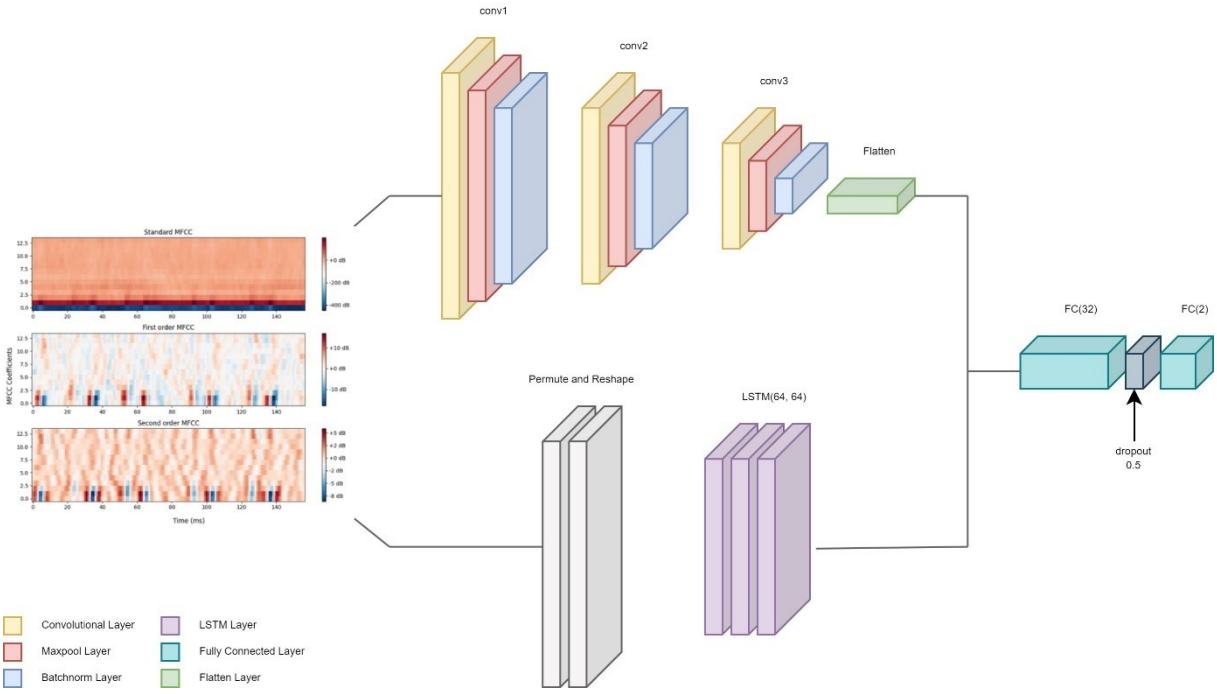


Figure 34: Visualization of the proposed Hybrid CNN-LSTM model architecture.

The model architecture varies based on the segmentation method and input features to ensure compatibility. In general, the model employs a CNN network with 3 convolutional blocks and a flatten layer. The first block includes a 3-channel conv layer with a 3x3 kernel

size, followed by ReLU activation, a 2x2 MaxPool2d layer, and batch normalization. The other two blocks follow a similar structure, with the second block outputting 32 features and utilizing a 4x4 max pooling, and the third block outputting 64 features. For MFCC features, the last two conv blocks omit max pooling since the dimensions are already small. The CNN output is flattened and concatenated with the LSTM output, which is obtained from a 64-layer LSTM network. Before inputting to the LSTM layers, the input is reshaped and permuted to [batch_size, seq_length, n_features × in_channels] by concatenating the different channels hence the LSTM requires an input size of (n_features × in_channels). The LSTM network's hidden states and cell states are initialized with zeros, and the output is taken from the last state of the LSTM layers.

The fully connected network includes a layer with a number of neurons based on the segmentation and features used (9,408 for windowed MFCCs, 3,136 for windowed spectrograms, 4,416 for cycle-segmented MFCCs, 1,216 for cycle-segmented spectrograms). This layer outputs 32 neurons, followed by a ReLU activation function and a dropout layer with a rate of 0.5. Finally, a final fully connected layer is present, which outputs 2 classes for classification. The predicted class is determined by taking the index of the maximum value from the output (0 for normal, 1 for abnormal).

3.3.5 Training Procedure

Before proceeding to the model training, the dataset is preprocessed by reading the original audio data from the PhysioNet dataset, segmenting them, and converting them into the corresponding feature representations. These processed datasets are then stored as tensors for efficient and faster training, eliminating the need to process each sample individually during training. This results in the creation of four new datasets, each representing a different segmentation and feature extraction technique.

After that, a DataLoader with a batch size of 64 is utilized to divide the dataset into mini-batches. To address the issue of class imbalance, a WeightedRandomSampler is employed, which oversamples the minority class and undersamples the majority class. Each mini-batch tensor is normalized before being used for training. Additionally, both the model and input tensors are transferred to the GPU device to enhance the training performance.

During training, a CrossEntropyLoss function is utilized in combination with an Adam optimizer and a learning rate of 0.001. To optimize the learning rate, a OneCycleLR scheduler is employed, increasing the rate linearly until it reaches a maximum value of 0.01, after which it starts decreasing. The number of training epochs is set to 500, with early stopping implemented if the validation accuracy fails to improve for 30 consecutive epochs.

3.3.6 Experimental Setup

The experimental environment was set up on a local machine equipped with an 11th Gen Intel(R) Core(TM) i7-11800H CPU with 8 cores, an NVIDIA RTX 3070 GPU, and 16GB of RAM. The software environment relied on Python as the main programming language, chosen for its popularity in the field of artificial intelligence and the availability of the various libraries:

a. PyTorch:

PyTorch, an open-source machine learning framework, was utilized for its flexibility in expressing deep learning models using Python. It is commonly used in natural language processing and computer vision tasks, known for its debugging capabilities and ease of use.

b. TorchAudio:

TorchAudio, a PyTorch library for audio and signal processing, offered essential functionalities for the project. It facilitated tasks such as audio file loading, resampling, and transformations like MelSpectrogram, AmplitudeToDB, and MFCCs. Additionally, it aided in computing deltas and implementing time and frequency masking.

c. TorchMetrics:

TorchMetrics is another library for PyTorch that provides a collection of metrics implementations and an easy-to-use API.

d. SciPy:

SciPy is a free and open-source Python library used for scientific computing and technical computing. It was mainly used in our project for its Butterworth filter function.

e. Pandas:

Pandas, a Python library for data manipulation and analysis, proved useful in efficiently handling and manipulating datasets. It offered data structures and operations designed for numerical tables and time series.

f. NumPy:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

g. Matplotlib:

Matplotlib is a plotting library for the Python programming language. It is used to visualize the different inputs and plot the training performance.

Additionally, the Springer Segmentation algorithm was executed using MATLAB, while the project implementation was carried out using Jupyter Notebook and Visual Studio Code.

3.4 Summary

In this methodology chapter, we presented our approach for classifying PCG signals using a CNN-LSTM hybrid model. We discussed the data collection process and the dataset utilized in our research. Additionally, we described the two segmentation techniques and feature extraction methods employed. We outlined the training procedure, including the various hyperparameters involved. Lastly, we provided an overview of the experimental setup and the tools utilized throughout the study.

