Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)

HEART MURMUR DETECTION USING DEEP LEARNING AUDIO CLASSIFIER

Submitted in partial fulfillment of the requirement
of the degree of

Bachelor of Technology in

Computer Engineering

By

Dhruvi Jodhawat	**60004190032**
Junaid Girkar	**60004190057**
Kanaad Deshpande	**60004190058**

Under the guidance of

Professor Sudhir Bagul

University of Mumbai

A.Y. 2022 – 2023

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)

CERTIFICATE

This is to certify that the project entitled, **“**HEART MURMUR DETECTION USING DEEP LEARNING AUDIO CLASSIFIER**”** is a bonafide work of **“Dhruvi Jodhawat” (60004190032)**, **“Junaid Girkar” (60004190057),** and **“Kanaad Deshpande” (60004190058)** submitted in the partial fulfillment of the requirement for the award of the Bachelor of Technology in Computer Engineering

Professor Sudhir Bagul
(Guide)

Dr. Meera Narvekar
(Head of the Department)

Dr. Hari Vasudevan
(Principal)

Place: Dwarkadas Jivanlal Sanghvi College of Engineering

Date:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)

CERTIFICATE

This is to certify that the project entitled, **"HEART MURMUR DETECTION USING DEEP LEARNING AUDIO CLASSIFIER"** is a bonafide work of **"Dhruvi Jodhawat" (60004190032)**, **"Junaid Girkar" (60004190057),** and **"Kanaad Deshpande" (60004190058)** submitted in the partial fulfillment of the requirement for the award of the Bachelor of Technology in Computer Engineering

Professor Sudhir Bagul **(Guide)**	**Name of the External Guide**
Dr. Meera Narvekar **(Head of the Department)**	**Dr. Hari Vasudevan** **(Principal)**

Place: Dwarkadas Jivanlal Sanghvi College of Engineering

Date:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that, we have adhered to all the principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources, which have thus not been properly cited or from whom proper permission has not been taken, when needed.

Dhruvi Jodhawat (60004190032)

Junaid Girkar (60004190057)

Kanaad Deshpande (60004190058)

Place: Dwarkadas Jivanlal Sanghvi College of Engineering

Date:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)

APPROVAL SHEET

Project entitled, **"HEART MURMUR DETECTION USING DEEP LEARNING AUDIO CLASSIFIER"** submitted by **"Dhruvi Jodhawat" (60004190032), "Junaid Girkar" (60004190057),** and **"Kanaad Deshpande" (60004190058)** is approved for the award of the Bachelor of Technology in Computer Engineering

Signature of Internal Examiner **Signature of External Examiner**

Professor Sudhir Bagul
(Guide)

Dr. Meera Narvekar
(Head of the Department)

Dr. Hari Vasudevan
(Principal)

Place: Dwarkadas Jivanlal Sanghvi College of Engineering

Date:

Abstract

With cases of cardiovascular disorders increasing every day, people often regret not taking precautions to detect them early. It is a difficult and expensive task to diagnose such heart diseases or foresee them as it requires highly skilled doctors and exorbitantly priced equipment. One such disease that plagues humans and animals is heart murmurs. Heart murmurs are sounds caused by the turbulent flow of blood across the auricular or ventricular valves (the murmurs are called auricular and ventricular murmurs, respectively). They are a common phenomenon and sometimes innocent, but if not investigated and analyzed, they may prove to be dangerous. Heart murmurs are an indication of abnormality in the heart, and they simply indicate the presence of diseases including but not limited to bacterial infections in the bloodstream (Endocarditis), structural heart problems (congenital heart defects), low red blood cell count (Anaemia) and septal defects, failure of ductus arteriosus closure in newborns. In this report, we present an alternative approach to detecting heart murmurs in a quick and extremely cost-effective manner by making use of an everyday device like a smartphone as the medium of diagnosis. A medium can be used to record the sound of the heartbeat and present it to the model, which outputs the presence of murmurs in the heart. This project presents an approach to detecting heart murmurs in a quick and extremely cost-effective manner by making use of an everyday device like a smartphone as the medium of diagnosis. A medium can be used to record the sound of the heartbeat and feed it to the model, a modified 8-layer AlexNet, dubbed as a MurmurNet considering 175,154 parameters, treated to various preprocessing techniques which outputs the presence or absence of murmurs in the heart. The model boasts of a 92% validation accuracy that outperforms all the compared models and is trained on a dataset containing more than 3000 labeled heart sound recordings. Extensive research on different optimizer functions and loss functions has been conducted and the model with the highest stability, best performance, and greatest robustness which minimizes loss, has been selected.

