Phonocardiogram Signal Classification using Deep Learning Models

Eleni Kalaitzi

Bachelor's Thesis
Department of Physics
University of Crete

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Advisor: Prof. Yannis Stylianou
Supervisor: Dr George P. Kafentzis

October 12, 2023

Contents

Abstract

Heart disease is the leading cause of death in all continents except Africa. Heart diseases include coronary heart disease, heart failure, hypertensive heart disease, congenital heart disease, valvular heart disease, and structural heart abnormalities.

This thesis aims to improve the task of automatic risk detection of structural heart abnormalities from digital phonocardiogram (PCG) data with applications to pediatric heart disease screening. In a previous study, a number of convolutional neural network (CNN)-based systems operating on mel-frequency cepstral coefficients (MFCCs) of phonocardiogram signals are shown to perform better than systems using conventional machine learning guided by hand-crafted features. These systems were trained using time-frequency representations of segmental PCG frames. Various techniques for segmentation and time-frequency representations were designed to model the input recordings. Training and testing has been conducted on high-quality databases while each test produces five random experiments. Systems were evaluated using Receiver Operating Curve-Area under Curve, Sensitivity, Specificity, Accuracy, F1 score, and Matthews Correlation Coefficient.

In the context of this thesis, we emphasize on the improvement of pre-existing Convolutional Neural Network (CNN) models. We undertake this endeavor while adhering to the identical preprocessing methodology employed in a prior study. A Batch Normalization layer is included in pre-existing architectures, showing improvement in all performance metrics. Additionally, we introduce an enhanced CNN architecture combining Batch Normalization and an additional convolutional layer that outperforms previous models and presents significant improvement in all performance metrics.

Chapter 1

Introduction

Phonocardiography (PCG) is the term for the recording and analysis of acoustic vibrations made at the patient's chest using a microphone-transducer [1]. Analysis of phonocardiogram signals can be used to successfully study a number of heart disorders, including rheumatic valvular lesions, mitral and aortic regurgitation, and mitral and aortic stenosis murmurs. Clinicians listen to a patient's heart sounds to track cardiac tissue functionality, particularly the opening and closing of valves. Auscultation is the term for this type of evaluation of heart sounds aimed at making a diagnosis. It normally entails analyzing the time and frequency characteristics of heart sounds and murmurs, the general periodicity characteristics, and the quality of sounds. Heart disease is a serious health problem that has significant costs everywhere. Congenital heart disease (CHD), a major cause of pediatric morbidity and mortality, is a structural heart condition that can be present from birth and includes a variety of heart defects [2]. According to the severity of the underlying heart abnormality, up to 1% of newborn children are thought to have some kind of CHD [3]. A valuable asset in facilitating the screening of pediatric CHD is the utilization of automated heart sound classification technologies [4]. Employing an affordable, non-intrusive, and rapid screening approach would also enable the examination of a substantial population of individuals at a young age, leading to the timely detection of certain medical conditions. Due to recent strides in machine learning and computational capabilities, significant progress has been made in achieving levels of performance in various audio classification tasks, such as heart sound classification, that are approaching human-level accuracy [5]. Our research aims to enhance the effectiveness of automated technology for classifying heart sounds within the context of an ongoing study designed to identify and assess the risk of CHD during screening.

1.1 Biosignal Interaction: PCG, Mel-Spectroscopy, Signal Processing

The convergence of signal processing, mel-spectroscopy, phonocardiography (PCG), and biosignals has ushered in a paradigm shift in comprehending human physiology and detecting medical disorders in the dynamic environment of biomedical science and healthcare[6]. Biosignals, which include various electrical, mechanical, and chemical signals coming from within the body, provide a special window into the complex operation of physiological systems. These signals, however, are frequently hidden by noise, artifacts, and difficulties related to their capture. To unravel hidden patterns and glean priceless insights from biosignals, the combination of signal processing, mel-spectroscopy, and PCG is crucial in this situation.

Data collection, which includes recording electrocardiograms (ECG), electromyograms (EMG), electroencephalograms (EEG), and now phonocardiograms (PCG), is the first step in the process of biosignal processing [7]. However, these signals rarely arrive pure; they might be distorted by electrode placement, patient movement, or outside interference. During data preprocessing, signal processing methods such as mel-spectroscopy and PCG analysis take center stage. By converting complicated biosignals into visual representations and displaying their frequency content across time, mel-spectroscopy, which has its roots in speech and audio processing, can help medical experts handle biosignals. Mel-spectroscopy offers a distinctive viewpoint on how to analyze biosignals. It displays their frequency content across time through the transformation of those signals into visual and auditory representations. Researchers and physicians can discover temporal trends, fleeting abnormalities, and long-term patterns within biosignals because of this integration of auditory and visual dimensions. In addition, PCG adds a sound component to biosignal analysis by simultaneously recording the acoustic signature of cardiac events. The heart's symphony, composed of murmurs, rhythms, and beats, holds vital information about its health and functioning. PCG transforms these acoustic signals into interpretable data, allowing clinicians to listen to the heartbeat's nuances. The integration of PCG completes the convergence, adding an auditory dimension that enhances our grasp of biosignal intricacies.

Key elements with the potential for diagnosis and prognosis can be found within the intricate tapestry of biosignals. These characteristics, including as amplitude, frequency, duration, and others, are crucial indicators of physiological events. These traits are precisely retrieved with the use of PCG and mel-spectroscopy. Comprehensive spectrograms that map the frequency content and its changes over time are revealed this way. Contrarily, PCG converts cardiac events into recognizable sound patterns. By combining these tools, we can explore physiological dynamics in several dimensions and gain a deeper knowledge of biosignals.

Understanding the intricate patterns of biosignals is crucial because they contain a plethora of information. Signal processing, mel-spectroscopy, PCG analysis, and machine learning all work together in this situation. [8].Biosignals can be correctly characterized using algorithms for pattern recognition and classification. Machine learning algorithms can decipher complex patterns that could evade standard approaches using mel-spectroscopy-enhanced visuals and specific sound patterns acquired by PCG as vital inputs. For instance, the combination of PCG analysis and mel-spectroscopy enables the detection of heart murmurs, assisting in the diagnosis of cardiac diseases [9].

The foundation of contemporary healthcare is the real-time monitoring of biosignals. Imagine that ECG and PCG readings are used to continuously monitor a patient's cardiac health. Signal processing algorithms, led by mel-spectroscopy and PCG, can quickly identify anomalies, triggering immediate actions should an arrhythmia or aberrant sound occur. This combination gives medical professionals a toolkit for dynamic monitoring, enabling quick reactions to physiological changes.

Beyond clinical applications, cutting-edge research is fueled by the combination of signal processing, mel-spectroscopy, PCG analysis, and biosignals. With the aid of these methods, researchers can reveal the dynamics and underlying relationships that underlie both health and disease. Mel-spectroscopy and PCG analysis combined increase depths of understanding. For instance, using mel-spectroscopy on PCG recordings offers a special insight into fluctuations in heart sound, illuminating cardiac function that was previously obscure.

The integration of signal processing, mel-spectroscopy, PCG analysis, and biosignals is crucial in the complex mosaic of biomedical science. Their mutually beneficial partnership opens the door to accurate diagnoses, prompt solutions, and significant insights. By accurately revealing the richness of biosignals, which is frequently hidden by noise and complexity, this synergy enables physicians, researchers, and healthcare professionals to make well-informed judgments. A future in which human health is illuminated by the harmonic interaction of signals, spectroscopic transformations, audio signatures, and algorithms is promised as technology develops.

1.2 Automatic PCG classification

1.2.1 Basics of cardiac auscultation

Before creating an automated system to classify heart murmurs, it is necessary to have a fundamental understanding of how the heart produces sound. Heart sounds are acoustic signals that are produced inside the heart by blood flow and cardiac apparatus motion (mostly valve motion)[10]. These signals are then transported to the chest surface, where they can be heard directly with the ear or by utilizing a stethoscope. The study of murmurs or additional heart sounds, if present, as well as the temporal and frequency characteristics of the normal fundamental heart sounds (S1 and S2, which stand for input and outflow valve closure, respectively), constitutes proper cardiac auscultation. The key factors under investigation include the pitch, duration, location, and form characteristics of heart sounds and murmurs. Heart murmurs are clinically classified as innocent or abnormal primarily based on their temporal classification within the heart cycle, spatial classification based on punctum maximum, and finally due to a very subjective validation of murmur sound quality (with innocent murmurs frequently described as having ”musical” or ”vibratory” properties).

1.2.2 Automatic heart sound classification

In the area of automated phonocardiogram (PCG) classification, a number of tasks have been completed, and there is still need for improvement. The following are some tasks that have been completed and probable classification categories: Tasks which have been accomplished are:

- **Heart Sound Classification:**

  In order to categorize heart sound recordings into different groups, such as normal heart sounds, murmurs, gallops, and other aberrant sounds, algorithms have been devised. On this task, machine learning methods like neural networks have been used [5],[11].

- **Murmur Classification:**

  The detection and classification of murmurs, which are aberrant heart sounds brought on by turbulent blood flow, has received particular attention in the field of heart sound classification. Using machine learning algorithms, different murmur types (such as systolic and diastolic) and their intensity can be identified [11],[12],[13].

- **Valvular Disorder Identification:**

  Based on the patterns of heart sounds, certain algorithms attempt to diagnose particular valve illnesses (such as aortic stenosis and mitral regurgitation). The specific audio properties of these illnesses can be detected using cutting-edge signal processing and machine learning approaches [14],[15].

However, there is still some classification work that has not been completed.Some of them are the followings:

- **Severity Assessment:** Create algorithms to categorize conditions and determine their severity. For instance, based on the strength or features of heart sounds, one can distinguish between minor and severe valvular abnormalities.

- **Arrhythmia Detection:**

  Include in the automated analysis the identification and classification of arrhythmias (irregular heartbeats). Finding timing issues with heart sounds may be necessary.

- **Long-Term Monitoring:**

  Create algorithms that can analyze ongoing recordings of heartbeats for long- term surveillance, assisting in the identification of sporadic conditions or changes over time.

- **Clinical Decision Support:**

  Create systems that offer recommendations to clinicians rather than con- clusive diagnoses, assisting them in making more educated choices based on the automated analysis.

The requirement for high-quality labeled data, dealing with inter-patient variability, and making sure that automated algorithms support clinical skill rather than replace it are just a few of the obstacles that each of these jobs presents. Deep learning and explainable AI are two advancements in machine learning techniques that can considerably improve the success of these tasks [16],[17].

According to the specific classification goal, the quality of the data, and the complexity of the target cardiac diseases, the general phonocardiogram (PCG) classification using automated algorithms has had various degrees of success. Here are some broad conclusions regarding the effectiveness of PCG classification:

It has been pretty successful for algorithms to discern between normal and pathological cardiac sounds. This includes listening for any odd noises, such as heart murmurs. The complexity of the abnormalities and the unpredictability in the data, however, may have an impact on the accuracy. Some unusual or complicated disorders could be more challenging to appropriately classify. Particularly for frequent forms of murmurs, algorithms created expressly for classifying murmurs have reached respectable accuracy. It can be difficult, though, to distinguish between various murmur varieties, particularly delicate or uncommon ones. When addressing a wider variety of murmurs, accuracy could suffer. Based on distinctive auditory characteristics, several algorithms have been successful in recognizing specific valve problems, such as aortic stenosis or mitral regurgitation [18], [19].However, the quality and variety of the training data, as well as the complexity of the conditions classified, are key factors in this project's success. Also, convolutional and recurrent neural networks, in particular, have demonstrated encouraging results in PCG classification using deep learning models [20]. From the unprocessed audio data, they may automatically learn the necessary features. Deep learning techniques need a large amount of well-labeled data, and while their performance is often interpretable for medical applications, this is not always the case. Moreover, some PCG classification algorithms have been effectively included as decision support tools in clinical contexts, assisting medical practitioners in establishing precise diagnoses [21]. However, because of regulatory issues, the demand for ongoing validation, and the significance of retaining medical oversight, the adoption of these algorithms may differ. Generally, there are challenges related to the variability in heart sounds due to factors like age, sex, body type, and patient positioning poses challenges for accurate classification.

In conclusion, despite significant progress in automating PCG categorization jobs, there are still issues to be resolved. These algorithms' success is frequently context-dependent, and it can differ depending on the precise condition being addressed, the caliber and variety of the data, and the sophistication of the categorization techniques employed. To achieve accurate and therapeutically useful automated PCG categorization, cooperation between medical professionals, signal processing specialists, and machine learning practitioners is crucial.

1.2.3 Time-frequency features and neural networks for automatic heart sound classification

A time-frequency representation of a sound source is frequently used to refer to stacking (any representation of) spectra derived from windowed segments of the signal to construct a two-dimensional representation. The underlying assumption in employing time-frequency representation in audio classification is that some patterns exist within these image-like representations for specific classes of sounds. While there are numerous methods for constructing such representations, the most typical is to translate a linear representation such as Short-Time Fourier Transform (STFT) in a logarithmic scale to approximate human auditory responses such as Bark or Mel. Mel Frequency Cepstral Coefficients (MFCC) is also widely and efficiently employed in pathology detection research using automatic PCG categorization. As a time-frequency representation, wavelet-based features are used in a vast number of automatic PCG classification investigations.

1.3 Proposed Model

The existing study includes tests on two high quality data sets (PhysioNet-2016 and UOC-murmur; these are discussed in the following sections). From these tests it emerged that sub-band envelopes are preferable to other options in many settings using a CNN with a relatively simple architecture. Specifically, sub-band envelopes are the envelopes calculated for individual frequency bands. A CNN with two convolutional layers named M1SubEnv64by16eSyn1000 employs sub-band envelopes with time resolution of 64 and 16 frequency bands computed period synchronous 1 s frames as the feature, is the best system produced through tests on our data (UoC-murmur database). The system with the highest performance among the asynchronous systems has been evaluated on the PhysioNet-2016 challenge data after tests with the UoCmurmur database. With 0.845 sensitivity, 0.785 specificity, and 0.815 mean accuracy, it has been demonstrated that our asynchronous system performs similarly to the top-ranked state-of-the-art systems. At first, we tried to improve the performance of the already existing convolutional layers by adding a batch normalization layer with a momentum parameter of 0.999.Then we changed the number of filters in the uocseq2 model using a filter sequence 16 to 32 to 64. Since some important improvement was achieved from these tests we added a new model with three convolutional layers (presented in detail in the Paragraph 2.1.4) which provided the best results.