Chapter 4: Results and Discussion

4.1 Introduction

In Chapter 4, we present the results and discussion of our PCG Classification, focusing on the performance evaluation and comparisons of different experiments. Firstly, we introduce the performance metrics employed to assess the effectiveness of our methodologies. Subsequently, we discuss the outcomes of each experiment, examining the performance of windowed segments, cycle segments, MFCCs representation, and Spectrogram representation individually. Additionally, we thoroughly compare the results obtained from the two segmentation techniques and the two feature extraction techniques, highlighting their relative strengths and weaknesses. Through this in-depth analysis, we aim to gain insights into the optimal approaches for PCG classification.

4.2 Performance Metrics

During the evaluation phase, a useful measure that can be employed is the confusion matrix. This matrix provides a clear overview of the network's performance by displaying the actual and predicted output classes in a table format. The matrix contains information on the number of correctly and incorrectly predicted samples for each class. The confusion matrix labels correctly predicted samples as "True" and incorrectly predicted samples as "False". While the predicted values are labeled either "Positive" or "Negative," which correspond to "Abnormal" and "Normal" respectively.

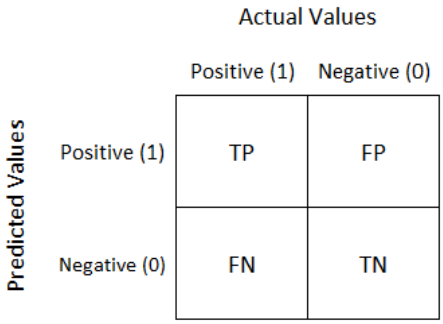


Figure 35: Confusion matrix.

There are various metrics available, and the choice of a particular metric depends on the specific task. Accuracy is one of the most commonly used metrics, which measures the proportion of correctly classified samples in the set. However, depending on the problem, other metrics such as precision, recall, F1-score, or mean squared error (MSE) may be more relevant.

- **Accuracy:** It is a metric that provides an overall measure of how well a model performs across all classes, and it is particularly useful when all classes are equally important. It is given by:

$$Accuracy = \frac{n\_correct\_predictions}{n\_total\_predictions}$$

- **Precision:** measures the accuracy of a model in correctly classifying a sample as positive in binary classification [72].

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

- **Sensitivity:** it is also known as Recall; it measures the model's ability to correctly identify positive samples. A higher recall indicates that the model is able to detect more positive samples correctly.

$$Sensitivity = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

- **Specificity:** measures the proportion of true negatives that are correctly identified by the model.

$$Specificity = \frac{True_{negative}}{True_{negative} + False_{positive}}$$

- **F1 Score:** The F1 score combines precision and recall such that maximizing the F1 score implies simultaneously maximizing both precision and recall.

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

One additional metric used in this research is the MAcc introduced in [70] which is used as the overall score in the challenge. It involves the Sensitivity and Specificity metrics as follows:

$$MAcc = \frac{Sensitivity + Specificity}{2}$$

4.3 Results

4.3.1 Experiment 1: Windowed Segments with MFCC Features

In the first experiment, we applied the windowing segmentation technique combined with the MFCC feature extraction method. The implemented model consisted of a total of 2,449,250 parameters. The training process was monitored, and the model stopped training at epoch 151 due to a consecutive decrease in the validation accuracy. The full training process took well above 43 minutes. Fig.36 illustrates the training and validation accuracy curves for this experiment.

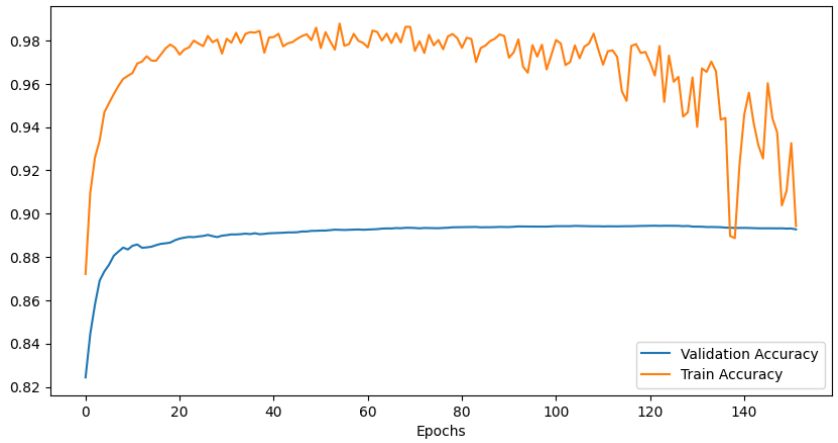


Figure 36: Training and Validation accuracy curves for the first experiment.

From the Fig.36 we can see that the model was doing well on the training data and descent on the validation data. It took approximately 15 to 20 epochs for the model to converge and gradually improve over time, eventually stabilizing and experiencing a slight decrease around epoch 130. The highest achieved validation accuracy was recorded at 89.44%.

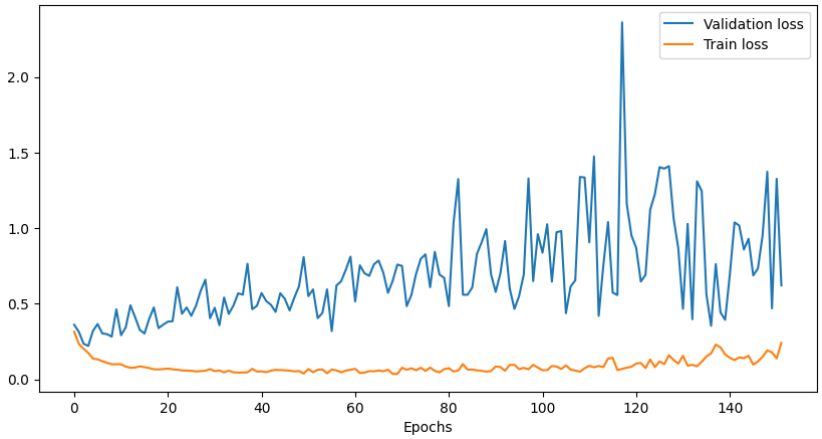


Figure 37: Training and Validation loss curves for the first experiment.

The loss curves depicted in Figure 37 demonstrate similar trends to the accuracy curves. The training loss revealed a steady increase after epoch 120. As for the validation loss, it showed higher values compared to the training loss with a great noise, making it less informative for our analysis.

After evaluating the trained model, the confusion matrix is displayed in Fig.38:

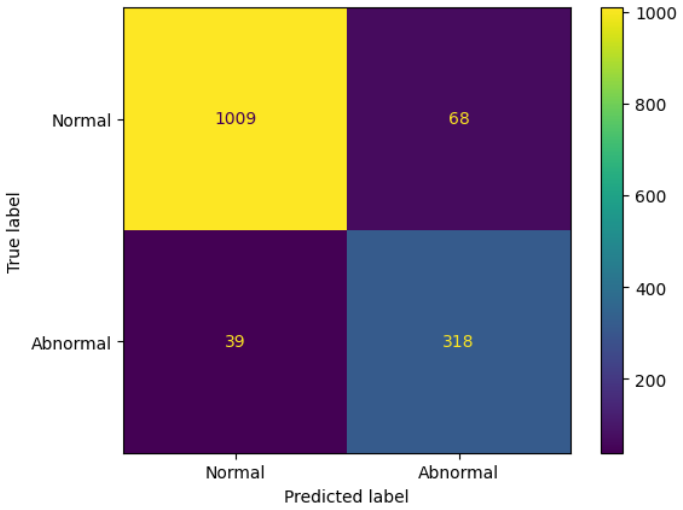


Figure 38: Confusion matrix of the first experiment.