Contents

List of Figures

List of Tables

List of Abbreviations

Sr. No.	Abbreviations	Expanded Form
i	MFCC	Mel Frequency Cepstral Coefficients
ii	TSNE	T-Distributed Stochastic Neighbor Embedding
iii	CNN	Convolutional Neural Networks
iv	PCG	Phonocardiogram
v	CVD	Cardiovascular Diseases

Chapter 1
Introduction

1.1 Description

The end goal of this project is to quickly identify the presence or absence of murmurs from heart sound recordings in a highly cost-effective manner. The sound recordings can be collected from multiple auscultation locations on the body using everyday devices like a smartphone or a digital stethoscope if implemented in a more clinical environment. Murmurs are abnormal sound waves generated due to the turbulent blood flow in cardiovascular structures and can be closely associated with specific cardiac diseases like septal defects, ductus arteriosus closure defects in newborns and defective cardiac valves. In this document, the authors have trained a deep learning model using convolutional neural networks (CNN) on a heart recordings dataset containing over 3000 data samples.

The preliminary detection of cardiovascular diseases (CVD) can be achieved through cardiac auscultation, a non-invasive procedure, using phonocardiograms (PCGs). With the recording obtained from this procedure, a deep learning model trained to detect the presence of heart murmurs from heart sound recordings can facilitate the identification of people with acquired or congenital heart diseases. Technology for observing cardiac activity has improved; however, the authors want to build an easy and accessible solution for all.

The authors have experimented with different audio processing and pre-processing models to achieve the highest possible prediction accuracy. A detailed analysis and description of the methodologies have been done to aid future researchers.

1.2 Problem Formulation

There is always a lot of scope in the field of medical diagnosis using advanced machine learning techniques. One such domain is the classification of heart sounds for murmurs from phonocardiogram recordings. The traditional method for the diagnosis of heart abnormalities is the ECG, but with advancements in non-invasive imaging technologies, detection of such abnormalities has become a lot smoother and pain-free for the patient. Yet, a lot of people still do not undergo regular heart check-ups. Only when they start experiencing enhanced heart-related symptoms do they go for a check-up at the physician. Many cases of heart attacks were reported with many pre-symptoms and the chances of a patient's death are usually very high in cases of the first heart attack. This can result in scenarios that would have been easily avoided through a routine heart check-up but not done due to lack of time or money.

Technology can aid in addressing the above issue. The Phonocardiogram (PCG) is the method of retrieval of the heart's sound. This sound can be captured with a simple stethoscope. In this work, we are proposing a deep learning model that has the potential to detect heart abnormalities by finding murmurs in heart sounds. We are trying to build a cost-effective early detection of heart diseases. One of the aims is to provide easy access to health services in geographically large areas with a lack of infrastructure and cardiology specialists—the potential of automated approaches that provide accessible pre-screening for less-resourced populations. The main goal is to identify the presence or absence of murmurs from multiple auscultation locations on the body using a digital stethoscope or an everyday device like a smartphone. This can help us make a real-world impact by deploying a model that helps with this detection with a low error rate.

1.3 Motivation

A sudden rise in the medical cases of cardiovascular diseases has been observed in recent years, where the affected people are often young and healthy individuals. Upon researching further about this, we found out that the adumbration of these problems lay in the detection of murmurs in a heartbeat. During our research, we also discovered that the primary issue with the diagnosis was the high charges required for a cardiologist's consultation, which is not available at readily accessible centers to people living far from big cities. We aim to build a product that would eliminate these costs and make preliminary diagnoses available to everyone everywhere. This also allays a medical professional's efforts by providing a tool that identifies heart murmurs reliably and accurately, speeding up the cause-determination process in life-and-death situations.

1.4 Proposed Solution

An audio classification problem is typically solved using a Convolutional Neural Network (CNN). This project involved the training of a Sequential CNN on the given dataset to classify the heart sounds as either containing heart murmurs or not containing them. Dimensionality reduction and other analysis techniques, when applied to the dataset, help build a network that trains on the features present in the dataset and outputs a simple yet healthily accurate model. Experimenting with different parameters, loss functions, and optimizers boosts the model's accuracy. The model proposed takes input data in the form of sound files of heartbeats and gives a binary output as to whether the heartbeat does or does not possess a murmur. The project aims at helping end users themselves carry out a test of their heartbeat to check for murmur abnormalities before consulting a doctor. It eliminates the need for the patient to be present at a technologically advanced facility to have murmurs detected.

1.5 Scope of the Project

The project is intended to be deployed globally and can be utilized by people of all ages and demographics. This will allow people to do preliminary testing for heart murmurs by themselves using just a recording of their heart sounds and will therefore reduce the load on healthcare centers. This project will not give a complete diagnosis of the disease, and the patient will still have to visit a doctor for proper diagnosis and treatment in case the results come back abnormal. This project can also be implemented in a clinical setup to help the doctors with faster diagnosis and can be used with a digital stethoscope for clearer recordings.