1.4 Outline

The organization of the chapters is as follows. Chapter 2 presents some general archives about CNNs and their operation.Then it describes the convolutional layers that have already been used for this study, as well as our own proposal for improving their operation and extending them. Chapter 3 presents the existing databases for this study, followed by a detailed description of the PhysioNet-2016 database. Chapter 4 reports on processing, the proposed architecture and the results obtained.Finally, chapter 5 presents the conclusions and future work.

Chapter 2

General principles of Convolutional Neural Network (CNN)

Artificial intelligence techniques called neural networks educate computers to interpret data in a manner that is modeled after the structure and operation of the human brain [22],[23]. More specifically, their function tries to combine the way the human brain works with abstract mathematical thinking. Thus, neural networks use ideas such as, for example, that a network learns and trains, remembers or forgets a numerical value, etc., things that have so far been attributed only to human thinking [24]. Of course, they can also make use of sophisticated mathematical operations and a wide range of mathematical analysis tools. Any type of neural network draws its influence from biology. Numerous neural networks with specialized functions make up the neurological systems of species. Each neural network consists of neurons, which are the fundamental units of the system. The tiniest autonomous unit in the network is the neuron. Neurons receive and transmit electrical impulses to other neurons in a constant and ceaseless process of information. Networks that mimic how the human brain operates are called Artificial neural networks (ANNs). The purpose of the neural network is to perform certain functions, such as image or sound recognition, after it has been trained with a significant amount of related data.

As a result, neural networks are made up of layers of connected nodes, or artificial neurons, that process different types of information and produce an output based on a set of inputs, or inputted data. Each network has a specific set of inputs and outputs (input-output). This is because they can be trained on vast amounts of data and model complicated correlations between input and output data, drawing conclusions and applying a variety of generalizations.

A basic neural network has interconnected artificial neurons at three levels [25]:

1. **Input Layer** The first level of nodes in an artificial neural network is referred to as the input layer. Artificial neurons make up the input layer, which is in charge of transforming the raw input data into a format that the network can comprehend.

2. **Hidden Layer** Hidden layer refers to a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function [26].It is possible to have more than one hidden layer in a neural network.

3. **Output Layer** The output layer in an artificial neural network is the last layer of neurons that produces given outputs for the program. Though they are made much like other artificial neurons in the neural network, output layer neurons may be built or observed in a different way, given that they are the last "actor" nodes on the network [27].

Usually we denote the input layer as vector X, the output layer as vector Y and the hidden layer, which is the "activation" node,as vector W.Each node is connected with each node from the next layer and each connection has particular weight. Weight can be considered as impact that that node has on the node from the following tier. So, if we were to look at just one node, it would appear as follows [28]:

All values are multiplied by their weight to give the value of the node. The node has a predefined activation function which determines whether the node is activated or even how active it is. We can also add a "bias" node. In particular, a bias value enables the activation function to be shifted to the left or right and aids in improving data fit (better prediction function as output).

Vector A is commonly used to represent hidden layers. The weights between the input and hidden layers is a 3x4 matrix,while the weight between the hidden and output layers is a 1x4 matrix.If network has a units in layer j and b units in layer $j+1$, then W_j will be of dimension $b{\times}(a+1)$ [28].

The "activation" nodes for the hidden layer should then be computed. The activation function $g(\cdot)$ must then be used after multiplying the input vector X and weights matrix w_1 for the top layer (Xw_1). The result

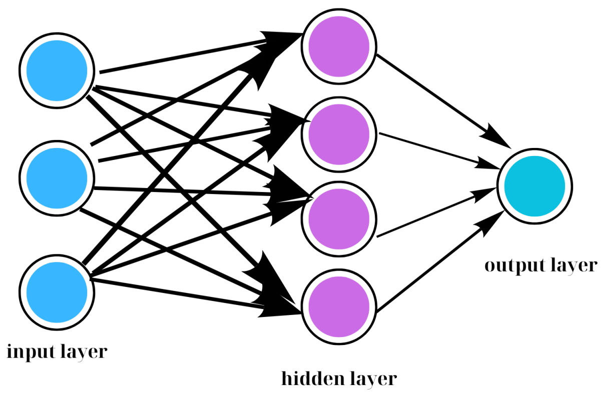


Figure 2.1: *An example of a Fully Connected Feed-forward Neural Network.*

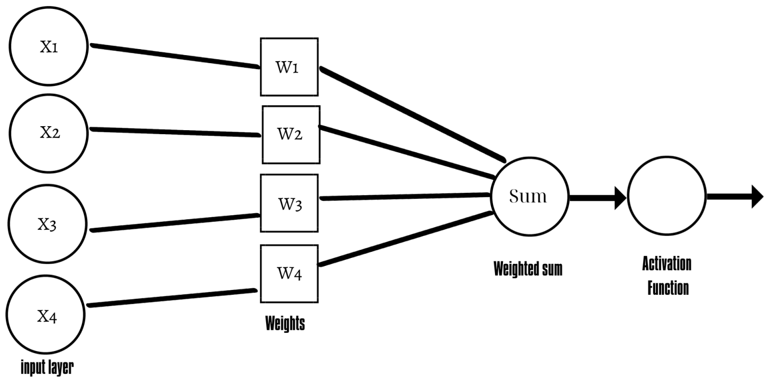


Figure 2.2: *An example of a Fully Connected Feed-forward Neural Network in terms of input variables and weights.*

is:

$$a_1^{(2)} = g\left(w_{10}^{(1)}x_0 + w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3\right) \tag{2.1}$$

$$a_2^{(2)} = g\left(w_{20}^{(1)}x_0 + w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3\right) \tag{2.2}$$

$$a_3^{(2)} = g\left(w_{30}^{(1)}x_0 + w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3\right) \tag{2.3}$$

Additionally, we obtain the output for the hypothesis function by multiplying the hidden layer vector by the weights w matrix for the second layer (Aw):

$$h_w(x) = a_1^{(3)} = g\left(w_{10}^{(2)}a_0^{(2} + w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)}\right) \tag{2.4}$$

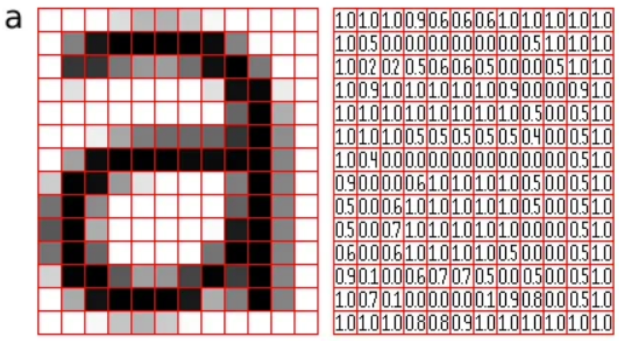
In this example, there is just one hidden layer and four nodes. The following formula would be obtained if we were to generalize for neural networks with multiple hidden layers and numerous nodes in each layer.

$$a_n^L = \left[\sigma\left(\sum_m w_{nm}^L \left[\cdots\left[\sigma\left(\sum_j w_{kj}^2\left[\sigma\left(\sum_i w_{ji}^1 x_i + b_j^1\right)\right] + b_k^2\right)\right]\cdots\right]_m + b_n^L\right)\right]_n \tag{2.5}$$

where we have L layers with n nodes and $L-1$ layers with m nodes.

2.1 Convolutional Neural Network

A neural network type called a convolutional neural network, or CNN or ConvNet, is particularly adept at processing input with a grid-like architecture, like an image. A binary representation of visual data is a digital image. It is made up of a grid-like arrangement of pixels, each of which has a pixel value to indicate how bright and what color it should be [29], as in Figure 2.3.



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

Figure 2.3: *Representation of image as a grid of pixels [29]*

Our brain, when we see an image, coordinates all its neurons (each operating in its own receptive field) in such a way that the image is captured as best as possible. So, CNNs are constructed in such a way to mimic the human brain. Each neuron in the CNN processes a specific piece of information. The CNN layers are arranged in such a way that they first detect simpler patterns and then combine them to create more complex ones until they finally identify the intended object. Node layers, which include an input layer, one or more hidden layers, and an output layer, make up CNN. Each node has a threshold and weight that are connected to one another. Any node whose output exceeds the defined threshold value is activated and begins providing data to the network's uppermost layer. Otherwise, no data is transmitted to the network's next layer [30]. The first layer in a convolutional network is the convolutional layer. It can be followed by other convolutional layers and pooling layers, whose number depends on the particular work. The final layer is a fully-connected layer. These three types of layer are described in detail below.

2.1.1 Convolution Layer

In convolutional neural networks, the majority of computations occur.This layer creates a dot product between two matrices, one of which is the kernel—a collection of learnable parameters—and the other of which is the constrained area of the receptive field. Assume that the input will be a color image that is composed of a 3D pixel matrix. Thus, the input will have three dimensions: height, width, and depth, which are equivalent to RGB in an image. Compared to a picture, the kernel is smaller in space but deeper. Since the depth goes up to all three channels, the kernel height and breadth will be spatially small. In order to determine whether the feature is there, the kernel will traverse over the image's receptive fields. Convolution describes this process. As for the feature detector, it is a 2D array of weights, usually represented as a 3x3 table. This size also determines the receptive field. Following the application of the filter to an area of the image, the dot product between the input pixels and the filter is determined as in Figure 2.4. The output array is then fed with this dot product.

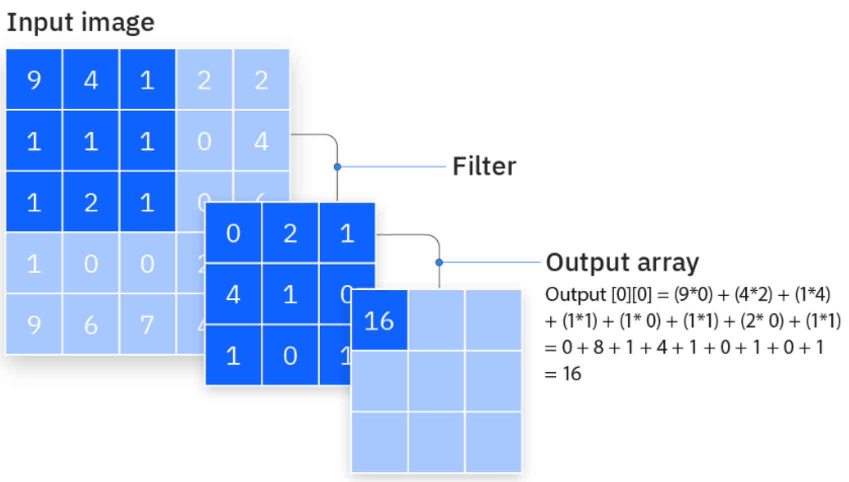


Figure 2.4: *Feature Map [30]*

Once the kernel has swept through the entire image, the filter shifts by a stride and repeats the operation. A feature map, activation map, or convolved feature is the ultimate result of the series of dot products from the input and the filter. These include:

- **Number of filters**: The number of filters describes how many distinct convolutional kernels were used to transform the input data. A feature map or an activation map is the result of element-wise multiplications and additions performed by each filter, which is a tiny matrix that slides over the input data. The variety of patterns or features that a convolutional layer may learn from the input data depends on how many filters are included in the layer [31].

- **Stride**: The kernel's stride is how many pixels or how far it travels across the input matrix. Despite the rarity of stride values of two or higher, a longer stride results in a lesser output [30].

- **Padding**: Padding is the technique of increasing the input image's pixel count before applying convolution. These extra pixels serve as a barrier around the image, preserving more spatial information for the network. Padding is typically applied uniformly to all sides of the input image. Zero padding in CNN is the most popular technique, where you add a lot of zeros all around your input feature map. The CNN architecture's padding feature allows you to keep the feature maps' spatial dimensions. By extending the input's borders, it makes sure that every pixel is treated equally during convolution. Padding is also important for figuring out how big each layer's output should be in the CNN. If not, each convolution will gradually diminish the feature maps' spatial dimensions, which will result in a considerable downsampling. Padding allows us to regulate the output size of each layer. The degree of downsampling depends on how much padding is used [32]. There are three types of padding:

  1. Valid padding: Also known as no padding.In this situation, if dimensions do not line up, the final convolution is dropped.
  2. Same padding:the output has the same size as input layer
  3. Full padding:This kind of padding enlarges the output by padding the input border with zeros.

  For determine the size of the output volume, the following equation is used:

$$\text{output size} = \frac{W - F + 2P}{S} + 1 \tag{2.6}$$

  where W is the input size (width and height), F is the spatial size of the convolutional filter (or kernel), S is the stride and P is the amount of padding.

2.1.2 Pooling Layer

Downsampling, sometimes referred to as pooling layers, carries out dimensionality reduction and lowers the amount of parameters in the input. The pooling operation sweeps a filter across the entire input similarly to the convolutional layer, with the exception that this filter lacks weights. Instead, the kernel populates the output array by applying an aggregation function to the values in the receptive field. There are principally two forms of pooling:

- **Max pooling**: This operates by choosing the highest value possible from each pool. The most noticeable aspects of the feature map are retained by Max Pooling, and the resulting image is crisper than the original [33].

- **Average pooling**: The average values of the features on the feature map are preserved through average pooling. While maintaining the essential qualities of the feature, it blurs the image [33].

The pooling layer loses a lot of information, but it also offers the CNN a number of advantages. They lessen complexity, increase effectiveness, and lower the risk of overfitting. An example of a pooling operator for both average and max pooling is shown in Figure 2.5.