The evaluation results are given in Table 3:

Accuracy	Precision	Specificity	Sensitivity	F1-Score	MAcc
92.54%	82.38%	93.69%	89.08%	85.60%	91.38%

Table 3: Evaluation results of the first experiment.

The previously presented results were obtained before implementing the voting scheme. After applying the voting scheme, the confusion matrix of the first experiment can be seen in Fig.39 below:

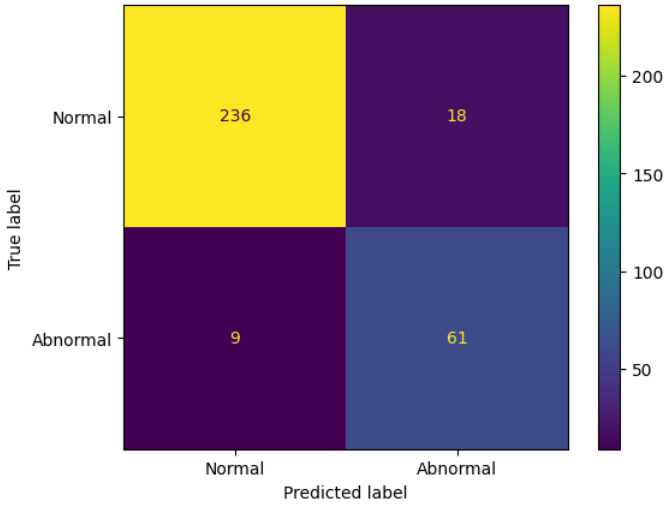


Figure 39: Confusion matrix of the first experiment after the voting scheme.

Multiple threshold values were tested, and the optimal threshold value was determined to be *tr=1.25×n_cycles/2*. The evaluation results following the implementation of the voting scheme are presented in Table.4 .

Accuracy	Specificity	Sensitivity	MAcc
91.67%	92.91%	87.14%	90.03%

Table 4: Evaluation results of the first experiment after the voting scheme.

In the first experiment, the evaluation results indicate a good performance of the model. The confusion matrix demonstrates a well-balanced prediction, with only 68 normal samples incorrectly classified as abnormal and 39 abnormal samples incorrectly classified as normal. The achieved accuracy of 93% and a score of 92% highlight the effectiveness of the approach in identifying abnormalities. The model reveals a specificity of 94%, indicating accurate predictions of normal samples, while the sensitivity of 89% shows relatively lower accuracy in predicting abnormal samples, but still reasonable.

After applying the voting scheme, the accuracy slightly decreased due to the reduced number of samples. However, the confusion matrix continues to show favorable results, with only 27 signals being wrongly predicted. This solidifies the effectiveness of the voting scheme. It is worth noting that the absence of abnormalities in certain parts of the signals, where an abnormality can only be observed in one cycle, may have contributed to these results. Although the accuracy did not reach the desired level, the voting scheme remains a robust method for combining predictions and improving overall classification performance.

4.3.2 Experiment 2: Cycle Segments with MFCC Features

In the second experiment, we employed the cycles segmentation technique along with the utilization of MFCC features. The CNN-LSTM model utilized in this experiment consisted of a total of 2,289,506 parameters. The model was trained for 129 epochs which took around 215 minutes. Figures 40 and 41 depict the accuracy and loss curves for this experiment.

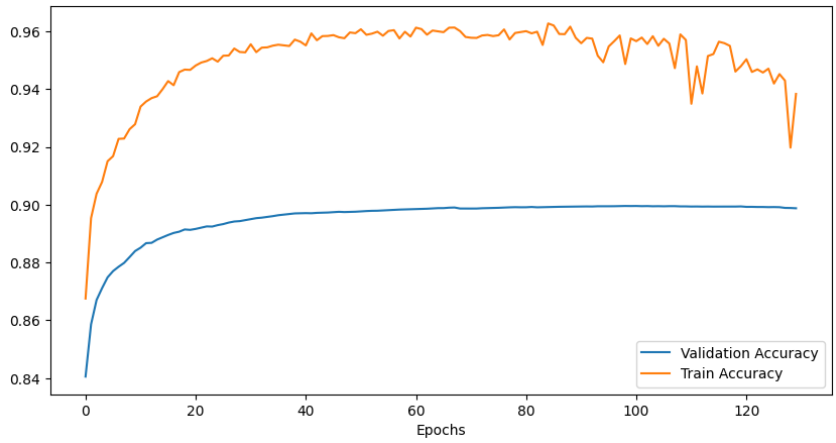


Figure 40: Training and Validation accuracy curves for the second experiment.

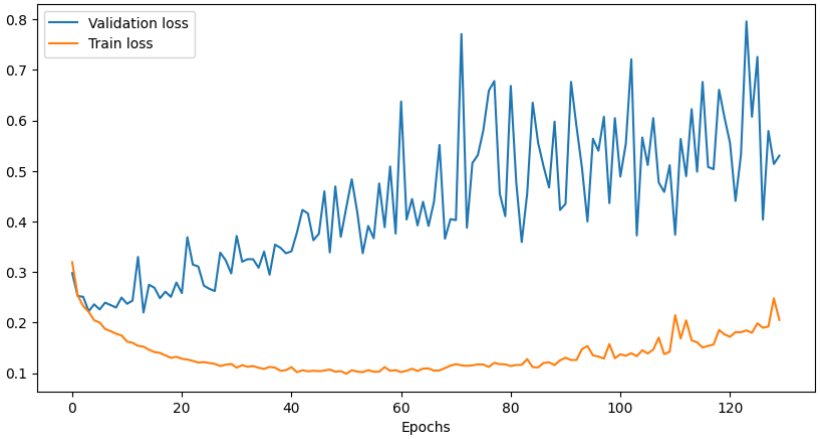


Figure 41: Training and Validation loss curves for the second experiment.

The accuracy and loss curves demonstrate favorable outcomes, with the model reaching its highest point at around epoch 100, achieving a validation accuracy of 89.96%. Following this peak, the training accuracy notably decreased, possibly indicating a higher learning rate due to the scheduler employed in the training process.

The trained model is evaluated on the testing set and the confusion matrix of the second experiment is given in Fig.42 by:

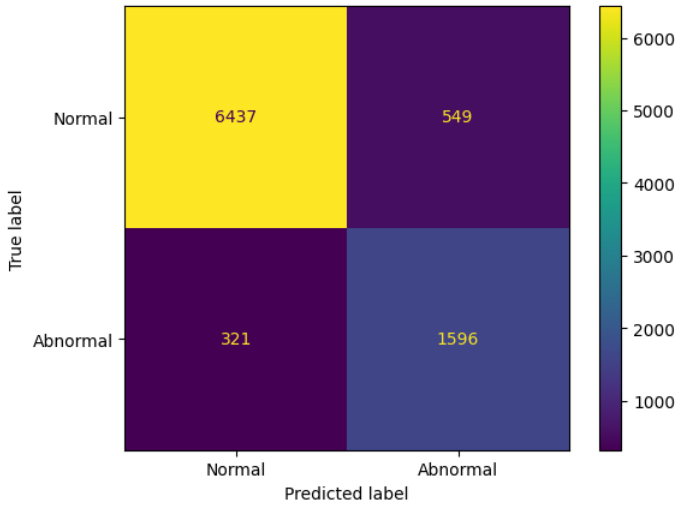


Figure 42: Confusion matrix of the second experiment.