Chapter 2

Review of Literature

“Phonocardiogram Classification Using 1-Dimensional Inception Time Convolutional Neural Networks [1]” presents a method for classifying phonocardiograms (PCGs), which are recordings of heart sounds using a type of convolutional neural network (CNN) called 1-dimensional inception time CNN’s. The authors aim to classify the heart sound sample into three categories – the presence of murmurs, absence of murmurs, or unclear decision- and predict normal or abnormal clinical outcomes of phonocardiogram (PCG) recordings using Machine Learning algorithms. The Deep Learning technique used for heart sound sample classification is 1 Dimensional Convolutional Neural Network (CNN) which is trained on the biggest pediatric heart sample dataset based on data collected from 1568 people. The PCG signals were downsampled to 100Hz from 4000Hz with additional padding added to make all the signals of uniform length. Then the recording was re-labeled so that labels were according to individual recordings labeled the same as the original patient label. For classification, two models were trained – one for detecting murmurs which were trained using weighted categorical cross-entropy, and the other classifier for clinical outcome was trained using weighted binary cross-entropy. The main advantage of using CNN is that it can detect the presence of abnormaltieis in PCG recordings without knowing the auscultation location. A weighted accuracy of 49.7% was achieved in classifying murmurs, and 71.3% was achieved in classifying clinical outcomes.

In “Deep Learning Algorithm for Automated Cardiac Murmur Detection via a Digital Stethoscope Platform [2]”, the authors begin by describing how using a stethoscope can help detect cardiovascular diseases but requires a skilled individual to use it. They believe the same detection can be done using Deep Learning models. To do so, they collected almost 40.000 recordings from 5878 patients which were labeled and reviewed by three cardiologists. The neural network algorithm to classify the murmurs used the ResNet deep convolutional neural network architecture. The input signal was filtered through an eight-order Butterworth high-pass filter at 30Hz to downsample the recoding to 2000Hz. The network consisted of 34 layers, each one a 1-D convolution layer. Initialization was done using random weights and optimization was done using ADAM optimizer. The downside of this approach was that the data was only collected from a few heart clinics and could be missing some very important diagnoses. Plus, the algorithm doesn’t work with poor-quality of recordings and has been tested in ideal scenarios only and could have different results in the real world. But one advantage of the data collected in this paper is that it was all recorded using the Eko Mobile app connected to the Eko Core/Duo Stethoscope, making the real-life data sample very close to the trained samples. Another thing noticed in the model training phase was that if Grade 1 murmurs were exempted from the data, the algorithm’s accuracy showed a considerable increase.

In “Deep Convolutional Neural Networks for Heart Sound Segmentation [3]”, the authors have implemented a CNN-trained model for heart murmur detection and have achieved amazing accuracy of 93% by making use of Hidden Markov Models and Hidden Semi-Markov Models to maintain sequence along with the CNN model. Unlike other approaches, this algorithm has no restriction to just performing binary classification on S1, and S2 sounds but rather performs the segmentation of the entire heart sound, therefore automatically providing the boundaries and locations of fundamental heart sounds. During pre-processing of the data, the signals are passed through filters having cut-off frequencies 25Hz and 400Hz, and spike removal is done. The CNN implementation makes use of the UNet CNN with Relu activation and weights. Temporal Modelling is done, and four different strategies are compared. Finally, 10-fold cross-validation is done on the dataset, comprising of 792 recordings. Overall, the approach presented is a new one and has shown great results, but the accuracy cannot be fully relied upon as the dataset used is very small and, therefore, might not cover all scenarios. Real-world accuracy will be quite different from the claimed 92

“Two-stage Classification for Detecting Murmurs from Phonocardiograms Using Deep and Expert Features [4]” describes the creation of an ensemble classifier using Deep learning to detect heart murmurs and clinical abnormalities from phonocardiogram (PCG) recordings of multiple auscultation location. Recordings were first passed through gradient boosting algorithms to detect unknown samples, mainly due to low-quality recordings. Then two models were trained – a gradient boosting ML model and an ensemble of convolutional neural networks (CNN)- using Mel-frequency cepstral coefficients (MFCC) and time-frequency features as inputs, respectively. The ensemble consisted of three classifiers, one to detect murmurs and two to categorize them. Both the models were connected using Logistic Regression, and the CNN network was trained using five layers. S1 and S2 segments were treated separately to improve accuracy, meaning S1 can be positive and S2 can be negative for a single recording. The model was strained on the Circor dataset with 5282 recordings and had a weighted accuracy of 75% using five-fold cross-validation.