2.1.3 Fully-Connected Layer (FC)

As was already noted, partially connected layers do not have a direct connection between the input image's pixel values and the output layer. In contrast, every node in the output layer of the fully-connected layer is directly connected to a node in the layer above it.Based on the features that were retrieved from the preceding layers and their various filters, this layer conducts the classification operation. FC layers often utilize a softmax activation function to categorize inputs appropriately, producing a probability ranging from 0 to 1. Convolutional and pooling layers typically use ReLU functions. The activation functions are described in detail below.

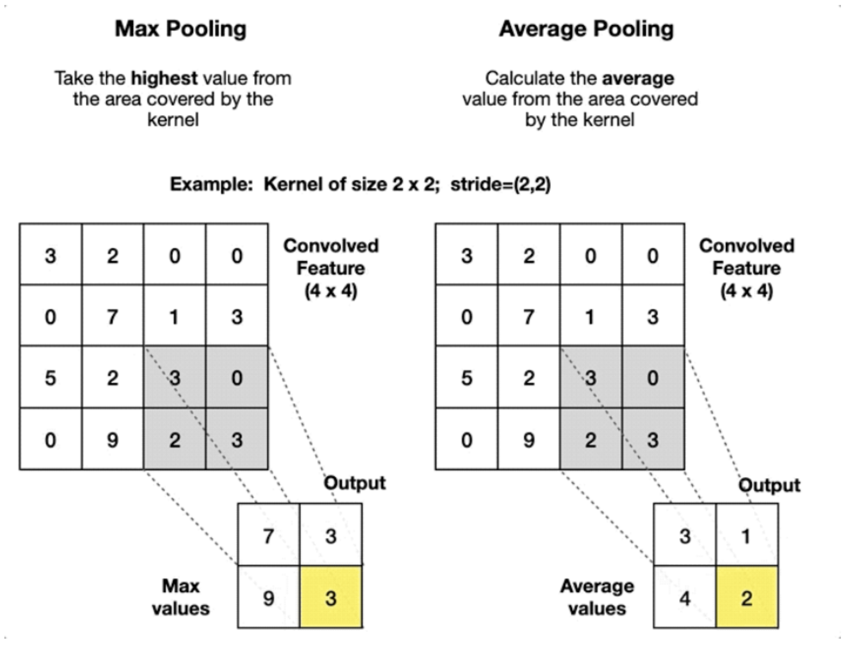

Max Pooling
Take the highest value from the area covered by the kernel
Average Pooling
Calculate the average value from the area covered by the kernel
Example: Kernel of size 2 x 2; stride=(2,2)
Convolved Feature (4 x 4)
3 2 0 0
0 7 1 3
5 2 3 0
0 9 2 3
Output
7 3
9 3
Max values
3 1
4 2
Average values

Figure 2.5: *Max Pooling and Average Pooling [34]*

2.2 Hyperparameters

Hyperparameters are parameters whose values are set before starting the model training process [35]. These hyperparameter values govern the training algorithm's behavior and how it learns the parameters from the data.

2.3 Learning Rate

It is in charge of the basic learning feature and must be chosen in such a manner that it is not too high, preventing us from converge to minima, and not too low, preventing us from speeding up the learning process. It is advised to experiment with powers of ten. The value of the learning rate is heavily influenced by the optimizer utilized.

2.4 Batch Size

It is an indication of how many patterns were presented to the network prior to an update to the weight matrix. Less repeated patterns would result in more erratic weights and problematic convergence if the batch size were smaller. Learning would be delayed if batch sizes were large because the batch size wouldn't vary for a while.

2.5 Number of Epochs

The number of epochs refers to how frequently the model is presented with the whole training set.

2.6 Activation Functions

Based on the weighted total, the activation function in a neural network determines whether a specific node should be "activated" or not. This weighted total value will be defined as z. Some of the commonly used activation functions in CNNs are:

2.6.1 Sigmoid Function

It is not a linear function and ideal for binary classification problems. Moreover, it allows the nodes to have range values between $(0, 1)$. Different probability of "activation" for each output class would result in the case of numerous output classes. The option with the highest "activation" (probability) value will be picked. It is represented as:

$$S(x) = \frac{e^x}{e^x + 1} \tag{2.7}$$

2.6.2 Tanh Function (Hyperbolic Tangent)

Similar with the sigmoid function,but with different output range (-1,1).It is a zero centered function,so it will be easy to map the output values. It can be represented as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

2.6.3 ReLU (Rectified Linear Unit)

It is the most used activation function [36]. All negative numbers are reset to zero, while positive values are left unaltered. ReLU tackles the vanishing gradient issue and aids in training convergence more quickly. Not every neuron is activated simultaneously by the ReLU function. Only if the output of the linear transformation is less than 0 will the neurons become inactive.The ReLU function is far more computationally efficient than the sigmoid and tanh functions since it only activates a subset of neurons. Due to ReLU's linear, non-saturating nature, gradient descent's convergence towards the loss function's global minimum is sped up. Mathematically,

$$\text{ReLu}(x) = \max(0, x) = \frac{x + |x|}{2} \tag{2.9}$$

2.7 SoftMax

A vector of K real numbers is transformed into a probability distribution of K potential outcomes using the softmax function, also known as softargmax or the normalized exponential function. It is applied in multinomial logistic regression and is a generalization of the logistic function to many dimensions. Furthermore,the softmax function normalizes a vector z of K real numbers into a probability distribution with K probabilities proportional to the exponentials of the input values. Thus,it accepts this vector as an input. In other words, some vector components might not sum to 1 or be negative before using softmax, but each component will be in the interval after applying softmax.They can be regarded as probabilities because the components are $(0, 1)$ and sum up to 1. In simple words, it applies the standard exponential function to each element z_i of the input vector z and normalizes these values by dividing by the sum of all these exponentials; this normalization ensures that the sum of the components of the output vector $\sigma(z)$ is one [37]. The mathematical representation of softmax is:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{2.10}$$

where $i = 1, \cdots, K$ and $z = z_1, ..z_k$ are real numbers.

2.8 Optimization

In order to change the attributes of the neural network, algorithms or methods are used,which are called **optimizers**.Their aim is to minimize the losses.Loss in deep learning,inform us about the performance of the model. Therefore,by minimizing losses we achieve better model performance as well as faster training model.This process is called **optimization**.

It's impossible to predict your model's weights from the beginning. You can ultimately get there, though, with a little bit of trial and error based on the loss-cost function.The optimizers you employ determine how you should modify the weights or learning rates of your neural network to minimize losses.Below,there are different types of optimizers with their own benefits and drawbacks.

2.9 Gradient Descent (GD)

An optimization method called gradient descent can be used to locate the local minimum of a differentiable function.[38] Gradient descent is simply used to identify the parameters (coefficients) of a function that minimize a cost function as much as is practical.

Starting by initializing the parameter's value,gradient descent adjust the values in such a way as to minimize the cost function.The way is adjusted in accordance with the update equation after being initialized using a few initialization procedures.The gradient of the cost function $J(\theta)$ computes with this repeated equation until convergence:

$$w_j \leftarrow w_j - a\frac{\partial}{\partial w_j}J(w) \tag{2.11}$$

The aim is to reach the local minimum.To reach this local minimum,it should be set the learning rate to an appropriate value. If this rate is big enough,may not be possible to it may not be possible to achieve the desired outcome. While if this rate is too small,It will take some time to get the minimum.

The advantages of this technique is that it easy to compute and to implement.On the other hand, may trap at the nearby minimum. Also, after computing the gradient on the entire dataset, weights are modified. So,it could take years for this to converge to the minima if the dataset is too vast. Therefore,to calculate the gradient across the entire dataset, a lot of RAM is needed.

2.9.1 Stochastic Gradient Descent (SGD)

SGD is an improved algorithm which is computed taking over one point at a time.[38] It can be thought of as a stochastic approximation of gradient descent optimization since it substitutes an estimated gradient for the true gradient, which is generated from the whole set of data. This lessens the extremely high computational burden, particularly in high-dimensional optimization problems, allowing for faster iterations at the cost of a reduced convergence rate. Parameter of the SGD performance is:

$$w = w - a\frac{\partial J(w; x(i), y(i))}{\partial w} \tag{2.12}$$

where $x(i)$, $y(i)$ are the training examples. So, less memory is used than with the GD technique, because the derivative is computed using only one point at a time. However, it may also stuck at local minima and might take a long time to converge.

2.9.2 Mini Batch Stochastic Gradient Descent (MB-SGD)

The MB-SGD algorithm is an extension of the SGD algorithm that solves the SGD algorithm's issue with high time complexity.[38] A group or subset of points from the dataset are used by the MB-SGD algorithm to generate the derivate. After a certain number of iterations, it is noted that the derivative of the loss function for MB-SGD is almost identical to a derivative of the loss function for GD. However, MB-SGD requires more repetitions than GD does to reach minima, and the cost of computing is also higher.The derivative of loss for a group of points determines how the weight is updated. Because the derivative is not always heading towards minima, the updates in the case of MB-SGD are very noisy. The parameters are updated:

$$w = w - a\frac{\partial(J(w; B(i)))}{\partial w} \tag{2.13}$$

The benefit of this method is takes less time complexity to converge compared to standard SGD algorithm [9]. However, when compared to the update of the GD algorithm, the MB-SGD update is significantly noisier. Also,converge more slowly than the GD algorithm and may get stuck at local minima.

2.9.3 SGD with momentum

The MB-SGD method has a significant disadvantage in that weight changes are quite noisy.[38] By denoising the gradients, SGD with momentum eliminates this issue. Weight updates are dependent on noisy derivatives, and if the derivatives can be denoised , convergence time will be reduced.The objective is to denoise the derivative using an exponential weighting average, which gives more weight to recent updates than to older ones.It promotes convergence in the relevant direction while decreasing fluctuation in the irrelevant direction. This approach employs another hyperparameter called as momentum, denoted by $'\gamma'$.

The weights are updated by:

$$w = w - v(t) \tag{2.14}$$

where

$$v(t) = \gamma v(t-1) + \alpha\frac{\partial(J(w))}{\partial w} \tag{2.15}$$

Usually,the term γ is set to 0.9. Momentum at time 't' is calculated using all prior updates, with current updates receiving more weight than previous updates. This accelerates the convergence.

However, for each update, one more variable is need to be computed.

2.9.4 Nesterov Accelerated Gradient (NAG)

The NAG algorithm's concept is extremely similar to SGD with momentum, with a little variation.[38] The momentum and gradient are computed on the last updated weight in the case of SGD with a momentum method. Momentum is a decent strategy, but if it is too strong, the algorithm may miss the local minima and continue to rise. As a result, the NAG algorithm was created to address this issue. It is a strategy of planning ahead of time. We know we'll be adjusting the weights with $\gamma.v(t-1)$, thus $w - \gamma v(t-1)$ approximates the future position. We'll now compute the cost using this future parameter rather than the current one.

$$v(t) = \gamma v(t-1) + \alpha \frac{\partial(J(w - \gamma v(t-1)))}{\partial w} \tag{2.16}$$

while the parameters are updated by:

$$w = w - v(t) \tag{2.17}$$

Again,the parameter γ has a value of around 0.9.

2.9.5 Adaptive Moment Estimation (Adam)

To speed up the optimization process of a typical optimizer in a CNN model, an Adam optimizer with a power-exponential learning rate is used to train the CNN model, with the power-exponential learning rate controlling the iteration direction and step size [38],[39].The power-exponential learning rate can be adaptively modified based on the previous stage's learning rate and the gradient relationship between the previous stage and the present stage. To meet the requirements of adaptive adjustment, the prior gradient value is used to alter the correction factor.This helps to adjust the learning rate in a narrow range, the parameters of each iteration are relatively stable, and the learning step is chosen based on the appropriate gradient value to change the convergence performance of the network model and ensure the network model's stability and effectiveness.

For each parameter, Adam computes adaptive learning rates. Adam, like momentum, preserves an exponentially decaying average of previous squared gradients v_t in addition to an exponentially decaying average of past gradients m_t.

The exponential decay rates of these moving averages are controlled by the hyper-parameters $\beta_1, \beta_2, \epsilon[0, 1]$. The decaying averages of past and past squared gradients, m_t and v_t, are computed as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.18}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.19}$$

where m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively [38].

2.10 Forward Propagation

Forward propagation is the process through which input data is fed forward through a network to generate an output. The data is accepted by hidden layers and processed according to the activation function before being passed to the next layer. The forward flow of data is intended to avoid data travelling in a circular motion, which produces no output. Pre-activation and activation occur at each hidden and output layer node of a neural network during forward propagation. The weighted total is calculated via the pre-activation function. Based on the weighted sum, the activation function is used to make the neural network flow non-linearly employing bias.

However, a neural network relies on both forward and backward propagation to be taught.

2.11 Backpropagation

Backpropagation is a popular method for training feedforward neural networks. It determines the gradient of the loss function in relation to the network weights. It is far more efficient than explicitly computing the gradient for each weight. Because of this efficiency, gradient methods can be used to train multi-layer networks and update weights to minimize loss; variations like as gradient descent or stochastic gradient descent are frequently utilized. To prevent unnecessary computation of intermediate terms in the chain rule, the backpropagation algorithm computes the gradient of the loss function with respect to each weight using the chain rule, layer by layer, and iterating backward from the last layer.

2.12 Overfitting

Overfitting occurs when a model accurately models the training data. Precisely, overfitting occurs when a model learns the information and noise in the training data to the point where it severely impairs the model's performance on new data. This means that the model picks up on noise or random fluctuations in the training data and learns them as ideas. The issue is that these notions do not apply to new data, limiting the models' ability to generalize. Testing machine learning models on more data with a thorough representation of possible input data values and kinds is the best way to find overfit models. A portion of the training data is typically utilized as test data to check for overfitting. Overfitting is indicated by a high error rate in the testing data.One approach of testing overfitting is **Cross validation**.

2.13 Cross Validation

The training set is divided into k equally sized subsets or sample sets termed folds in this method.A sequence of iterations comprise the training process. The following steps are performed during each iteration:

- Use one subset for validation and train the machine learning model on the remaining K-1 subsets.
- Examine the model's performance on the validation sample.
- Evaluate model performance based on the quality of the generated data.