The evaluation results are given in Table 5:

Accuracy	Precision	Specificity	Sensitivity	F1-Score	MAcc
90.23%	74.41%	92.14%	83.26%	78.58%	87.70%

Table 5: Evaluation results of the second experiment.

After applying the voting system, we found that the default threshold of *n_cycles/2* provides the highest accuracy during evaluation. The confusion matrix and metrics results are presented in the Fig.43 and table 6, respectively.

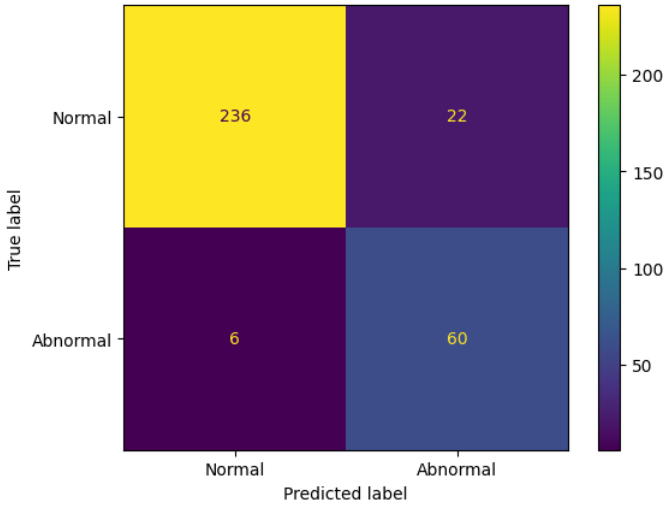


Figure 43: Confusion matrix of the second experiment after the voting scheme.

Accuracy	Specificity	Sensitivity	MAcc
91.36%	91.47%	90.91%	91.19%

Table 6: Evaluation results of the second experiment after the voting scheme.

In the second experiment, the use of cycle segmentation resulted in a larger number of samples compared to the first experiment, leading to a slower training process. The evaluation results indicate a decent performance of the model. The confusion matrix demonstrates a nearly balanced prediction, where 549 normal cycles were incorrectly classified as abnormal out of 6,986, and 321 abnormal samples were incorrectly classified as normal out of 1,917.

However, the performance metrics in this experiment are slightly less promising compared to the first experiment. The accuracy achieved is 90%, with a score of 88%, indicating that the MFCC features extracted from cycle segments may be less representative compared to windowed segments. The specificity of 92% suggests that the model performs better at accurately predicting normal samples, while the sensitivity of 83% indicates poorer accuracy in predicting abnormal samples.

After applying the voting scheme, the accuracy improved to 91%, and both specificity and sensitivity reached balanced values of 91% each. This shows that the model achieves a more balanced performance in predicting both classes after the voting scheme is applied.

4.3.3 Experiment 3: Windowed Segments with Spectrogram Features

In the third experiment, we employed the windowed segmentation technique along with the mel spectrogram features. The model included a total of 2,336,098 parameters and was

trained for 113 epochs which took around 34 minutes. The accuracy and loss curves and given in the Figures 44 and 45.

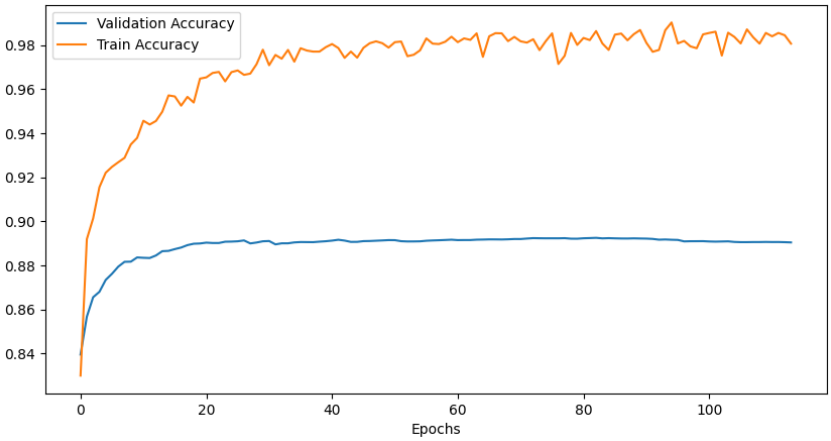


Figure 44: Training and Validation accuracy curves for the third experiment.

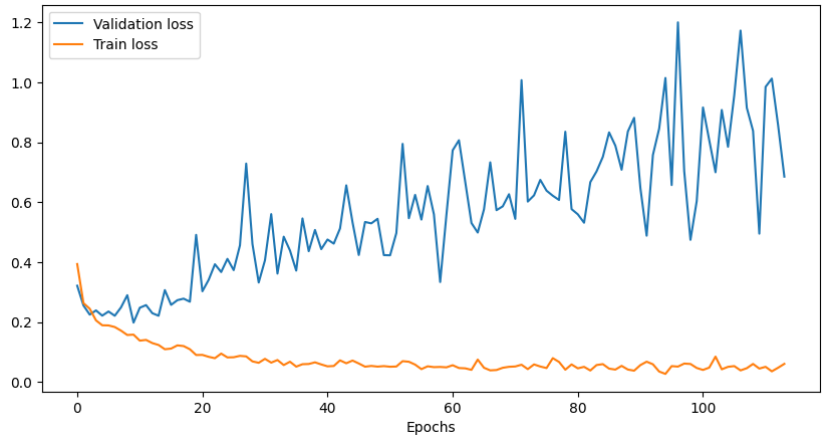


Figure 45: Training and Validation loss curves for the third experiment.

In this particular experiment, the model demonstrated better performance on the training set, as there was no decline in accuracy over time. However, a decrease in validation accuracy was observed, which could potentially indicate overfitting. Fortunately, the training process was stopped early, and we saved the model's state at epoch 100 to use it for further evaluation. The model peaked at the epoch 82 with a validation accuracy of 89.25%.

The confusion matrix of this experiment is shown in Fig.46:

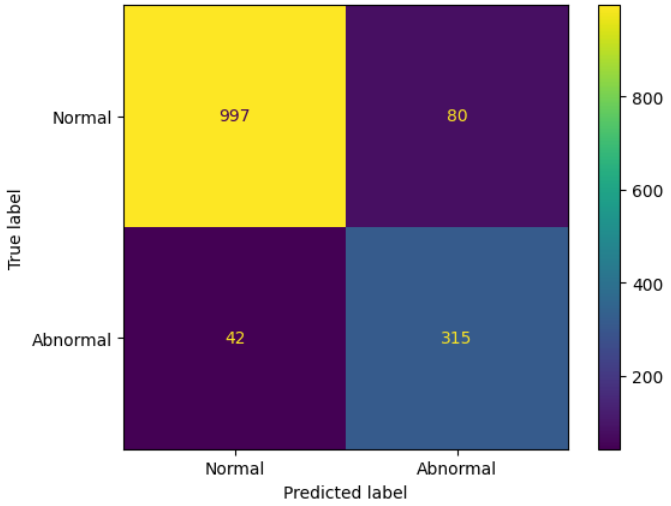


Figure 46: Confusion matrix of the third experiment.