In “Automatic Murmur Grading from Phonocardiogram [5]’, murmur grade was considered for measuring murmur intensity. This measure is deeply connected with the clinical conditions of the patient. For the representation of PCG recording, the mel spectrogram was used. These recordings were classified using 15 convolutional residual neural networks with channel-wise attention techniques. They used 3456 PCG recordings from 1007 patients. The CirCor DigiScope PCG dataset used is a portion of a larger dataset. The PCGs were captured for the aortic, pulmonary, tricuspid, and mitral valves—four well-known auscultation sites. Preprocessing techniques included the Mel filter bank, wavelet transformation, and Kalman-based spectro-temporal estimation techniques. The number of trainable parameters of the network is decreased when SepConv layers are used in place of conventional convolutional layers and used Squeeze-and-excitation layer to recalibrate feature responses in an adaptive manner. The authors used 2-D deep convolutional neural networks and obtained an unweighted average of sensitivities as 86.3%. The F1 score was 81.6% and they used a 10-fold cross-validation technique which resulted in a score of 90%. The drawbacks faced were relatively high uncertainty of the network and the algorithm was tested using systolic murmurs only. Soft murmurs were the hardest to accurately identify. Also patients used in this study were of limited population. The data was annotated by only one expert.

The “Joint Heart Sounds Segmentation and Murmur Detection with Masked Loss Function [6]” paper has combined the separated steps for accurate detection of abnormal cardiac sounds. They have proposed an approach that combines the identification of cardiac murmurs with the segmentation of stethoscope recordings into individual heart cycles using a single neural network model that is able to detect the individual heartbeat cycle. Then they forecasted the occurrence of cardiac murmurs by segmenting these into the four heartbeat periods. A loss function was used, which incorporates relationships between the different types of outputs and employs masking if labels are missing. The proposed model is a modified RNN that is a Recurrent Neural Network with the final layer divided into two groups, the first contains five possible classes, and the second describes the absence and presence of murmurs. One thing is that they support negative examples, such as recordings that lack heart sounds. They have used three baseline models and obtained an F1 score of 83.9%. Their method outperformed the best reference algorithm of Maknickas et al. by 7.6 percentage points. But they failed in only two cases where the null hypothesis with the same distributions were rejected. The one and only class in which the performance was inadequate was the one representing a lack of audible heartbeat. By increasing the sample size of recordings without a heartbeat in the training dataset, it will be possible to fix the issue of the method with hand-corrected augmentations labeling just 8.7% of audio frames as having ”no heart.”

In “Intelligent Diagnosis of Heart Murmurs in Children with Congenital Heart Disease [7]” paper, the authors proposed a technique for accurately diagnosing pediatric CHD murmurs. They have recorded 86 children’s PCG signals, including 24 with normal heart sounds and 62 with CHD murmurs. To find the first and second heart sounds, a discrete wavelet transform and Hadamard product based segmentation technique was used. Following segmentation, ten features unique to CHD murmurs were recovered and used as the classifier’s input. They created a classification method for CHD murmurs using 86 Artificial Neural Network classifiers. They also used several techniques like PCG Decomposition and Recombination & Envelope Extraction. Even in the cases of children with moderate and severe CHD illnesses, whose murmurs block S1 or S2, a new segmentation technique that makes use of both the Hadamard product and the discrete wavelet transform was successful in locating S1 and S2. This criteria helped in avoiding the problem of S1 and S2 displacement. They helped in encouraging the development of online diagnostic tools for children’s CHD and cardiac auscultation. The accuracy obtained was 93% along with 93.5% sensitivity. Low specificity, indicating that the current approach could detect CHD murmurs accurately but that increased the likelihood of false positives. Loud noises due to children crying and moving during the recording could not be eliminated by the denoising technique utilized in this study. An expert recorded PCG signals at one of five auscultation sites. These drawbacks led to areas for further study.

The paper “Murmur clinic: validation of a new model for detecting heart valve disease [8]” wanted to see if auscultation or a point-of-care scan could replace traditional echocardiography. They performed the experiment between 21 July 2015 and 9 May 2017 by a scientist, and between 1 October 2013 and 31 December 2014 by a cardiologist. They used a vivid seven system for it. With the patient at a 45-degree angle, auscultation was done at the four standard spots and in the intermediate positions. If there was no murmur or a likely benign systolic flow murmur, auscultation was classified as normal. If there was a loud murmur, it was classified as abnormal. The scientist’s auscultation was 83% sensitive, whereas the cardiologist’s was 91% sensitive. Comparing the murmur clinic model with 52 TTE and 123 point-of-care scans to the traditional model with 175 TTE, which cost £14303, the murmur clinic model would have cost £7600. This saved them £6703 and scanning time to 82 hours. But 91 patients missed their scheduled appointments, which made recruitment difficult. In addition, due to general departmental requirements, two scientists could not be released for each murmur clinic.