2.14 Dropout

It is possible to think of the keep-probability of the Dropout layer as a hyper-parameter that may function as a regularizer to direct our search for the ideal bias-variance point. It achieves this by cutting off some connections after each iteration so that the hidden units cannot heavily rely on any one feature. The range of possible values is 0 to 1, and it only depends on how much the model is overfitting.

2.15 L1/L2 Regularization

This regularizer reduces extremely high weight values to make the model less reliant on a single feature. This typically decreases variation at the expense of raising bias, or diminishing accuracy. Useful when the model remains over-fit despite a markedly increased Dropout value.

2.16 Performance Metrics

In order to assess the performance of the system and ascertain whether any improvements have been realized, a set of specific metrics were employed. These metrics include:

2.16.1 Accuracy

Particularly in binary or multiclass classification problems, accuracy is a performance parameter used to evaluate how accurately predictions made by a classification model as a whole are. It measures the percentage of accurately predicted instances among all of the dataset's instances. Accuracy is calculated by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.20}$$

where

- True Positives (TP): The percentage of occurrences that were accurately categorized as positive
- True Negatives (TN): The number of occurrences that were accurately labeled as negative
- False Positives (FP): The number of instances that were mislabeled as positive when they actually fall within the negative class.
- False Negatives (FN): The number of instances that should have been rated positive but were instead labeled negatively.

The ratio of the total number of instances to the sum of true positives and true negatives is known as accuracy. It displays the overall percentage of accurate predictions made across all classes by the model. Low accuracy signifies that the model is producing more incorrect predictions, whereas high accuracy shows a high percentage of correctly identified occurrences. Although accuracy is a frequently used data to assess classification models, it may not be the best one in situations where datasets are unbalanced and one class considerably outnumbers the other.Other metrics, like as precision, sensitivity, F1-Score, or area under the ROC curve (AUC-ROC), may offer a more thorough evaluation of model performance in such circumstances.

2.16.2 Sensitivity

Sensitivity is a classification metric that assesses how well a model can detect all pertinent positive instances.Sensitivity is especially significant when it comes to preventing false negatives. The sensitivity equation is:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2.21}$$

Sensitivity is defined as the proportion of true positives to all positive occurrences. It is helpful in circumstances where omitting positive cases can have important repercussions because it measures how well the model performs in accurately capturing all positive incidents.

2.16.3 Specificity

Specificity evaluates a model's capacity to accurately detect instances of the negative class, or examples that don't fall under the focus class.When assessing a model's capacity to minimize false alarms or false positive errors, specificity is very crucial.Specificity is calculated by:

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.22}$$

Specificity is typically expressed as a value between 0 and 1, where higher values indicate a better ability to correctly identify negative instances. Specificity is particularly useful in scenarios where false positives can have significant consequences or where it's essential to minimize incorrect alarms.

2.16.4 Precision

Precision,also known as positive predictive value,quantifies the accuracy of positive predictions made by a model, indicating how many of the predicted positive instances were actually correct.To compute precision:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.23}$$

In most cases, precision is expressed as a number between 0 and 1, with larger numbers indicating a greater capacity for precise positive prediction.In situations when false positives have major implications or costs, precision is especially important.

2.16.5 F1 Score

F1 score provides balance between precision and sensitivity.When classes in the dataset are distributed unevenly or when false positives and false negatives have varied costs, the F1 score is particularly helpful. It must first be calculated the precision and recall before combining them with the harmonic mean to get the F1 score.

$$\text{F1 score} = \frac{2(\text{Precision} \cdot \text{Sensitivity})}{\text{Precision} + \text{Sensitivity}} \tag{2.24}$$

It provides an accurate assessment of the model's performance by combining the two criteria into one value. The F1 score penalizes models with a large discrepancy in precision and sensitivity because the harmonic mean gives more weight to lower values. It ranges between 0 and 1, where a higher score indicates better overall performance. It's particularly useful in situations where false positives and false negatives have different implications, as it helps find a compromise between precision and sensitivity.

2.16.6 Matthews Correlation Coefficient (MCC)

The Matthews Correlation Coefficient (MCC), also known as the phi coefficient, takes into account true positives, true negatives, false positives, and false negatives to provide a balanced measure of a classification model's performance.The MCC value ranges from -1 to $+1$, where $+1$ indicates perfect prediction, 0 indicates random prediction, and −1 indicates total disagreement between the model's predictions and the actual outcomes. The calculation is made in the following way:

$$\text{MCC} = \frac{TP \cdot TN + FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.25}$$

The MCC provides a more robust evaluation of a model's performance than accuracy.

2.16.7 ROC Curve

An illustration of a binary classification model's performance at various discriminating thresholds or decision limits is called a Receiver Operating Characteristic (ROC) curve. It is a crucial tool for evaluating the trade-off between the model's true positive rate and its false positive rate as the threshold for identifying occurrences as positive or negative is changed.On the vertical axis,the ROC curve plots the true positive rate(sensitivity) while on the horizontal axis,ROC curve plots the false positive rate(specificity) which is equal to 1. The ROC curve provides a visual representation of how well a model can distinguish between positive and negative instances across various threshold values. The ideal ROC curve hugs the top-left corner of the plot, indicating high sensitivity (few false negatives) and low false positives. In contrast, a random classifier's ROC curve would be a diagonal line from the bottom-left corner to the top-right corner, indicating no discriminatory power. The Area Under the ROC Curve (AUC-ROC) can be used to summarize a binary classification model's performance. A higher AUC value indicates better discriminatory power, and the AUC-ROC measures the model's capacity to distinguish between positive and negative examples. While an AUC of 1.0 indicates flawless discrimination, one of 0.5 suggests random performance.

2.17 Convolutional Neural Network (CNN) in our study

The CNN-based classifiers are designed to test various settings of common time-frequency representation used directly as the input to a CNN classifier. So, it is presented here the various features, various options for segmentation and CNN's architectures.

2.17.1 Feature Extraction

Both the entire signal level and the frame/segment level, where numerous frames are extracted via windowing, can be used for feature extraction. Frame level decisions are fused in order to categorize specific files. A signal must be automatically segmented into frames in order to extract features at the frame level. Both period synchronous and asynchronous segmentation techniques can be used, and we test both of them in our comparisons. Period marking is required for period synchronous segmentation.

2.17.2 Period Marking And Segmentation

Period marking can be carried directly on PCG signals (segmentation into heart cycles or marking of cycle starting occurrences).Since ECG signals are less noisy and contain a major peak that can be monitored for accurate period marking,automatic marking can be done more consistently when ECG signals are captured in parallel. So, ECG signals are recorded simultaneously together with PCG signals. So by following the following procedure we managed to extract period marks from the ECG signals. At first, ECG signal was filtered by High-pass filter, where its purpose is to remove the low frequencies.Then,energy signal was computed and there was an amplitude normalization of this signal and an autocorrelation based period detection from energy signal.Then,the number of heart cycles in the signal was estimated.Afterwards the peaks were detected by applying a threshold of 0.5 and finally the spurious peaks were removed using amplitude comparison and distance to surrounding peaks.

Once the period markings are available, various ways for segmentation can be utilized to retrieve PCG frames. The strategies used are as follows :

- Period synchronous segmentation with segment length defined proportionally to the local period (half period, one period, two periods, etc). Overlap is inevitable for segment lengths longer than one period.
- Period synchronous segmentation with fixed segment length (0.5s, 1s, 2s etc.)[2]. Overlap occurs whenever the segment length exceeds the period length.
- Period asynchronous segmentation with or without overlap and fixed segment length.

2.17.3 Computing time-frequency features

When considering automatic audio/sound categorization systems with CNN architectures, the time-frequency representations listed below were chosen to be among the most frequent:

- Spectogram
- MFCC (with or without delta coefficients)
- Mel-Spectogram (with or without delta coefficients)

We added sub-band envelopes as a time-frequency representation to these standard representations. Also,sub-band envelopes of a given PCG signal are created by stacking temporal modulation envelopes of sub-bands obtained by band-pass filtering the PCG signal. Various time and frequency resolutions were tested for all functionalities. Just before the computation, a Tukey window ($r = 0.08$) is added to signal segments. Where Tukey window is a cosine lobe of width $\frac{Na}{2}$ that is convolved with a rectangular window of width $N(1 - \frac{a}{2})$

2.17.4 Sub-band envelopes as a time-frequency feature

Envelope signals are used in segmentation tasks and also as features fed into neural network classifiers.
For computing sub-band envelopes of a PCG segment,the following steps are followed:

- Using Gammatone filter banks to apply band-pass filtering to PCG signals.
- Accomplish envelope detection by computing an analytical signal using the Hilbert transform.
- Resampling of envelopes to a specific time-resolution.
- Using logarithmic algorithm to compress the final envelope signals.
- Generate an image-like time-frequency representation,all envelopes are layered.
- The resultant matrix is processed to have a zero-mean and normalized amplitude.

In Fig. 2.6, it is represented the process flow diagram and an example of feature extraction, with the computed sub-band signal envelopes and final feature obtained in matrix form. The top sub-plots show 8 sub-band signals as well as their resampled equivalents taken from the original PCG signal (in blue). In this case, after stacking 8 vectors (corresponding to 8 sub-bands) of size 128 (number of time bins), an 8×128 image-like representation is derived and plotted with color coding element values, yielding the bottom sub-plot, which is the main feature used as input to the classifier.

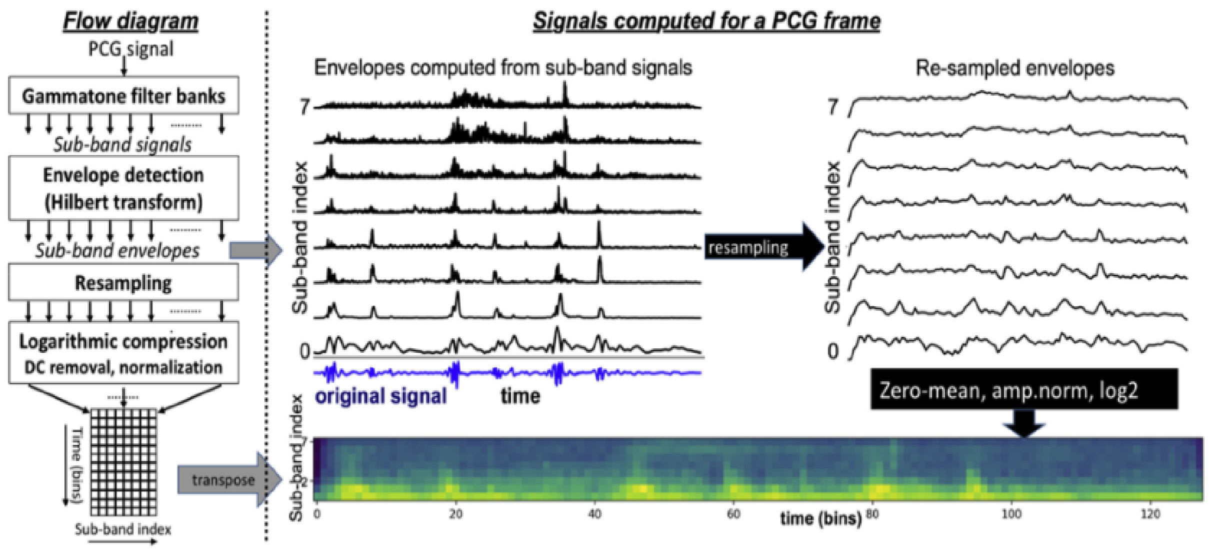


Figure 2.6: *Sub-band envelopes feature computation. Original PCG signal depicted in blue. Sub-band envelope feature is the matrix obtained at the output of the feature extraction process which is depicted as a colored image via mapping matrix coefficients to color code (low values: dark, high values: bright) (image from [2])*

2.18 Machine Learning

For the PCG classification task, a wide range of neural network models can be used. The tests are restricted to the usage of feed-forward CNN models. To keep the number of tests limited , three similar models with common sequence of layers are used : 2D convolutional layers (with kernel size 3 by 3, rectified linear unit activation) followed by max-pooling and drop-out layers.The input dimension is the same as the feature dimension, and the output dimension has two categories (normal and abnormal). Because PCG database sizes are frequently small (in comparison to other automatic classification tasks using CNNs), complicated models with high capacity learn to memorize the train data. As a result, the number of layers was maintained to a minimum, and L1-regularisation was used to avoid over-fitting. The three models have 1, 2, and 4 2D convolutional layers, respectively. Each model is intended to compute the probability of a segment belonging to a recording of a patient with pathology. To compute the probability of a file/patient belonging to a problematic class, all frame probabilities are sorted, the 15% lowest and 15% highest values are removed, and the probability for a file is computed as the mean probability of these remaining frames.

2.18.1 Model With Single Convolutional Layer

More specifically, the first model is a single convolutional layer. It is a sequential model where the output of one layer is passed as input to the next layer. In this layer 32 filters are applied to the input data. This indicates that the layer will yield 32 separate feature maps. Kernel size as mentioned is 3x3.

2.18.2 Model With Two Convolutional Layers (uocseq1)

The second model consists of two convolutional layers. The first one is the same as the previous model as well as the second one with the difference that we do not specify the shape of the input data. Next comes the Max Pooling, the Dropout and the Flatten layer with the values being the same as the model with one layer. Then follows a Dense layer with 64 units and a ReLu activation. This is followed by a Dropout layer with value 0.5 and a Dense layer with 16 units and L1 regularization with strength of 0.0002. There is also a Dense layer that includes the number of classes as well as the softmax activation. It contains the same parameters with the difference in the learning rate which is increased and has the value 0.001. As evaluation metric is used accuracy again.