The evaluation results are given in Table 7:

Accuracy	Precision	Specificity	Sensitivity	F1-Score	MAcc
91.49%	79.75%	92.57%	88.24%	83.78%	90.40%

Table 7: Evaluation results of the third experiment.

Using a threshold *tr=0.8×n_cycles/2* we achieved an accuracy of 89.20% after the voting system. Fig.47 and Table 8 present the confusion matrix and the metrics results of the evaluation.

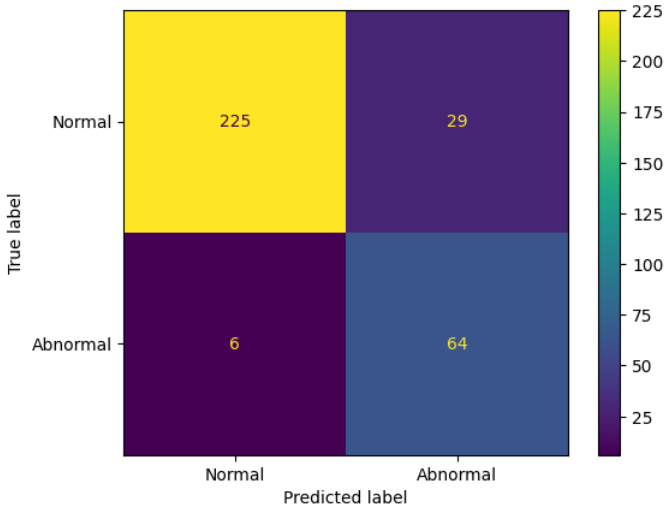


Figure 47: Confusion matrix of the third experiment after the voting scheme.

Accuracy	Specificity	Sensitivity	MAcc
89.20%	94.09%	82.86%	90.01%

Table 8: Evaluation results of the third experiment after the voting scheme.

In the third experiment, we utilized windowed segments with spectrogram features. The performance metrics achieved in this experiment are somewhat less favorable when

compared to the results of the first experiment. The accuracy achieved is 91%, with a score of 90%, indicating that spectrograms serve as good representations of heart sound signals. However, it is worth noting that MFCC features are more effective for windowed segments.

The specificity of 93% indicates that the model excels at accurately predicting normal samples, while the sensitivity of 88% suggests that it is somewhat less accurate in predicting abnormal samples. Although the model demonstrates good performance in identifying normal samples, it faces challenges in effectively detecting abnormalities.

After applying the voting scheme, there was a decrease in accuracy to 89%, and the specificity and sensitivity values became unbalanced, with 94% and 83%, respectively. This implies that the model performs relatively well in predicting abnormal samples but struggles with accurately identifying abnormalities. These findings suggest that spectrogram features, while informative, may not capture all the distinctive characteristics necessary for precise abnormality prediction in this context.

4.3.4 Experiment 4: Cycle Segments with Spectrogram Features

For the final experiment, we employed the cycles segmentation technique along with the spectrogram feature extraction method. The utilized model consisted of a total of 2,274,658 parameters. The training process extended for 200 epochs, but it was stopped early due to slow convergence and excessive time consumption. The training process took approximately 450 minutes to complete.

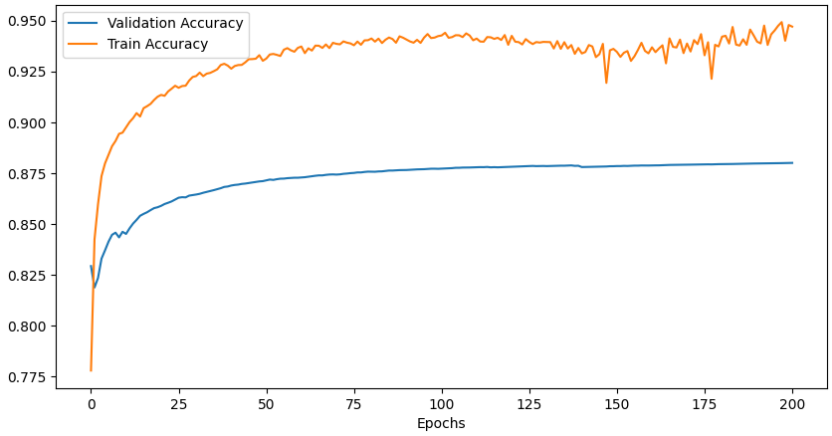


Figure 48: Training and Validation accuracy curves for the fourth experiment.

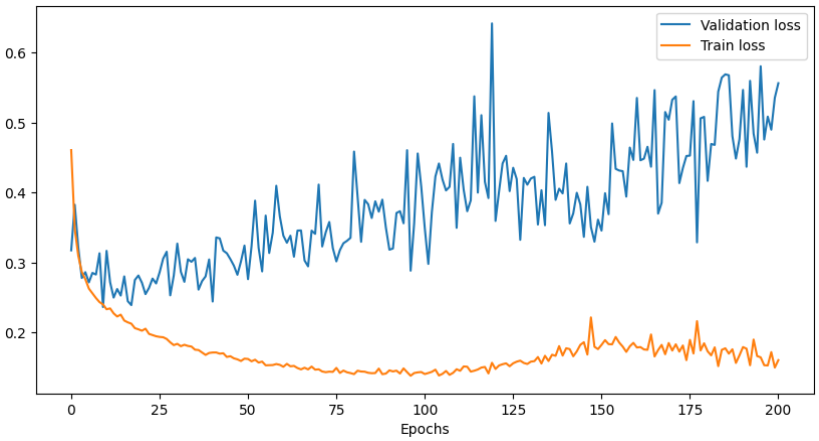


Figure 49: Training and Validation loss curves for the fourth experiment.

The depicted Figures 48 and 49 illustrate promising training results for the last experiment. The validation accuracy consistently increased throughout the training process, while the training accuracy displayed minor fluctuations around epoch 150 before resuming its climb. This trend is also reflected in the loss curves, where the loss steadily decreased. The experiment achieved a maximum validation accuracy of 88.00%.

Upon completing the training of the model, we proceeded to evaluate its performance using the saved states at epochs 50, 100, 150, and 200. Interestingly, the evaluation results revealed that the model's state at epoch 50 exhibited the highest performance in terms of overall accuracy and MAcc. Fig.50 below shows the confusion matrix of the evaluation before the voting scheme.