In “Convolutional neural network approach for heart MurMur detection, [9]” the authors have worked on a dataset of 4000 samples of 1-second-long time frequencies of heart auscultation signals obtained from continuous wavelet transforms presented in the George Moody Physionet Challenge 2022. In the data preprocessing step, the authors realized that the precision of the time representation of the highest frequency components could be much less than the initial 1/4000 of a second. Hence, the authors reduced the length of the dataset to 91. This saved them a lot of computational resources. All recordings in the dataset contained corrupt or noisy episodes, so the authors used only those with the lowest entropy. This data was trained with the convolutional neural network architecture designed to estimate whether murmurs were either present or absent. The architecture had 15 layers used in preprocessing design. Three two-dimensional convolution layers with sliding convolutional filters. Two average pool layers were present to perform down-sampling. This straightforward architecture proposed was optimal, gave the best performance, and had a short training time. The accuracy obtained on the validation set was only 59.9%. The authors avoided signal segmentation as they concluded that the neural network would learn about the signal as it got trained. While the results obtained via this method were optimal, no tests were done based on it. Since the entire dataset wasn’t preprocessed, the authors don’t have the performance of their architecture on the whole dataset.

In “A Fusion of Handcrafted Feature-Based and Deep Learning Classifiers for Heart Murmur Detection[10],” the authors have proposed two approaches to solve the problem. The first being the deep learning classifier, which used Mel-frequency cepstrum coefficients (MFCC) & bidirectional long-short-term memory (LSTM) from raw Paediatric Phonocardiograms signals. In this approach, the authors worked on a dataset of PCG signals in 10 seconds segments with additional zero padding to make the signals of ten seconds where required. The architecture involved a base convolutional neural network containing convolution kernels of size eight, and four consecutives, with Rectified Linear Unit (RELU) activation functions. Variations included adding an attention mechanism or Bi-LSTM. The feature-based technique was used in the first approach, in which the authors preprocessed the data by performing non-negative matrix factorization to remove noise from PCG signals, then based on the full and segmented recordings, extracted the features to eliminate noise, followed by feature selection. Based on 10-fold cross-validation, the top hundred features were considered and tested. The deep learning method based on neural networks outperformed the feature-based method. The merging of the two approaches improved the results of 10-fold cross-validation. The authors obtained accuracies of 62.6% in the deep learning approach and 46% in the feature-based method. The authors demonstrated high variation between cross-validation and hidden validation sets due to under-fitting.

In “Heart Murmur Detection using Supervised Machine Learning, [11]” the authors used the “heartbeat sound” dataset from Kaggle containing 832 audio files, which they classified into six classes. For faster extraction and encoding, they limited the categories of encoding to “artifact,” “normal,” and “murmur.” The proposed architecture included the usage of simple LSTMs with gated units in its hidden layer to help decide what information was to be transferred to the next unit. The model’s simplicity gave the following results: a 70% training accuracy, a 68% testing accuracy, and a 70% validation accuracy.

In “Heart Murmur Detection and Clinical Outcome Prediction using Multilayer Perceptron Classifier, [12]” the authors have used a dataset containing 3163 PCG recordings from 942 patients. To preprocess the dataset, they applied a fifth-order Butterworth band-pass filter which is used in the range of 20-500 Hz for all PCG recordings. Sequential Forward Floating (SFFS) was used to select features with the best performance. To train the model, the authors proposed a multilayer perceptron-based classification model that classified heart murmurs as being “present”, “unknown” or “absent”. The MLP model was utilized by the authors, which uses the ”Adam” solver for weight optimisation and has four hidden layers with 256 neurons, 128 neurons, 64 neurons, and 32 neurons. The authors have discussed the importance of the Frequency and Time-Frequency domains over the Time domain in heart murmur detection. They have concluded that the former two might provide more accurate results than the latter. This model gave a relatively lower accuracy of 57.69%.

In “Heart Sound Segmentation Using Bidirectional LSTMs With Attention [13]”, the authors have worked on a dataset comprising animal and human recordings. Attention based learning was used for segmenting the Phonocardiogram signal obtained in the dataset. To train the model, the authors used a Recurrent Neural Network (RNN) as they observed that the combination learned well from irregular & noisy PCG recordings. The encoder LSTMs’ hidden state dimension was experimentally assessed and adjusted to 80 units. Empirical evaluation with different feature combinations, including MFCCs, entropy of wavelets, homomorphic envelope features, power spectral density and Hilbert envelope, was performed for signal segmentation. The system was time efficient and did not require a special GPU. It only used 17K trainable parameters. According to the authors, the suggested framework is not limited to heart sound recordings and could be used to analyze any one-dimensional signal, such as lung sounds, electrocardiograms, and electroencephalograms. The authors have also evaluated and compared the proposed technique with various other networks, including DNNs, CNNs, GRNNs, LSTMs, and Bi-LSTMs. This model gave high accuracies of above 90% obtained.

Chapter 3

System Analysis

3.1 Functional Requirements

The user should be able to receive the output by performing minimum steps.