2.18.3 Model With Four Convolutional Layers (uocseq2)

The first two convolutional layers, Max Pooling, and dropout are the same as in the previous model. Then there are two identical convolutional layers, followed by Max Pooling, Dropout, and a Flatten layer. Next, there is a Dense layer with 32 units, L1 regularization with a strength of 0.001, and ReLU activation. Following that is a Dropout layer and another Dense layer with 16 units, applying the same regularization and activation as the previous Dense layer. Subsequently, there is another Dropout layer, and finally, a Dense layer is added, which includes the number of classes and employs softmax activation. In the optimization process, SGD is employed with a learning rate of 0.1. To enhance convergence, a learning rate decay of 10^6 and momentum with value 0.9. Furthermore, momentum with a value of 0.9 is used, allowing earlier gradient information to impact the current update step, allowing for faster convergence. The NAG is true and a clip-norm mechanism is implemented to restrict the gradient norms to a maximum value of 5. At the end, similar to the other models, the model is compiled using the binary cross-entropy loss function, accuracy as an early stopping metric, and an SGD optimizer.

2.18.4 Our proposal

Our proposal to improve the existing model is a model with three convolutional layers.In the first layer there are 16 filters, kernel size 3x3 and striding is $(1, 1)$. As for the padding, we change it to "same" to ensure that the output feature map have the same spatial dimensions as the input image. To achieve this, padding is added to the input sound so that the convolutional operation covers the entire input image without going beyond its boundaries. Also,there is an input shape parameter. Then, we have added a batch normalization layer. Generally, it is used to normalize the activation of the layer. It substracts the mini-batch standard deviation. This ensures that the activation have a mean of approximately zero. Following normalization, learnable parameters, specifically a scaling factor (gamma) and a shifting factor (beta) are used to scale and shift the activation. The network may adjust and learn the best scale and shift for the normalized activation by the learning these parameters during training. So,we add this lager to our model,in order to normalize the activation of the previous layer within each mini-batch during training.Also we add the parameter momentum with value 0.999 which controls the computation of the moving average of the batch-wise mean and variable.

Following batch normalization, a ReLU activation layer is added that introduces non-linearity to the model. Moreover, a Max Pooling layer is added to reduce the spatial dimensions of the feature maps, using a 2x2 pool size. The code repeats a set of layers twice, each containing a convolutional layer, batch normalization, ReLU activation, and max pooling. The number of filters increases (32 and 64) in these blocks, enabling the model to learn more complex features as it goes deeper. Regularization for dropouts is applied at a rate of 0.25. Dropout prevents overfitting by setting a portion of the input units to zero at random during training. In order to prepare the data for fully connected layers, the flatten layer is utilized to transform the 2D feature maps into a 1D vector. Then, two fully connected layers are added. The first has 32 neurons with ReLU activation.a Dropout layer with a rate of 0.25 follows to further regularize the model. The second fully connected layer has 16 neurons and uses L1 regularization. The final fully connected layer has as many neurons as there are classes in your classification problem. It uses the softmax activation function for multi-class classification. The model is compiled with a binary cross-entropy loss function and an Adam optimizer. Again, accuracy is used as an early stopping criterion.

Chapter 3

Databases

Physionet is a web-based resource intended to support ongoing research and drive new explorations into complex physiologic and clinical data. Its purpose is to facilitate and catalyze biomedical research and education. It encompasses innovative work on the analysis of physiological markers from both basic and translational perspectives. Some of the databases from Physionet containing PCG and ECG recordings are presented:

3.1 EPHNOGRAM

The current database was collected from 24 healthy persons aged 23 to 29 (average: 25.4 ± 1.9 years) in 30-min stress-test sessions while resting, walking, jogging, and biking on indoor fitness center equipment using version 2.1 of the created hardware. The dataset also includes many 30s sample records collected when at rest.It consists, in particular, of 69 simultaneous ECG and PCG recordings, each lasting 30 seconds (8 records) to 30 minutes (61 records) and collected concurrently from a three-lead ECG and a single PCG stethoscope. It sheds light on the interactions between the mechanical and electrical systems of the heart during rest and physical exercise [40].

3.2 SUFHSDB

The Shiraz University (SU) fetal heart sounds database (SUFHSDB) contains 109 pregnant women's fetal and maternal phonocardiogram (PCG) recordings from single and twin pregnancies. The recordings were made with a digital stethoscope on the lower maternal abdomen at Hafez Hospital of Shiraz University of Medical Sciences in Shiraz, Iran, on mothers aged 16 to 47 years. In total, 99 patients had one signal captured, three subjects had two, and seven cases of twins were individually recorded, for a total of 119 recordings. Each record lasts roughly 90 seconds on average. The sampling rate was mainly 16000 Hz with 16-bit quantization and a few 44100 Hz recordings. The data was collected using the digital stethoscope's wide-band mode and a frequency response of 20 Hz to 1 kHz [41].

3.3 MIMIC-IV-ECG

The MIMIC-IV-ECG module has about 800000 diagnostic electrocardiograms from nearly 160000 different patients. These diagnostic ECGs have 12 leads and last 10 seconds. They are sampled at a rate of 500 Hz. MIMIC-IV-ECG includes patients from the MIMIC-IV Clinical Database who had ECGs taken between 2008 and 2019 [42].

3.4 Brno University of Technology ECG Signal Database with Annotations of P Wave (BUT PDB)

Brno University of Technology ECG Signal Database with Annotations of P Wave (BUT PDB) is an ECG signal database with highlighted peaks of P waves established by the cardiology team at Brno University of Technology's Department of Biomedical Engineering. The database contains 50 2-minute 2-lead ECG signal recordings with diverse pathologies. The ECGs were chosen from three existing ECG signal databases: the MIT-BIH Arrhythmia Database, the MIT-BIH Supraventricular Arrhythmia Database, and the Long Term AF Database. Two ECG professionals manually annotated the P wave positions in all 50 samples.Each record additionally includes annotation of the positions and types of QRS complexes (from the original database) as well as the prevailing diagnosis (pathology) included in the record. This database was built for the purpose of developing, evaluating, and objectively comparing P wave detection techniques [43].

3.5 Norwegian Endurance Athlete ECG Database

The Norwegian Endurance Athlete ECG Database contains 12-lead ECG recordings from 28 competitive athletes in Norway representing various sports. This study enlisted the participation of 28 healthy athletes. Men made up 19 (68%) of the participants, while women made up 9 (32%). The ages of the participants ranged from 20 to 43 years old (Mean = 25 years, standard deviation = 4.7 years). The sports distribution was as follows: 24 rowers (86%), 2 kayakers (7%), and 2 cyclists (7%).All recordings are resting ECGs of 10 seconds taken with a General Electric (GE) MAC VUE 360 electrocardiograph. All ECGs are interpreted using the GE Marquette SL12 algorithm (version 23 (v243)) and one cardiologist who has received training in interpreting athlete ECGs. [44]

3.6 ECG-ID Database

Tatiana Lugovaya produced and supplied the ECG-ID Database, which comprises 310 ECGs from 90 volunteers. The database includes 310 ECG recordings from a total of 90 individuals.[45] Each recording consists of ECG lead I, which was captured for a duration of 20 seconds and digitized at 500 Hz, featuring a 12-bit resolution across a nominal 10 mV range. Additionally, each recording contains annotations for 10 beats (unaudited R- and T-wave peak annotations derived from an automated detector), along with accompanying information such as age, gender, and recording date, which is stored in the respective record's. These recordings were obtained from a diverse group of volunteers, comprising 44 men and 46 women ranging in age from 13 to 75. The volunteers consisted of students, coworkers, and friends of the author, and the number of recordings collected for each individual varies, spanning from two recordings acquired on a single day to twenty recordings collected over a six-month period.

3.7 Lobachevsky University Electrocardiography Database

Lobachevsky University Electrocardiography Database (LUDB) is an ECG signal database with indicated P, T, and QRS complex boundaries and peaks [46]. The database contains 200 10-second 12-lead ECG signal records representing various ECG signal morphologies. In $2017 - 2018$, ECGs were obtained from healthy volunteers and patients at Nizhny Novgorod City Hospital No 5. The patients had a variety of cardiovascular illnesses, and some had pacemakers. Cardiologists personally labeled the margins of P waves, T waves, and QRS complexes for all 200 records. Each record is also marked with the appropriate diagnosis.

3.8 MIT-BIH ECG Compression Test Database

This database contains 168 brief ECG recordings (20.48 seconds each) chosen to present a variety of problems to ECG compressors, including lossy compression algorithms [47].

3.9 Databases for our classification

For the purpose of conducting comparative experiments, two databases were employed, each showcasing notable differences in terms of patient ages and medical conditions. The first database, known as the UoC-murmur database, is a proprietary collection that accurately portrays real-life scenarios encountered in pediatric cardiology screening through the utilization of phonocardiograms (PCGs). Distinguished by its focus on pediatric cases, this database stands as one of the most extensive compilations of digital phonocardiograms exclusively within the pediatric context.The second database, in contrast, is publicly accessible and goes by the name PhysioNet-2016 database [48]. This repository predominantly consists of heart sound recordings from adult subjects. It stands as the most expansive and up-to-date collection of phonocardiograms globally, serving as a cornerstone for the comparison of various classification studies.

3.9.1 University of Crete, PCGs with murmur (UoC-murmur) database

This database contains anonymized digital phonocardiograms ($4 - 10$ seconds in length, including 4 to 18 PCG cycles with an average of 8 cycles) obtained from pediatric cardiology outpatients as standard of care and from a pilot pediatric cardiology screening program for school-age children (8 years old), which included digital phonocardiogram as a component for pediatric heart disease screening. The database contains aberrant murmurs associated with various types and levels of CHD severity. As a result, this collection contains samples with aberrant murmurs recorded from children of diverse ages, often under inadequate recording conditions, or harmless murmurs that were difficult for their pediatricians to define as such.

The database comprises 336 recordings obtained from 327 healthy children exhibiting innocent murmurs, alongside 130 recordings stemming from 117 children spanning various age groups, encompassing infants through adolescents, who presented with diverse forms of congenital heart defects (CHD). Each patient underwent four recording sessions, corresponding to specific anatomical locations: apical, lower left (fourth intercostal space), and upper (second intercostal space) left and right parasternal positions.

The data encompassed both ECG signals and digital acoustic data, with a sampling rate of 44100 kHz and a dynamic resolution of 16 bits. These data were then stored in the form of wave files. To streamline the data, an initial preprocessing phase was undertaken. Firstly, the ECG data underwent downsampling, reducing its original frequency to 882 Hz. Similarly, the phonocardiogram (PCG) data was downsampled to 4410 Hz. Secondly, both the ECG and PCG signals underwent amplitude adjustment, setting their maximum levels at 0.9 to achieve uniformity and facilitate subsequent analyses.

3.9.2 PhysioNet-2016 database

The 2016 PhysioNet Challenge seeks to create a large database with the aim of providing automated diagnosis on the mobile phone, providing access to patients wherever they are. The current Challenge aims to promote the creation of algorithms that can accurately categorize heart sound recordings obtained from various clinical or nonclinical environments as normal or abnormal, and subsequently determine whether the subject of the recording needs to be forwarded on for an expert diagnosis.In this competition 348 open-source entries were submitted by 48 teams.

The database comprises nearly 30 hours of recordings, encompassing 233512 heart sounds derived from 116,865 individual heartbeats. These recordings are extracted from a collection of 4430 individual recording sessions, involving a total of 1072 subjects.The heart sound recordings were categorized into two types: normal recordings from healthy subjects, and abnormal recordings from patients diagnosed with heart valve defects and coronary artery disease (CAD). Specifically, the evaluated heart valve defects included mitral valve prolapse, mitral regurgitation, aortic regurgitation, aortic stenosis, and instances involving valvular surgery. A detailed sub-classification for these abnormal recordings was not provided.

Labeling

Signal quality labels are provided to identify recordings with low signal quality. These recordings, labeled as "unsure" ,also received reference segmentation annotations for the four heart sound states (first sound S1, systole, second sound S2, and diastole).

Train and test data

Four of the eight independent heart sound databases were separated into training and test sets, with a 70:30 training-test split. The remaining four databases were assigned to either the training or test sets. The training and test populations are mutually exclusive (no recordings from the same subject/patient are included in both training and test sets). The challenge training set (a through f) has 3153 heart sound recordings from 764 subjects/patients, whereas the test set (b through e, plus g and i) contains 1277 heart sound recordings from 308 subjects/patients. Following the manual correction technique for the segmentation annotations, there are 84425 beats in the training set and 32440 beats in the test set.The recordings range in length from a few seconds to more than a hundred seconds.All recordings have been resampled to 2000 Hz and are available in uncompressed WAV format.

Algorithms

The Challenge presented a very simple benchmark classifier example. Initially, a balanced heart sound database consisting of 472 abnormal and 472 normal recordings was selected from the training set. Springer's segmentation algorithm was employed to segment each chosen cardiac sound recording, thereby generating time durations corresponding to the four states: S1, systole, S2, and diastole. From the location data of these states, twenty distinct characteristics were derived.Subsequently, these twenty characteristics were utilized in a binary logistic regression classifier employing forward selection. This process aimed to identify the most advantageous characteristics for classification. It's important to note that the 'unsure' class was omitted from consideration, resulting in the binary outcome of either 'normal' or 'abnormal'. After conducting the analysis, it was determined that the optimal number of characteristics for effective categorization was seven.During the training phase, a set of five features was identified, leading to a sensitivity of 0.66, a specificity of 0.77, and a challenge score of 0.71 through a 10-fold cross-validation process. These specific characteristics were selected due to their evident significance; the classifier was not subjected to cross-validation, and only a random subset of the data was utilized in this analysis. Furthermore, the top N submissions from the competition were employed to devise a straightforward, unweighted voting mechanism. It's worth noting that by excluding submissions with lower performance, this strategy commonly results in achieving the highest score within the competition.