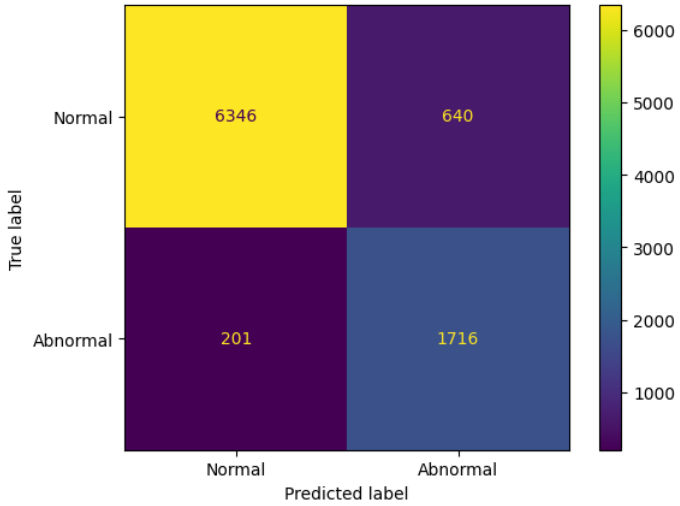


Figure 50: Confusion matrix of the fourth experiment.

The evaluation results are given in Table 10:

Accuracy	Precision	Specificity	Sensitivity	F1-Score	MAcc
90.55%	72.84%	90.84%	89.51%	80.32%	90.18%

Table 9: Evaluation results of the fourth experiment.

A threshold value of 0.67×*n_cycles/2* was found to give the best results in terms of accuracy for the voting system. Fig.51 and Table 10 present the confusion matrix and evaluation results.

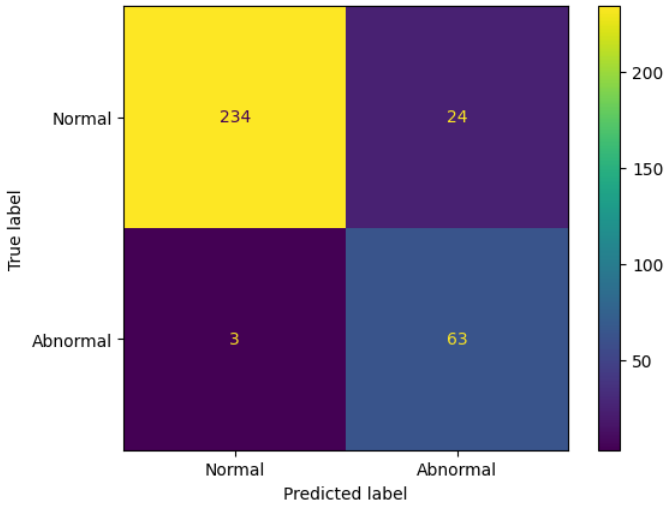


Figure 51: Confusion matrix of the fourth experiment after the voting scheme.

Accuracy	Specificity	Sensitivity	MAcc
91.67%	90.70%	95.45%	93.08%

Table 10: Evaluation results of the fourth experiment after the voting scheme.

In the final experiment, which involved cycle segments with spectrogram features, we observed favorable evaluation results compared to the second experiment even though the training process was relatively slower. Only 201 abnormal samples were incorrectly predicted as normal, while 640 normal samples were wrongly classified as abnormal. The performance metrics achieved a balanced outcome, with an accuracy of 91% and a score of 90%. Both the specificity and sensitivity values reached 91% and 90%, respectively, indicating that the model excels in predicting both classes in a balanced manner.

Upon implementing the voting scheme, the accuracy further improved to 92%. Additionally, the specificity and sensitivity values demonstrated a reasonable balance, with 90% and 95%, respectively. This indicates that the model performs significantly better in predicting abnormal samples while still maintaining good performance in detecting normal samples. This is particularly important as it signifies the model's enhanced ability to identify abnormalities accurately.

Overall, the final score of 93% highlights the effectiveness of using cycle spectrograms as representations of heart sounds. This experiment demonstrates that cycle spectrograms capture the acoustic characteristics of the heart more effectively, leading to improved performance in the classification of heart sound abnormalities.

Table 11 provides a summary of the evaluation and performance results for the four conducted experiments.

Model details						Before Voting				After Voting			
Model	**Segmentation**	**Features**	**Complexity**	**Epoch**	**Speed** (epoch/min)	**Accuracy**	**Sensitivity**	**Specificity**	**MAcc**	**Accuracy**	**Sensitivity**	**Specificity**	**MAcc**
CNN-LSTM (No Pooling)	Windowing	MFCCs	2,449,250	100	**3.5**	**92.54**	89.08	93.69	**91.38**	91.67	87.14	92.91	90.03
CNN-LSTM (No Pooling)	Cycles	MFCCs	2,289,506	100	0.6	90.23	83.26	92.14	87.70	91.36	90.91	91.47	91.19
CNN-LSTM	Windowing	Mel Spectrogram	2,336,098	100	3.3	91.49	88.24	92.57	90.40	89.20	82.86	94.09	90.01
CNN-LSTM	**Cycles**	**Mel Spectrogram**	**2,274,658**	**50**	0.4	90.55	89.51	90.84	90.18	**91.67**	**95.45**	**90.70**	**93.08**

Table 11: Evaluation and Performance results of the different experiments.

4.4 Discussion

4.4.1 Comparison between Segmentation techniques

By analyzing Table 12 and interpreting the results of the experiments, it becomes clear that windowing techniques offer a significant advantage in terms of training speed. This is because windowed segments have fewer samples and contain more informative inputs since they don't include zero padding. Comparatively, both experiments with windowed segments were nearly seven times faster than those with cycle segmentation.

In terms of evaluation results and metrics, windowed segments demonstrate better accuracy and score before the application of the voting system. On the other hand, cycle segmentation leads to a higher score after implementing the voting scheme. These findings highlight the trade-off between training speed and performance, where windowing techniques excel in speed and initial accuracy, while cycle segmentation combined with voting enhances the overall classification score.

4.4.2 Comparison between Feature Extraction techniques

In comparing the feature extraction techniques, it can be concluded from the aforementioned results that MFCCs offer a slightly faster training process. This can be attributed to their lower dimensionality and the absence of pooling layers in the model.

Overall, when considering accuracy and score, MFCCs outperformed spectrogram features when applied to windowed inputs. However, spectrogram features yielded the best results when used with cycle segments, surpassing the performance of all other experiments. These findings indicate that MFCCs are more suitable for windowed segments, while spectrograms are more effective for cycle segments.

4.4.3 Comparison with literature

When examining our results in comparison to previous works that employed similar settings and dataset, it appears that our approach using cycle segments and dynamic mel spectrogram features demonstrated promising performance, surpassing the average results. This can be observed in Table 12, where our approach achieved notable sensitivity and specificity values, resulting in a relatively reasonable score (MAcc) when compared to other approaches.

Model	Segmentation	Features	Results		
CNN-LSTM (ours)	Springer	Dynamic Mel Spectrogram	Sensitivity 95.45%	Specificity 90.70%	MAcc 93.08%
AdaBoost & CNN	Springer	Time and Frequency Features, MFCCs	Sensitivity 94.24%	Specificity 77.81%	MAcc 86.02%
DCNN	-	Chaogram	Sensitivity 84.49%	Specificity 91.63%	MAcc 88.06%
CRNN	-	Dynamic MFCCs	Sensitivity 98.66%	Specificity 98.01%	MAcc 98.34%

Table 12: Comparison of the obtained results with literature.

4.5 Summary

In the final chapter, we provided an overview of the performance metrics used to assess the model's performance. Subsequently, we analyzed and discussed the learning curves, confusion matrices, and evaluation results for each experiment separately. To consolidate the important findings, we presented a comprehensive summary table.