1. Requirement 1: A simple page input page with an age field, weight field, height field, gender field, Upload File (button), and Submit (button).
2. Requirement 2: An Output page where the result of the audio file analysis is displayed.
3. Requirement 3: A list of suggested doctor nearby in murmur is detected.
4. Requirement 4: Tracking of weekly, monthly or yearly checkups if needed.
5. Requirement 5: External interfaces like consideration of some other problems detected along with murmurs and if it is causing the growth of the murmurs.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

1. The website interface works best on modern-day web browsers such as Chrome, Mozilla, and Safari.
2. Requires a proper internet connection in order to upload the recording file. A Slow internet connection may cause the time to get the results, but the results would remain unaffected.
3. An examiner of the report needed for properly communicating about the report to the patient.
4. A recording application needed for recording the audio file of the heart beats.
5. A noise proof or silent room is recommended to not cause any noise in the audio file.

3.2.2 Safety Requirements

1. The user must ensure that the recording is done using clean, hygienic equipment and that no harm comes to him/her while recording.
2. The user needs to ensure no other confidential content is spoken in the background while recording.

3.2.3 Security Requirements

1. Since the users will be filling out the form and uploading the recordings themselves, they consent to the use of this information. This information will be kept private.
2. A trained examiner or consultant needed for the process of reading the report in detail.

3.2.4 Software Quality Attributes

1. If the internet service gets disrupted after giving the data and audio files input, the report will be generated and stored.
2. After restoring the connection, the user has the option to see the previously generated report or create a new one.
3. The web interface is easy to handle even for a complete layman.
4. A survey form after the whole process for user satisfaction and this can help update the application for more ease.

3.3 Business Rules

1. The entire software would be under copyright. Any kind of reproduction as a part or whole is punishable under Copyright Infringement Law.

3.4 Specific Requirements

3.4.1 Hardware Requirements

1. Mobile Operation
    - Android System (Marshmallow 6.0.1 and above) with a microphone.
    - iOS System (Version 10.3.5 and above) with a microphone.
2. Desktop Operation
    - Windows Operating System (Windows 8.1 and above).
    - MAC OS (MAC OS X 10.6 Snow Leopard and above).

Note: There is no additional hardware required.

3.4.2 Software Requirements

The Machine Learning and Deep Learning models are coded in Python programming language. NumPy, pandas, sci-kit learn, and TensorFlow are some libraries used to preprocess the data and train the models. We have also used Python wavelets for signal processing. The user interface is built on React, a JavaScript Framework. The backend is built on Django, a Python backend framework. Relational database PostgreSQL is used for storing the data and relevant information.

3.5 Use-case Diagram & Description

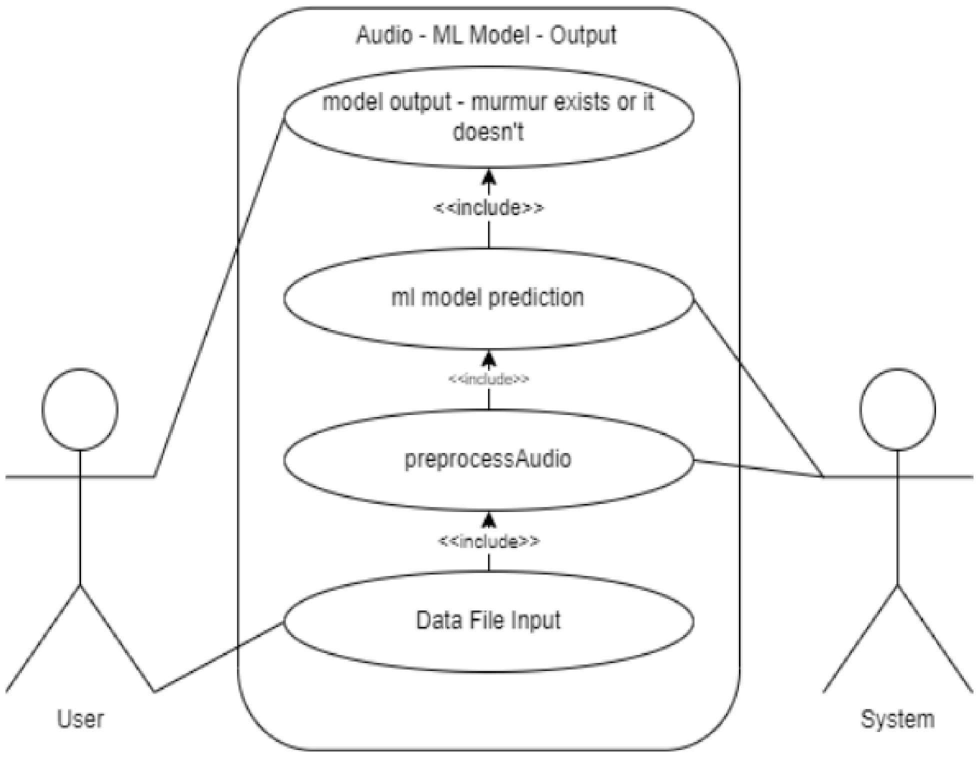


Figure 3.1: Use Case

The Figure 3.1 shows the use case diagram for the proposed system. It can be seen that the user has very little interaction with the processing of the data by the ML model. The user must input the data, which contains audio files and certain associated metadata. Rest of the functions are performed by the system, that is the ML model, to give the user output in the form of whether a murmur exists or it does not. On receiving the data, the model is used to predict whether heart murmurs exist in the provided data or not.