Scoring

The total score is determined by the number of recordings that are classified as normal, abnormal, or unsure.

$$\text{Sensitivity} : S_e = \frac{wa_1 + Aa_1}{Aa1 + Aq1 + An1} + \frac{wa_2(Aa_2) + Aq_2}{Aa_2 + Aq_2 + An_2} \tag{3.1}$$

$$\text{Specificity} : S_p = \frac{wn_1 + Aa_1}{Na_1 + Nq_1 + Nn_1} + \frac{wn_2 * (Nn_2 + Nq_2)}{Na_2 + Nq_2 + Nn2} \tag{3.2}$$

where wa_1 and wa_2 are the percentages that measure the quality of the signal in abnormal recordings,and they are weights for calculating S_e. Respectively, wn_1 and wn_2 are percentages that measure the quality of the signal in normal recordings,and they are weights for calculating S_p. Finally, the Challenge score is calculated by the so-called *Balanced Accuracy*:

$$\text{MAcc} = \frac{S_e + S_p}{2} \tag{3.3}$$

Results

Although there isn't much of a performance difference between the top three entries (in terms of the MAcc), we should notice that Potes et al.'s [49] highest scoring entry had a notable Se and low Sp. Potes et al. had the second-highest overall Se, as we also highlighted. (The highest Se was 0.9633, ranking 34th, but with a low Sp of 0.5589 and a high MAcc of 0.7611). The third-highest Se, which came in fifth, had a score of 0.8848. Rubin et al. [50] had the highest Sp (0.9521), but their Se was just 0.7278, placing them in eighth place overall. A higher sensitivity may be ideal for an application that sends subjects for additional screening, provided that the resources can handle the false-positive rate.However, with only a 1.5% difference between them, the second, third, and fourth candidates offer a decent mix between Se and Sp. There is a 2% gap between the top eight competitors. Finally, take notice of the sample algorithm's identical performance on the training and test sets of data. The score varied between 0.47 on training set b to 0.86 on training set c, with a mean ± SD of 0.59 ± 0.15 across all training databases, after stratifying by patient database and running a leave-one-database-out cross validation on the training data. This shows how challenging it is to train an algorithm for brand-new datasets under brand-new recording circumstances. Two entirely hidden databases (g and i) were included in the test data that was hidden.

In conclusion, the database made available for this Challenge is the largest open access database of heart sounds in the world.

Chapter 4

Experiments

4.1 Preprocessing

Regarding the prepossessing phase, no alterations were made as the available options had previously yielded optimal results in the prior study.

4.1.1 Segmentation

The segmentation is responsible for dividing a continuous signal into smaller, discrete segments or frames. It offers options for period-synchronous or asynchronous segmentation and has fixed frame size (2 seconds). The class's initialization parameters allow us to specify the source type of the signal (ECG or PCG), whether the segmentation should be synchronized with periodic events and the desired size (1 period,2 periods or 3 periods) and hop(overlap) of one second,setting for segment. For segmentation the following 2 libraries were used:

- Soundfile: This library is used for reading audio waveforms from sound files.
- Numpy: Numpy is a fundamental library for numerical operations in Python, and it is used for various array manipulations and calculations in the code.

Subsequent processing procedures were conducted on the PCG (Phonocardiogram) signals. Initially, the PCG waveforms were subdivided into smaller time intervals. A Tukey window was subsequently applied to each of these segments to ensure uniform sizing. Tukey window is a mathematical function that is zero-valued outside of some chosen interval, normally symmetric around the middle of the interval, usually approaching a maximum in the middle, and usually tapering away from the middle [51]. These segments were then either truncated or adjusted to conform to a predefined maximum length, measured in milliseconds. Subsequently, the amplitude spectrum of each segment was computed.In order to achieve further processing of PCG in addition to the two previous libraries, two more libraries were used:

- Spectrum: a library which is used to create a Tukey window
- Matplotlib.pyplot: a library which is used to create visualizations in python.

4.1.2 Features Extraction

For feature extraction is used two parameters: "timeDim" and "freqDim".The timeDim parameter, set to 32, dictates the number of time-based bins or frames into which each data segment is divided during extraction.Conversely, freqDim, set to 16, determines the number of frequency-based bins used to capture spectral characteristics within each time frame.

4.1.3 Data Splitting

Data splitting ensures that the model is trained on one set of data, validated on another set to tune hyperparameters, and finally tested on a separate set to evaluate its performance. In this study the ratios are defined as follows:

- Training set:65%.This set is used to train the machine learning model.
- Validation set:15%.The validation set is used during the training process to tune hyperparameters, monitor the model's performance, and prevent overfitting. It helps in selecting the best model from different training epochs.

- Test set:20%. The test set is not used during model training or hyperparameter tuning. Instead, it is used to evaluate the final model's performance. It provides an estimate of how well the model will perform.

4.1.4 Data Augmentation

Regarding augmentation, we abstained from implementing it due to the inadequacy of our graphics card resources.

4.2 Recommended Architecture and Results

In the initial phase of experimentation with the uocSeq2 model, a systematic investigation was undertaken to evaluate the effects of removing activation functions from all convolutional layers.As a pivotal step, a BatchNormalization layer was meticulously integrated into the model configuration. This BatchNormalization layer was meticulously configured with a momentum parameter assigned a value of 0.999. The primary objective of this addition was to instill stability into the training process and expedite the convergence of the model during the optimization phase. In a subsequent step, an activation layer was judiciously re-introduced into the architecture. Specifically, the ReLU activation function, which had been previously employed within the model, was reinstated. This particular activation layer was employed with the purpose of reinstating non-linearity to the model's computations. The subsequent phase of our experimentation involved the application of a custom model, extensively elucidated in Section 2.1.4 of this study. This custom model, meticulously detailed and described therein, was employed to further assess and validate its performance characteristics within the experimental framework.

First of all, our proposed model differs in the number of filters compared to previous models. By starting with 16 filters and progressively increasing to 32 and then 64, the model is designed to learn hierarchical representations of features. This approach allows the network to gradually build more abstract and higher-level representations of the input data.Furthermore,the number of filters in each layer impacts the computational complexity of the model. Using a relatively small number of filters in the initial layers helps keep the model computationally efficient, making it suitable for training on standard hardware. As the process moves deeper into the network, where feature maps are typically smaller due to max-pooling layers, it can afford to increase the number of filters to capture more complex patterns.Using fewer filters in earlier layers can also contribute to regularization, as it reduces the model's capacity to memorize training data.Finally, this number of filters provideσ a good balance between model complexity and performance through experimentation.

Then, we changed the padding from '*valid*' to '*same*' in all convolutional layers. In this way we preserve the spatial dimensions (height and width) of feature maps after applying convolution.The convolution operation includes zero-padding to the input feature map such that the output feature map has the same spatial dimensions as the input. So,we maintain a consistent spatial resolution throughout the network. Moreover this technique helps the network learn and retain fine details and local patterns in the data. Also,the receptive field (the area in the input image that influences a particular output feature) of each layer remains consistent. This can be beneficial to capture features at a specific scale throughout the network.

Also,we reduced the dropout rate from 0.5 to 0.25. A dropout rate of 0.25 means that during training, approximately 25% of neurons in the dropout layer are randomly set to zero during each forward and backward pass. This is a more moderate level of regularization compared to a dropout rate of 0.5. A lower dropout rate still introduces regularization but to a lesser extent than a higher dropout rate. This provide a balance between preventing overfitting and allowing the model to learn more from the training data. Lower dropout rates leads to faster convergence because the model is less likely to discard useful information during training. So, it requires fewer training epochs to reach a good result. Generally, our code relies on dropout regularization.

In our code kernel regularization (L1 regularization) is applied to only one of the dense layers (the second dense layer with a regularization strength of 0.0002). The other dense layers do not have explicit kernel regularization. Specifically, these layers are not explicitly encouraged to have sparse weights. Sparsity implies that some weights could become exactly zero, effectively reducing the complexity of the model.

In order to improve the performance of our model we changed the optimization. Adam adapts learning rates for each parameter individually. In particular, it adjusts the learning rates during training, potentially leading to faster convergence. In addiction, Adam incorporates bias correction, which helps in the early stages of training when the moving averages are biased towards zero. This leads to more stable training and improved convergence.

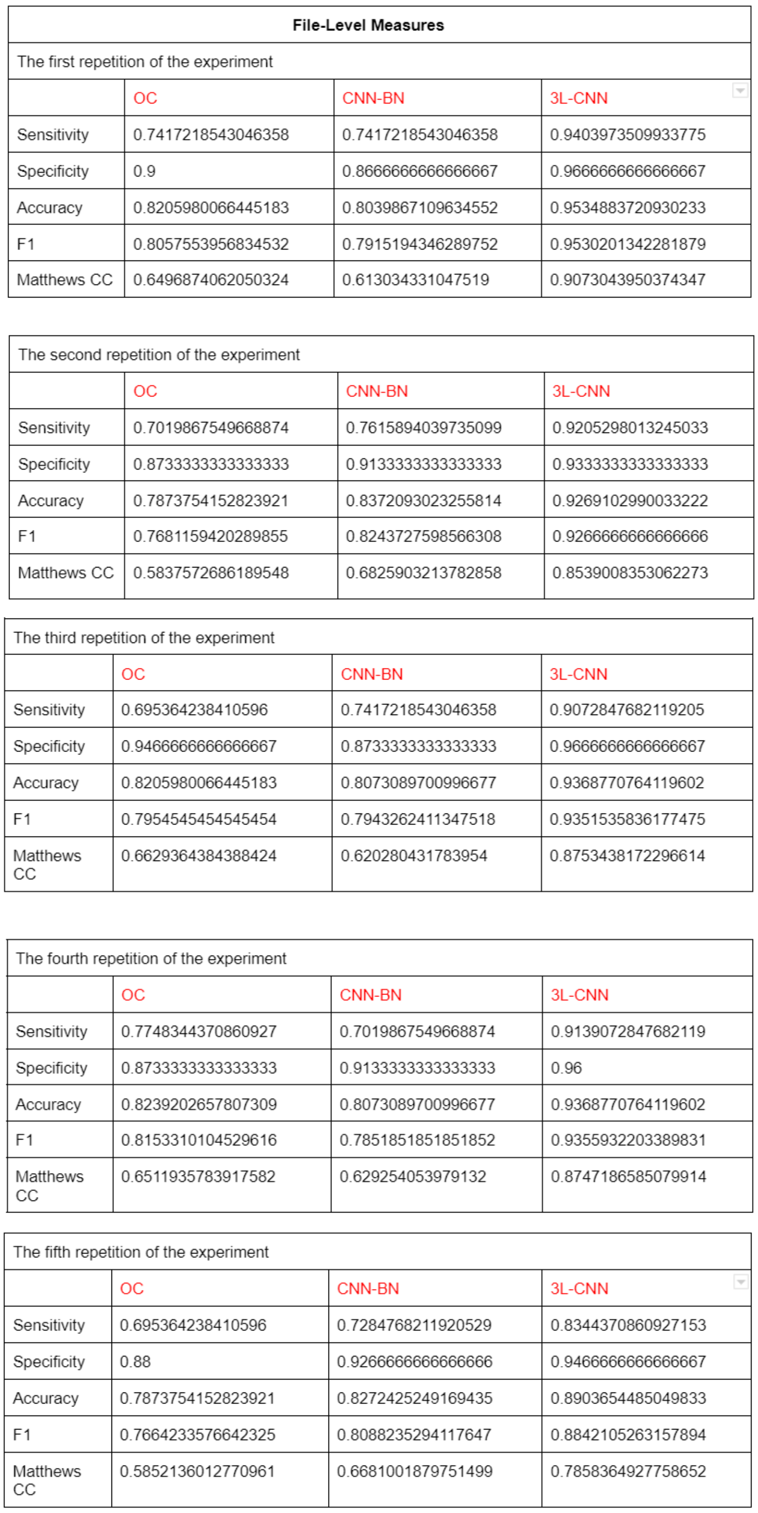
The aforementioned modifications implemented in the code, pertaining to the primary models, have significantly enhanced the effectiveness of our model. These improvements are substantiated by the results we have obtained. Each test generates five repetitions of the identical experiment. The results of each test we applied are shown in the tables and graphs below.

In the realm of experimental research, it is common practice to gather and assess data at both the file level and frame level. This methodology involves the acquisition of metrics or data points pertaining to individual files or units of analysis, as well as the collection and analysis of fine-grained measurements at the level of discrete frames . File-level measures serve to furnish a summarized perspective of the overarching characteristics or attributes of individual files, while frame-level measures are employed to capture nuanced variations, patterns, or features that manifest over sequential frames or time intervals. In classification tasks, file-level measures are pivotal for evaluating the performance of a model.These measures include accuracy, which assesses overall correctness; precision, which focuses on the accuracy of positive classifications; sensitivity, measuring the ability to identify all relevant positive documents; and the F1-Score, a balanced metric combining precision and sensitivity. Specificity is crucial for minimizing false positives, while ROC provide comprehensive assessments of a model's ability to distinguish between classes. Additionally, the confusion matrix offers a detailed breakdown of true positives, true negatives, false positives, and false negatives, enabling the calculation of these file-level measures. MFCC (Mel-Frequency Cepstral Coefficients) score is also used as a file-level measure to represent the spectral characteristics of audio files. On the other hand, frame-level measures is used to evaluate how well a classification model detects specific events or patterns within each frame.

Figure 4.1 and Figure 4.2 are the results tables for the file-level measures and for the frame-level measures.

The observation reveals that the sensitivity value in our proposed model is significantly elevated.So our proposed model is is better at correctly identifying positive cases relative to the total number of actual positive cases in the dataset. Moreover, a higher sensitivity means fewer false negatives. The increased specificity has been duly observed.Our model seems to be better at correctly identifying negative cases relative to the total number of actual negative cases in the dataset. It minimizes the chances of falsely accusing something as positive when it's not,something that is very important in our study.The higher accuracy of our model ensures that the model is more effective at making correct predictions across all classes or categories. Achieving a balance between precision and sensitivity in our model is superior to merely ensuring it through a higher F1 score. Finally, MCC score is better at evaluating the performance particularly when dealing with imbalanced datasets. The MCC takes into account both sensitivity and specificity (true negative rate) and aims to provide a balanced assessment of the model's performance, considering both types of errors (false positives and false negatives). Generally, MCC score indicates better discrimination between classes, robustness to imbalanced data, and balanced performance.