Furthermore, we delved into a comprehensive discussion of the outcomes of each experiment, comparing the two segmentation techniques and the two feature extraction techniques. This analysis allowed us to draw insightful conclusions regarding the effectiveness of different approaches.

In this project, our main objective was to conduct a comparative study between two audio segmentation techniques and two feature extraction techniques in classifying heart sound signals. Overall, all the experiments yielded satisfactory results in terms of evaluation metrics, although some approaches outperformed others in certain aspects.

The windowing techniques for segmentation demonstrated faster training and produced favorable outcomes. On the other hand, cycle segmentation exhibited slower training but still achieved good results, depending on the specific features used.

Regarding feature extraction, the utilization of the first and second order Mel Frequency Cepstrum Coefficients (MFCCs) showed superior performance when applied to windowed segments. Conversely, the Mel Spectrogram, along with its first and second order differences, proved to be more suitable for cycle segments.

Furthermore, the results indicated that Convolutional Neural Networks (CNNs) combined with Long Short-Term Memory (LSTM) networks are well-suited for audio classification tasks, supporting their effectiveness in capturing relevant patterns and features from the heart sound signals.

General Conclusion

In conclusion, this thesis aimed to detect cardiovascular diseases by classifying heart sounds using a combined CNN and LSTM model. Through a comparative study of two segmentation techniques (Windowed segments and Cycle segments) and two feature extraction techniques (MFCCs and Mel Spectrogram), valuable insights were gained regarding heart sound classification.

Our findings indicate that windowed segments offer faster training times and satisfactory results, while cycle segments, despite slower training, demonstrate exceptional evaluation metrics when combined with Mel Spectrogram feature extraction. This highlights the importance of selecting the appropriate segmentation technique based on the chosen features.

The implications of this research can be significant, as it aims to contribute to the development of advanced tools for predicting cardiovascular diseases. Future researchers can leverage these insights to enhance their work, improve efficiency, and ultimately create a final product with accurate predictions of cardiovascular diseases.

Several perspectives for future research in the field of heart sound classification can be considered; Exploring the application of new models, such as Transformers, holds promise for improving classification accuracy. Additionally, implementing robust data augmentation techniques, such as incorporating ambient background sounds or noise, can enhance generalization and adaptability of the model to real-world environments.

In summary, this thesis advances our understanding of detecting cardiovascular diseases through heart sound analysis. By evaluating segmentation techniques and feature extraction methods, effective approaches for heart sound classification have been identified. These research findings can potentially lay the groundwork for future advancements, leading to improved accuracy and efficiency in predicting cardiovascular diseases.

Bibliography

[1] World Health Organization, "Cardiovascular diseases (CVDs)," [Online]. Available: www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[2] Chengyu Liu et al, "An open access database for the evaluation of heart sound algorithms," 2016. [Online]. Available: www.dx.doi.org/10.1088/0967-3334/37/12/2181.

[3] Deng M et al, "Heart sound classification based on improved MFCC features and convolutional recurrent neural networks.," 2020. [Online]. Available: www.doi.org/10.1016/j.neunet.2020.06.015.

[4] Suyi Li et al, "A Review of Computer-Aided Heart Sound Detection Techniques," 2020. [Online]. Available: www.doi.org/10.1155/2020/5846191.

[5] Chen et al, "Deep Learning Methods for Heart Sounds Classification: A Systematic Review," 2021. [Online]. Available: www.doi.org/10.3390/e23060667.

[6] Cleveland Clinic, "Heart: Anatomy and Function," [Online]. Available: www.my.clevelandclinic.org/health/body/21704-heart.

[7] Shalom Education, "The Heart," [Online]. Available: www.shalom-education.com/courses/gcse-biology/lessons/transport-systems/topic/the-heart/.

[8] Cleveland Clinic, "Great Vessels of the Heart," [Online]. Available: www.my.clevelandclinic.org/health/articles/17057-your-heart--blood-vessels.

[9] Johns Hopkins Medicine, "Anatomy and Function of the Heart's Electrical System," [Online]. Available: www.hopkinsmedicine.org/health/conditions-and-diseases/anatomy-and-function-of-the-hearts-electrical-system.

[10] Cleveland Clinic, "Heart Conduction System (Cardiac Conduction)," [Online]. Available: www.my.clevelandclinic.org/health/body/21648-heart-conduction-system.

[11] World Health Organization, "Cardiovascular diseases," [Online]. Available: www.who.int/health-topics/cardiovascular-diseases.

[12] Centers for Disease Control and Prevention, "Heart Disease Facts," [Online]. Available: www.cdc.gov/heartdisease/facts.htm.

[13] Cleveland Clinic, "Cardiovascular Disease," [Online]. Available: www.my.clevelandclinic.org/health/diseases/21493-cardiovascular-disease.

[14] University Diagnostic Medical Imaging, P.C., "Cardiovascular Disease Risk," [Online]. Available: www.udmi.net/cardiovascular-disease-risk/.

[15] Centers for Disease Control and Prevention, "Coronary Artery Disease (CAD)," [Online]. Available: www.cdc.gov/heartdisease/coronary_ad.htm.

[16] Mayo Clinic, "Heart valve disease," [Online]. Available: www.mayoclinic.org/diseases-conditions/heart-valve-disease/symptoms-causes/syc-20353727.

[17] Mayo Clinic, "Heart failure," [Online]. Available: www.mayoclinic.org/diseases-conditions/heart-failure/symptoms-causes/syc-20373142.

[18] Mayo Clinic, "Cardiomyopathy," [Online]. Available: www.mayoclinic.org/diseases-conditions/cardiomyopathy/symptoms-causes/syc-20370709.

[19] Mayo Clinic, "Heart arrhythmia," [Online]. Available: www.mayoclinic.org/diseases-conditions/heart-arrhythmia/symptoms-causes/syc-20350668.

[20] Mayo Clinic, "Pericarditis," [Online]. Available: www.mayoclinic.org/diseases-conditions/pericarditis/symptoms-causes/syc-20352510.

[21] National Cancer Intitute, "noninvasive," [Online]. Available: www.cancer.gov/publications/dictionaries/cancer-terms/def/noninvasive.

[22] Mayo Clinic, "Heart disease Diagnosis," [Online]. Available: www.mayoclinic.org/diseases-conditions/heart-disease/diagnosis-treatment/drc-20353124.

[23] National Cancer Institute, "invasive procedure," [Online]. Available: www.cancer.gov/publications/dictionaries/cancer-terms/def/invasive-procedure.

[24] Healio, "Heart Sounds Topic Review," [Online]. Available: www.healio.com/cardiology/learn-the-heart/cardiology-review/topic-reviews/heart-sounds.

[25] Wikipedia, "Heart sounds," [Online]. Available: www.en.wikipedia.org/wiki/Heart_sounds.

[26] Samit Kumar Ghosh, Ponnalagu R N, "A Novel Algorithm based on Stockwell Transform," 2019. [Online]. Available: www.dx.doi.org/10.1109/INDICON47234.2019.9030299.

[27] Alila Medical Media, "Heart Sounds and Heart Murmurs, Animation.," [Online]. Available: www.youtube.com/watch?v=dBwr2GZCmQM.