Chapter 4

Analysis Modeling

4.1 Data Modeling

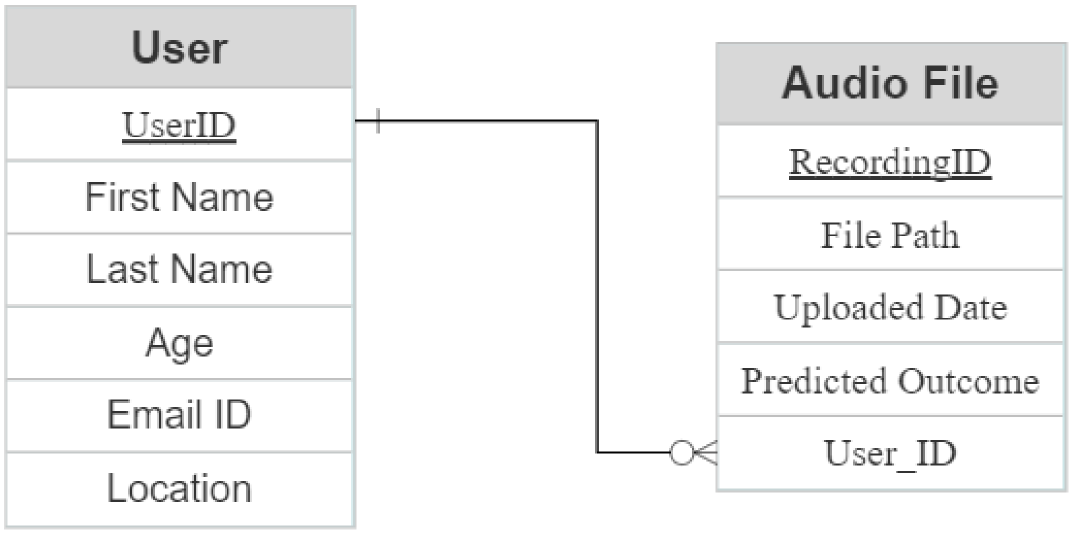


Figure 4.1: Entity Relationship Diagram

The database consists of 2 entities User and Audio Files consisting of the following relation - One user can submit N number of audio files but one audio file can belong to only one user.