Below are the diagrams derived from the five repetitions of each experiment. These diagrams encompass accuracy, loss, and ROC curves, where Results 0 corresponds to the original code (OC), Results 1 corresponds to the CNN after the addition of the batch normalization layer (CNN-BN) and Results 2 corresponds to our proposed three-layer CNN model (3L-CNN).



File-Level Measures

The first repetition of the experiment

	OC	CNN-BN	3L-CNN
Sensitivity	0.7417218543046358	0.7417218543046358	0.9403973509933775
Specificity	0.9	0.8666666666666667	0.9666666666666667
Accuracy	0.8205980066445183	0.8039867109634552	0.9534883720930233
F1	0.8057553956834532	0.7915194346289752	0.9530201342281879
Matthews CC	0.6496874062050324	0.613034331047519	0.9073043950374347

The second repetition of the experiment

	OC	CNN-BN	3L-CNN
Sensitivity	0.7019867549668874	0.7615894039735099	0.9205298013245033
Specificity	0.8733333333333333	0.9133333333333333	0.9333333333333333
Accuracy	0.7873754152823921	0.8372093023255814	0.9269102990033222
F1	0.7681159420289855	0.8243727598566308	0.9266666666666666
Matthews CC	0.5837572686189548	0.6825903213782858	0.8539008353062273

The third repetition of the experiment

	OC	CNN-BN	3L-CNN
Sensitivity	0.695364238410596	0.7417218543046358	0.9072847682119205
Specificity	0.9466666666666667	0.8733333333333333	0.9666666666666667
Accuracy	0.8205980066445183	0.8073089700996677	0.9368770764119602
F1	0.7954545454545454	0.7943262411347518	0.9351535836177475
Matthews CC	0.6629364384388424	0.620280431783954	0.8753438172296614

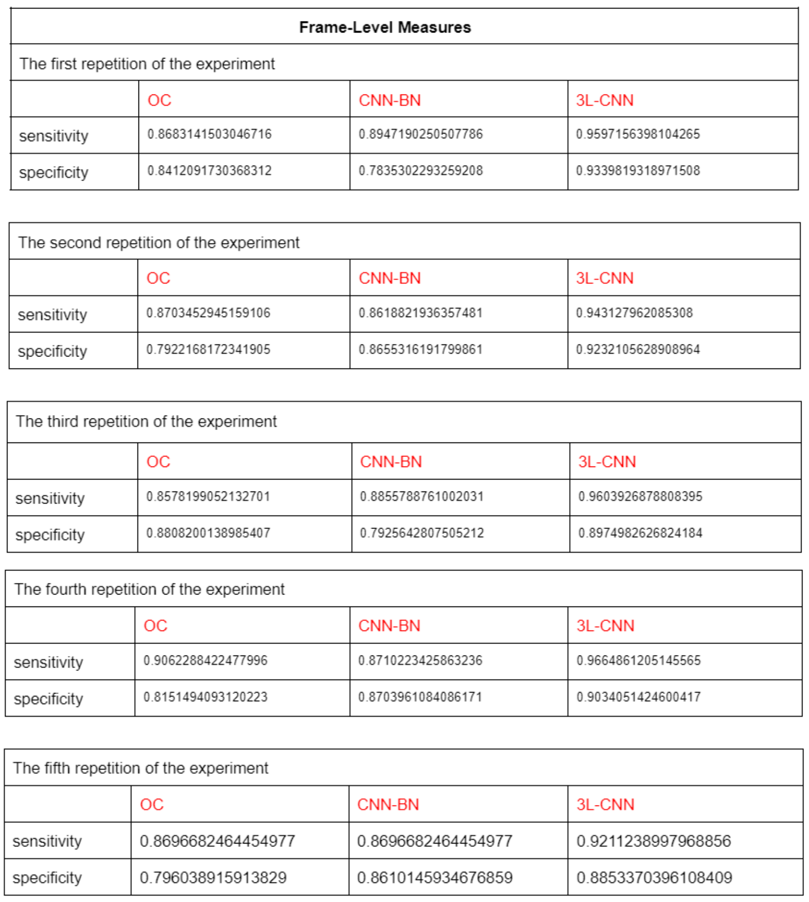
The fourth repetition of the experiment

	OC	CNN-BN	3L-CNN
Sensitivity	0.7748344370860927	0.7019867549668874	0.9139072847682119
Specificity	0.8733333333333333	0.9133333333333333	0.96
Accuracy	0.8239202657807309	0.8073089700996677	0.9368770764119602
F1	0.8153310104529616	0.7851851851851852	0.9355932203389831
Matthews CC	0.6511935783917582	0.629254053979132	0.8747186585079914

The fifth repetition of the experiment

	OC	CNN-BN	3L-CNN
Sensitivity	0.695364238410596	0.7284768211920529	0.8344370860927153
Specificity	0.88	0.9266666666666666	0.9466666666666667
Accuracy	0.7873754152823921	0.8272425249169435	0.8903654485049833
F1	0.7664233576642325	0.8088235294117647	0.8842105263157894
Matthews CC	0.5852136012770961	0.6681001879751499	0.7858364927758652

Figure 4.1: *Tables of metrics for all five replicates of each experiment. Results 0: OC,Results 1:CNN-BN,Results 2:3L-CNN.*



Frame-Level Measures

The first repetition of the experiment

	OC	CNN-BN	3L-CNN
sensitivity	0.8683141503046716	0.8947190250507786	0.9597156398104265
specificity	0.8412091730368312	0.7835302293259208	0.9339819318971508

The second repetition of the experiment

	OC	CNN-BN	3L-CNN
sensitivity	0.8703452945159106	0.8618821936357481	0.943127962085308
specificity	0.7922168172341905	0.8655316191799861	0.9232105628908964

The third repetition of the experiment

	OC	CNN-BN	3L-CNN
sensitivity	0.8578199052132701	0.8855788761002031	0.9603926878808395
specificity	0.8808200138985407	0.7925642807505212	0.8974982626824184

The fourth repetition of the experiment

	OC	CNN-BN	3L-CNN
sensitivity	0.9062288422477996	0.8710223425863236	0.9664861205145565
specificity	0.8151494093120223	0.8703961084086171	0.9034051424600417

The fifth repetition of the experiment

	OC	CNN-BN	3L-CNN
sensitivity	0.8696682464454977	0.8696682464454977	0.9211238997968856
specificity	0.796038915913829	0.8610145934676859	0.8853370396108409

Figure 4.2: *Tables of sensitivity and specificity for all five replicates of each experiment. Results 0: OC,Results 1:CNN-BN,Results 2:3L-CNN.*