[28] Sami Alrabie et al, "An Efficient Framework to Build Up Heart Sounds and Murmurs Datasets Used for Automatic Cardiovascular Diseases Classifications," 2021. [Online]. Available: www.doi.org/10.1007/978-981-33-6129-4_2.

[29] Wikipedia, "Heart murmur," [Online]. Available: www.en.wikipedia.org/wiki/Heart_murmur.

[30] Swash, Michael Glynn Michael, Hutchison's Clinical Methods 22e, 2007.

[31] Eko, "What Is a Phonocardiogram (PCG)," [Online]. Available: www.ekohealth.com/blogs/education/phonocardiogram-v1.

[32] R. S. Khandpur, Compendium of Biomedical Instrumentation, Volume 2, 2019.

[33] Wikipedia, "Artificial neural network," [Online]. Available: www.en.wikipedia.org/wiki/Artificial_neural_network.

[34] Jianwen Gan et al, "Underground Garage Patrol Based on Road Marking Recognition by Keras and Tensorflow," 2023. [Online]. Available: www.doi.org/10.3390/app13042385.

[35] Wikipedia, "Convolutional neural network," [Online]. Available: www.en.wikipedia.org/wiki/Convolutional_neural_network.

[36] S. Sumit, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," 2018. [Online]. Available: www.towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[37] Y. Muhamad, "Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail," 2019. [Online]. Available: www.dx.doi.org/10.1088/1742-6596/1201/1/012052.

[38] Wikipedia, "Recurrent neural network," [Online]. Available: www.en.wikipedia.org/wiki/Recurrent_neural_network.

[39] IBM, "What are recurrent neural networks?," [Online]. Available: www.ibm.com/topics/recurrent-neural-networks.

[40] K. Simeon, "How Recurrent Neural Networks work," 2017. [Online]. Available: www.towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7.

[41] Wikipedia, "Long short-term memory," [Online]. Available: www.en.wikipedia.org/wiki/Long_short-term_memory.

[42] Ryan T. J. J, "LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras," [Online]. Available: www.medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2.

[43] [Online]. Available: www.towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7.

[44] T. Adrian, "Training a PyTorch Model with DataLoader and Dataset," 2023. [Online]. Available: www.machinelearningmastery.com/training-a-pytorch-model-with-dataloader-and-dataset/.

[45] l. vikashraj, "Forward propagation in neural networks — Simplified math and code version," [Online]. Available: www.towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250.

[46] Shankar297, "Understanding Loss Function in Deep Learning," 2022. [Online]. Available: www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/#.

[47] Wikipedia, "Backpropagation," [Online]. Available: www.en.wikipedia.org/wiki/Backpropagation.

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, DEEP LEARNING, 2016.

[49] G. Sanderson, "Backpropagation calculus | Chapter 4, Deep learning," [Online]. Available: www.youtu.be/tIeHLnjs5U8.

[50] T. Gianluca, "Derivation of the Binary Cross Entropy Loss Gradient," 2021. [Online]. Available: www.python-unleashed.com/post/derivation-of-the-binary-cross-entropy-loss-gradient.

[51] Chauhan, Nagesh Singh, "Silver BlogOptimization Algorithms in Neural Networks," 2020. [Online]. Available: www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html.

[52] V. Bushaev, "Adam — latest trends in deep learning optimization.," 2018. [Online]. Available: www.towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.

[53] Kurtis Pykes, "The Vanishing/Exploding Gradient Problem in Deep Neural Networks," [Online]. Available: www.towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11.

[54] javaTpoint, "Issues in Machine Learning," [Online]. Available: www.javatpoint.com/issues-in-machine-learning.

[55] Ujwal Tewari, "Regularization — Understanding L1 and L2 regularization for Deep Learning," [Online]. Available: www.medium.com/analytics-vidhya/regularization-understanding-l1-and-l2-regularization-for-deep-learning-a7b9e4a409bf.

[56] Nitish Srivastava et al, "Dropout: A Simple Way to Prevent Neural Networks from," 2014. [Online]. Available: www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[57] Ketan Doshi, "Audio Deep Learning Made Simple: Sound Classification, Step-by-Step," [Online]. Available: www.towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5.

[58] C. Potes et al, "Ensemble of feature-based and deep learning-based classifiers for detection of abnormal heart sounds," 2016. [Online].

[59] Ren et al, "Deep Attention-based Representation Learning," 2021. [Online]. Available: www.arxiv.org/abs/2101.04979.

[60] Shahid Ismail et al, "PCG classification through spectrogram using transfer learning," 2023. [Online]. Available: www.doi.org/10.1016/j.bspc.2022.104075.

[61] Ketan Doshi, "Audio Deep Learning Made Simple (Part 2): Why Mel Spectrograms perform better," [Online]. Available: www.towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505.

[62] Leland Roberts, "Understanding the Mel Spectrogram," [Online]. Available: www.medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53.

[63] Mathworks, [Online]. Available: www.mathworks.com/help/dsp/ref/dsp.stft.html.

[64] Haytham Fayek, "Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between," [Online]. Available: www.haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

[65] S. Tanveer, "MFCC's Made Easy," [Online]. Available: www.medium.com/@tanveer9812/mfccs-made-easy-7ef383006040.

[66] Wikipedia, "Discrete cosine transform," [Online]. Available: www.en.wikipedia.org/wiki/Discrete_cosine_transform.

[67] Harimi et al, "Classification of Heart Sounds Using Chaogram Transform and Deep Convolutional Neural Network Transfer Learning," 2022. [Online]. Available: www.doi.org/10.3390/s22249569.

[68] Abbas et al, "Automatic Detection and Classification of Cardiovascular Disorders Using Phonocardiogram and Convolutional Vision Transformers," 2022. [Online]. Available: www.doi.org/10.3390/diagnostics12123109.

[69] Peter Bentley et al, "Classifying Heart Sounds Challenge," 2011. [Online]. Available: www.peterjbentley.com/heartchallenge/.

[70] G. D. Clifford et al, "Classification of normal/abnormal heart sound recordings: The PhysioNet/Computing in Cardiology Challenge 2016," 2016. [Online].

[71] Springer et al, "Logistic Regression-HSMM-based Heart Sound Segmentation," 2016. [Online].

[72] G. A. Fawzy, "Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall," [Online]. Available: www.blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/.

[73] S. Narkhede, "Understanding Confusion Matrix," 2018. [Online]. Available: www.towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

[74] Simon Fraser University, [Online]. Available: www.sfu.ca/sonic-studio-webdav/handbook/Mel.html.

[75] Dive into deep learning, [Online]. Available: www.d2l.ai/chapter_convolutional-neural-networks/conv-layer.html.

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

University M'Hamed
BOUGARA - Boumerdes

Institute of Electrical and
Electronic Engineering

Authorization for Final Year Project Defense

Academic year: 2022/2023

The undersigned supervisors: **Dr. E. BOUTELLAA**
authorizes the student:

Aymen RAHMANI Option: **Computer Engineering**

to defend his final year Master program project entitled:

Cardiovascular diseases detection from phonocardiograms using deep learning

during the ■ June ☐ ~~September~~ session.

Date: 13/06/2023

The Supervisor

Dr. E. BOUTELLAA

The Department Head