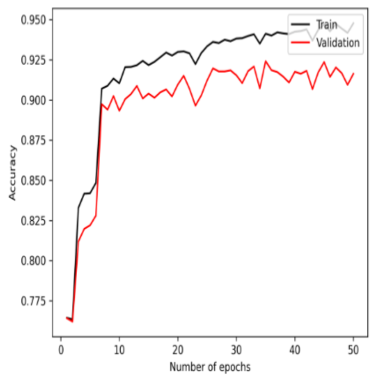


Figure 4.3: OC

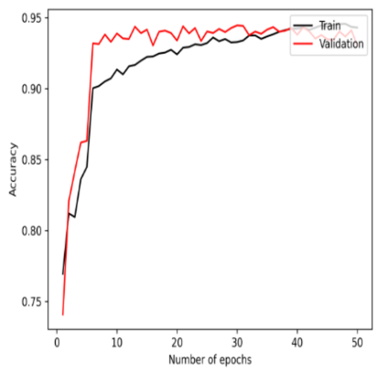


Figure 4.4: CNN-BN

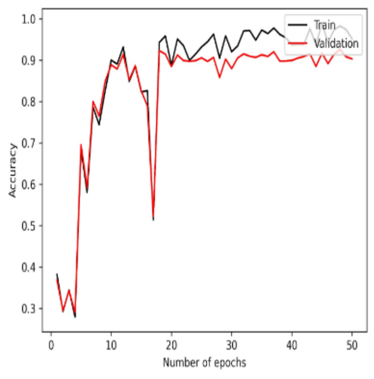


Figure 4.5: 3L-CNN

Figure 4.6: *Accuracy in the first repetition.*

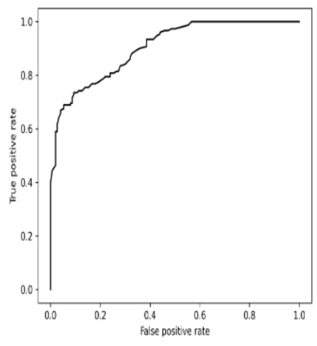


Figure 4.7: *OC*

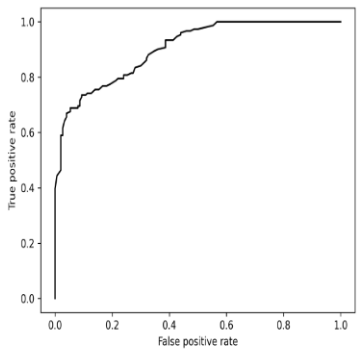


Figure 4.8: *CNN-BN*

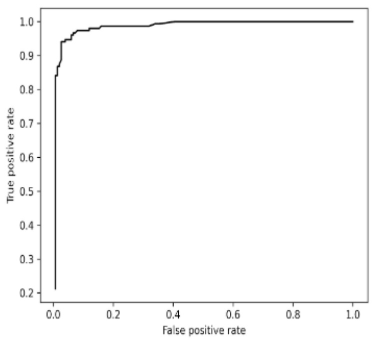


Figure 4.9: *3L-CNN*

Figure 4.10: *ROC-Curve in the first repetition.*

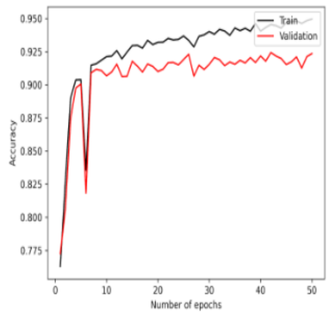


Figure 4.11: OC

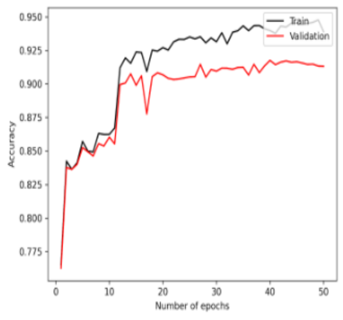


Figure 4.12: CNN-BN

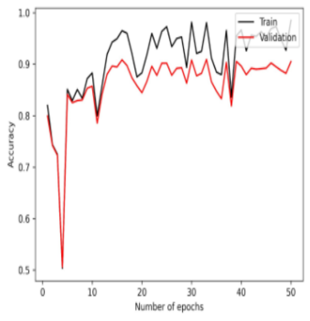


Figure 4.13: 3L-CNN

Figure 4.14: *Accuracy in the second repetition.*

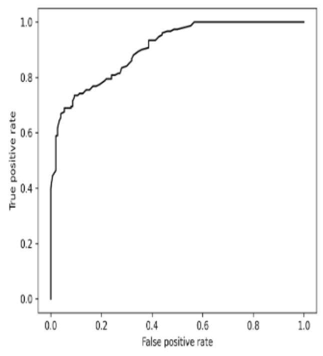


Figure 4.15: OC

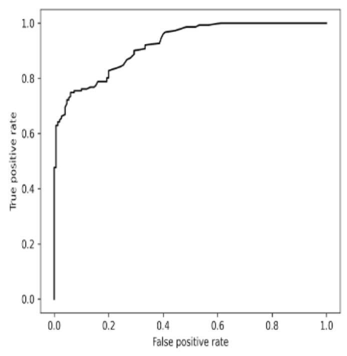


Figure 4.16: CNN-BN

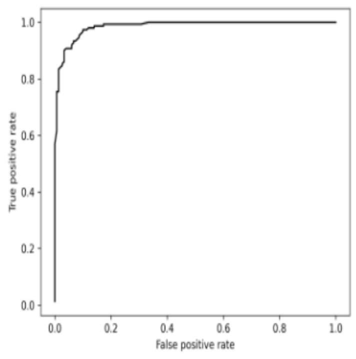


Figure 4.17: 3L-CNN

Figure 4.18: *ROC-Curve in the second repetition.*

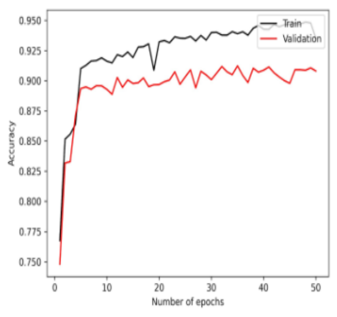


Figure 4.19: OC

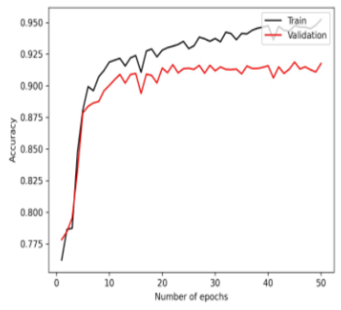


Figure 4.20: CNN-BN

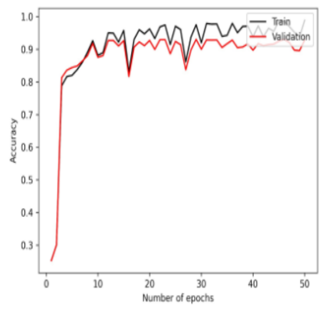


Figure 4.21: 3L-CNN

Figure 4.22: *Accuracy in the third repetition.*

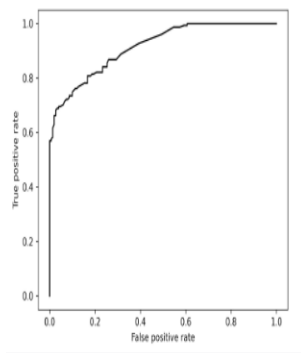


Figure 4.23: OC

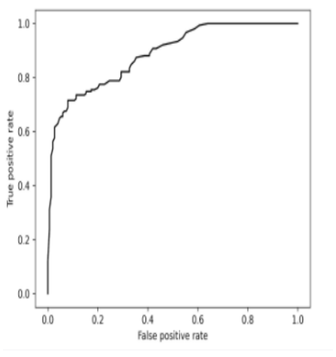


Figure 4.24: CNN-BN

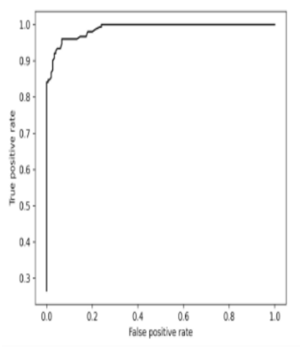


Figure 4.25: 3L-CNN

Figure 4.26: *ROC-Curve in the third repetition.*

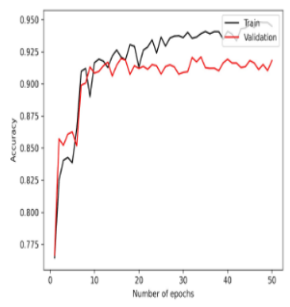


Figure 4.27: OC

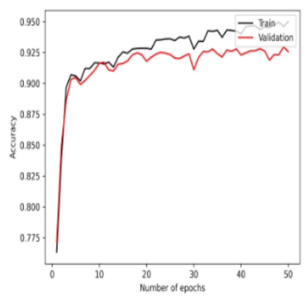


Figure 4.28: CNN-BN

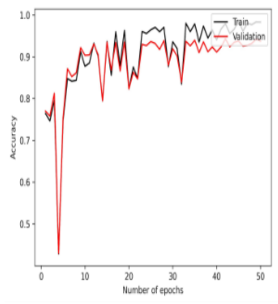


Figure 4.29: 3L-CNN

Figure 4.30: *Accuracy in the fourth repetition.*

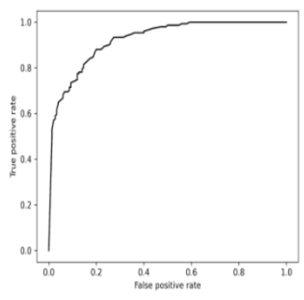


Figure 4.31: OC

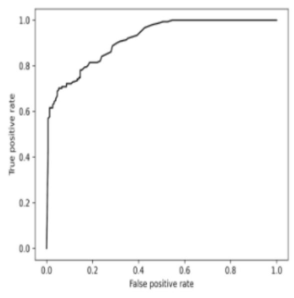


Figure 4.32: CNN-BL

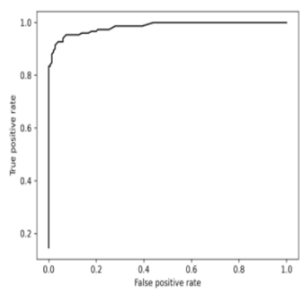


Figure 4.33: 3L-CNN

Figure 4.34: *ROC-Curve in the fourth repetition.*

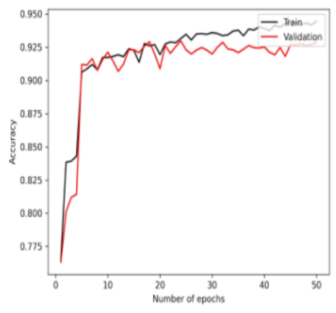


Figure 4.35: OC

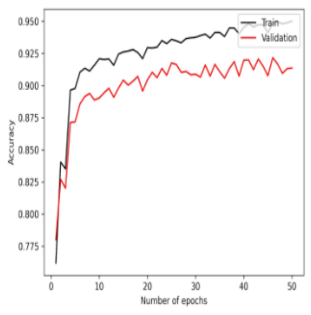


Figure 4.36: CNN-BN

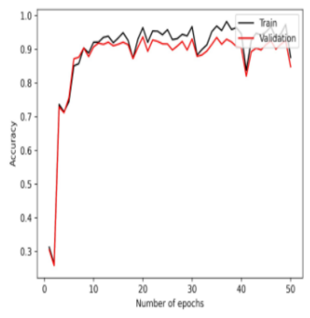


Figure 4.37: 3L-CNN

Figure 4.38: *Accuracy in the fifth repetition.*

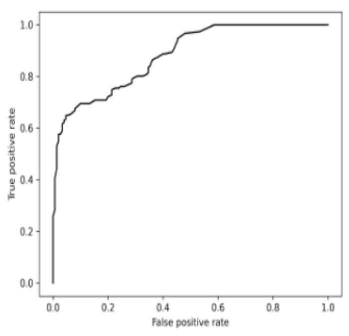


Figure 4.39: OC

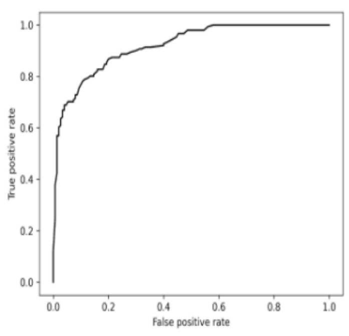


Figure 4.40: CNN-BN

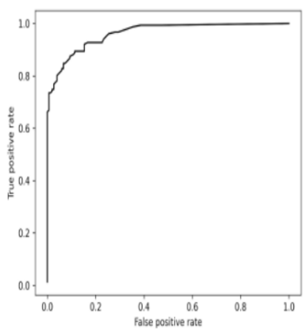


Figure 4.41: 3L-CNN

Figure 4.42: *ROC-Curve in the fifth repetition.*

Chapter 5

Conclusion and Future Work

5.1 Results

According to the results obtained, we see an obvious improvement of all metrics in all five experiments. More specifically, as far as the file-level measures for the metric sensitivity are concerned, 3L-CNN shows an increase of $\approx$ 25.2% compared to OC. On the other hand, CNN-BN compared to OC sensitivity shows a very small increase of $\approx$ 5.5% while in the fourth repetition of the experiment it decreased by $\approx$ 9%. In terms of specificity, 3L-CNN shows an increase of about $\approx$ 6.75% compared to OC and CNN-BN $\approx$ 5%. However, the first repetition of the CNN-BN experiment reduces specificity by 4% and the third by 7.7%. We then observe that 3L-CNN increases the accuracy by $\approx$ 14.98% compared to OC. As for the comparison between CNN-BN and OC in the first, third and fourth repetition we have a $\approx$ 2% decrease in accuracy while in the second and fifth repetition we have a $\approx$ 5.5% increase. F1 score is increased by an average of 17.32% and Matthews CC by 37.32% in 3L-CNN model. In the CNN-BN model in the first, third, and fourth repetition the F1 score decreases by $\approx$ 1.8% while in the second and fifth it increases by $\approx$ 6.5%. Similarly the Matthews CC decreases in the same repetitions by $\approx$ 9% while it increases by $\approx$ 15.5%.

As for the frame level measures we had two metrics; sensitivity and specificity. For sensitivity, we first compared the OC with the CNN-BN and observed in the first and third repetition an increase of $\approx$ 3% while in the second and fourth a small decrease of $\approx$ 2% is observed, in the fifth repetition no change is observed. In the comparison of OC with 3L-CNN, sensitivity generally increased by 9% while in the second repetition it decreased by 9%. The specificity in 3L-CNN increased by 10%, while in CNN-BN in the first and third repetitions it decreased by 8.5% and in the remaining repetitions it increased by 8.6%

In conclusion, our model outperforms in multiple aspects of classification accuracy and balance. The model is achieving better overall performance compared to the other models. That makes our model more robust and reliable in various situations and across different threshold settings. It's not overly biased towards either precision or sensitivity.

In all instances of evaluating model performance, a notably elevated accuracy, converging closely to unity, has been attained. Furthermore, a concordance between the training and validation sets has been observed, indicating a model that is exhibiting competence not only on its training data but also on hitherto unseen data. Of particular note is the observation that "result 1," in comparison to alternative outcomes, manifests a superior outcome in terms of minimizing the loss function across both the training and validation datasets. Additionally, it is noteworthy that the ROC curve, a metric of classification model performance, converges to a value of one within a shorter span of training epochs, suggesting expedited convergence.

5.2 Future Work

In our study, we have identified several avenues for future research to enhance the efficacy of our PCG (Phonocardiogram) classification system. Firstly, there is a need to investigate more sophisticated data augmentation techniques that consider the intrinsic physical constraints of the heart's cardiac cycles, moving beyond the current uniform resampling approach. Secondly, a comprehensive analysis of the sub-band filtering process, optimizing parameters for Gammatone filter banks, could lead to improved performance. Additionally, exploring alternative fusion methods for frame-level classification results, such as majority voting, should be considered. Further research could also focus on parameter optimization to fine-tune our PCG classification system. Moreover, the integration of multi-sensor signal processing techniques, including noise cancellation using multi-channel PCG recordings and joint analysis with other modalities like Ballistocardiography (BCG) and Electrocardiography (ECG), holds promise for enhancing the screening applications. Ultimately, building and testing end-products in real-life scenarios should be a pivotal direction for our research community to pursue.

References

[1]Phonocardiography

[2] Baris Bozkurt, Ioannis Germanakis, Yannis Stylianou (2018) A study of time-frequency features for CNN-based automatic heart sound classification for pathology detection.

[3] Reller M.D., Strickland M.J., Riehle-Colarusso T., Mahle W.T., Correa A. Prevalence of congenital heart defects in metropolitan Atlanta (1998–2005).

[4] Suyi Li,Feng Li,Shijie Tang,Wenji Xiong (2020).A Review of Computer-Aided Heart Sound Detection Techniques.

[5] Wei Chen, Qiang Sun, Xiaomin Chen, Gangcai Xie, Huiqun Wu, and Chen Xu,Deep Learning Methods for Heart Sounds Classification: A Systematic Review (2021)

[6] Abbas K. Abbas, Rasha Bassam. Phonocardiography Signal Processing

[7] Ganesh R. Naik,Wellington Pinheiro dos Santos. Biomedical Signal Processing: A Modern Approach (2023)

[8] Praharsh Ivaturi, Matteo Gadeleta, Amitabh C.Pandey, Michael Pazzani, Steven R.Steinhubl, Giorgio Quer. A Comprehensive Explanation Framework for Biomedical Time Series Classification (2021)

[9] Han Li, Xinpei Wang, Changchun Liu,Qiang Zeng, Yansong Zheng,Xi Chu, Lianke Yao,Jikuo Wang, Yu Jiao,Chandan Karmakar. A fusion framework based on multi-domain features and deep learning features of phonocardiogram for coronary artery disease detection.(2020)

[10] Sean Dornbush, Andre E. Turnquest.Physiology, Heart Sounds (2023)

[11] Jithendra Vepa. Classification of heart murmurs using cepstral features and support vector machines

[12] D Kumar ,P Carvalho, M Antunes, R P Paiva, J Henriques,Heart murmur classification with feature selection

[13] Ahmed Ali Dawud, Thamineni Bheema Lingaiah, Towfik Jemal. Classification of heart sounds associated with murmur for diagnosis of cardiac valve disorders(2022)

[14] Si-ji Ding, Hao Ding, Meng-fei Kan, Yi Zhuang, Dong-yang Xia, Shi-meng Sheng,Xin-ru Xu.A Computer-Aided Heart Valve Disease Diagnosis System Based on Machine Learning (2023)

[15] Qaisar Abbas, Ayyaz Hussain, Abdul Rauf Baig. Automatic Detection and Classification of Cardiovascular Disorders Using Phonocardiogram and Convolutional Vision Transformers (2022)

[16] Guang Yang, Qinghao Ye, Jun Xia. Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond (2020)

[17] Yogesh Kumar, Apeksha Koul, Ruchi Singla, and Muhammad Fazal Ijaz.Artificial intelligence in disease diagnosis: a systematic literature review, synthesizing framework and future research agenda (2022)

[18] Seth L. Thomas, Joseph Heaton, Amgad N. Makaryus. Physiology, Cardiovascular Murmurs (2023)

[19] Juan P Dominguez-Morales, Angel F Jimenez-Fernandez, Manuel J Dominguez-Morales, Gabriel Jimenez-Moreno.Deep Neural Networks for the Recognition and Classification of Heart Murmurs Using Neuromorphic Auditory Sensors (2017)

[20] Mohanad Alkhodari, Luay Fraiwan. Convolutional and recurrent neural networks for the detection of valvular heart diseases in phonocardiogram recordings (2021)

[21] Robinson Spencer, Fadi Thabtah, Neda Abdelhamid, Michael Thompson. Exploring feature selection and classification methods for predicting heart disease (2020)

[22] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning (Adaptive Computation and Machine Learning series) (2022)

[23] What is a neural network

[24] Neural.

[25] Neural Networks.

[26] Hidden Layer

[27] Output Layer.

[28] Neural Networks and Machine Learning.

[29] Mayank Mishra.Convolutional Neural Networks, Explained (2020)

[30] What are convolutional neural networks?

[31] Number of filters

[32] Padding

[33] Pooling

[34] Saul Dobilas. Convolutional Neural Networks Explained — How To Successfully Classify Images in Python (2022)

[35] Hyperparameter.

[36] ReLu Activation.

[37] Softmax.

[38] Nagesh Singh Chauhan.Optimization Algorithms in Neural Networks (2020)

[39] Adam optimizer

[40] EPHNOGRAM

[41] SUFHSDB

[42] MIMIC-IV-ECG

[43] Brno University of Technology ECG Signal Database with Annotations of P Wave (BUT PDB)

[44] Norwegian Endurance Athlete ECG Database

[45] ECG-ID Database

[46] Lobachevsky University Electrocardiography Database

[47] MIT-BIH ECG Compression Test Database

[48] Physionet

[49] Cristhian Potes, Saman Parvaneh, Asif Rahman, Bryan Conroy (2016) Ensemble of feature-based and deep learning-based classifiers for detection of abnormal heart sounds.

[50] Jonathan Rubin, Rui Abreu, Anurag Ganguli, Saigopal Nelaturi, Ion Matei, Kumar Sricharan (2016) Clas-

sifying heart sound recordings using deep convolutional neural networks and mel-frequency cepstral coefficients.

[51] Tukey Window