4.2 Activity Diagrams

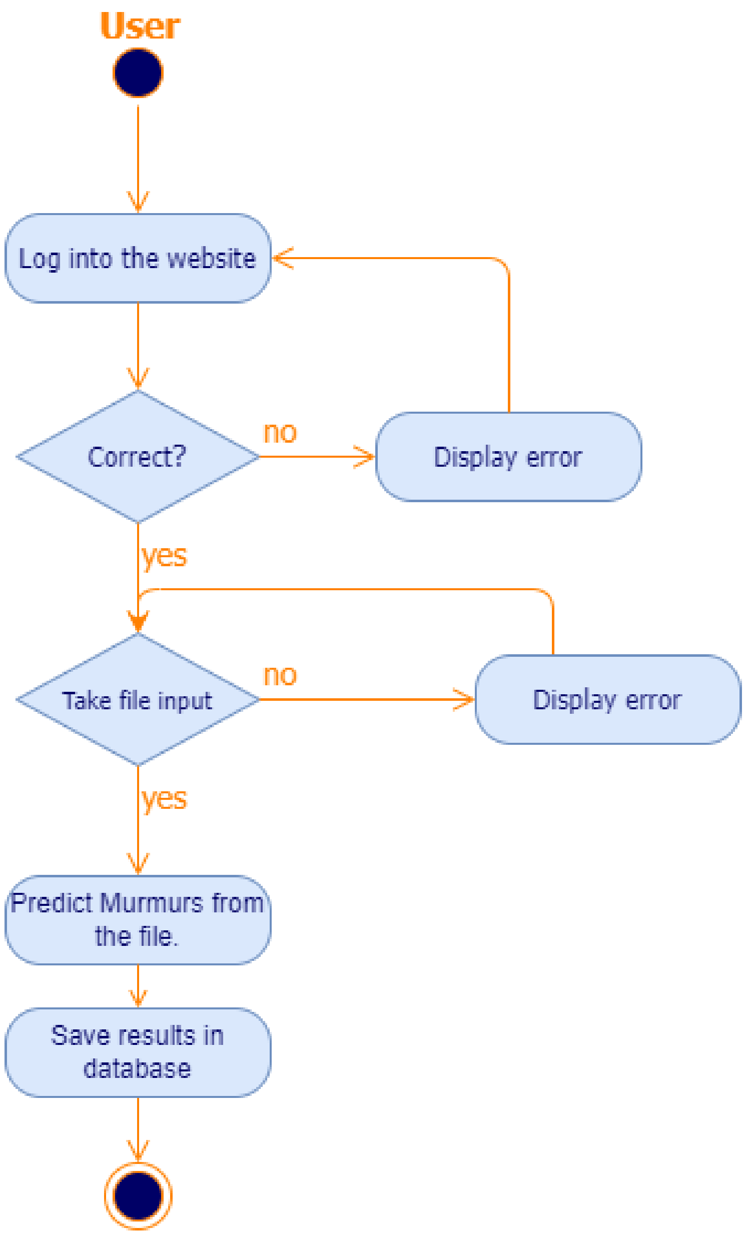


Figure 4.2: Activity Diagram

4.3 Functional Modeling

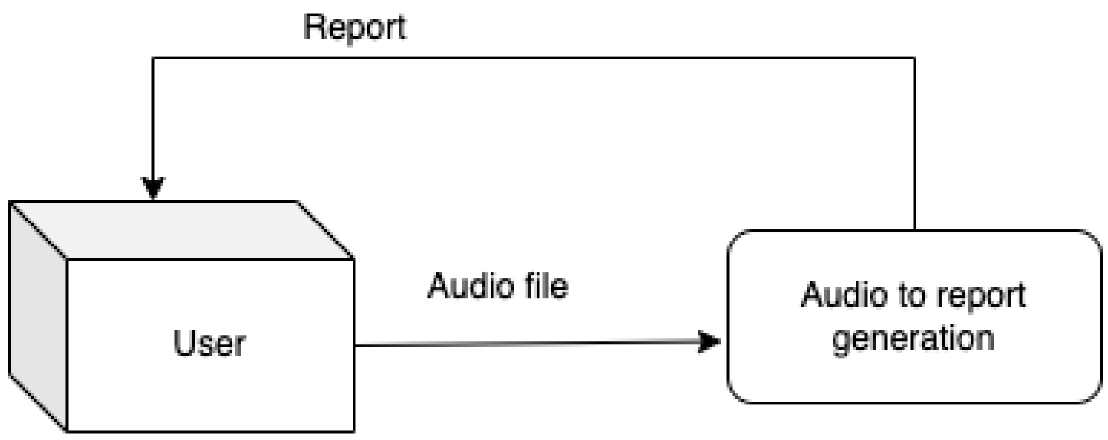


Figure 4.3: DFD Level 0

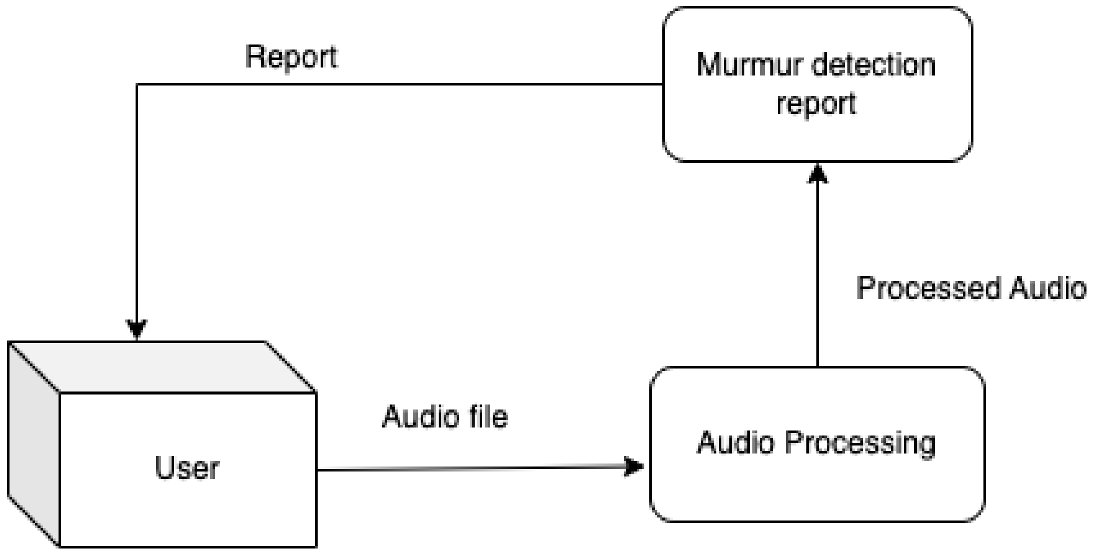


Figure 4.4: DFD Level 1

To illustrate the flow of data through the system pertinent to corresponding functionalities, data flow diagrams are used. Level 0 diagram (Figure 4.3) is the context diagram, which manifests the entire system as a single process. Level 1 diagram (Figure 4.4) is broken down into multiple processes. Only the cardinal or primary functions are discussed. Level 2 diagram (Figure 4.5) elucidates the level 1 diagram in more depth, providing a comprehensive overview and details of the system's functioning.

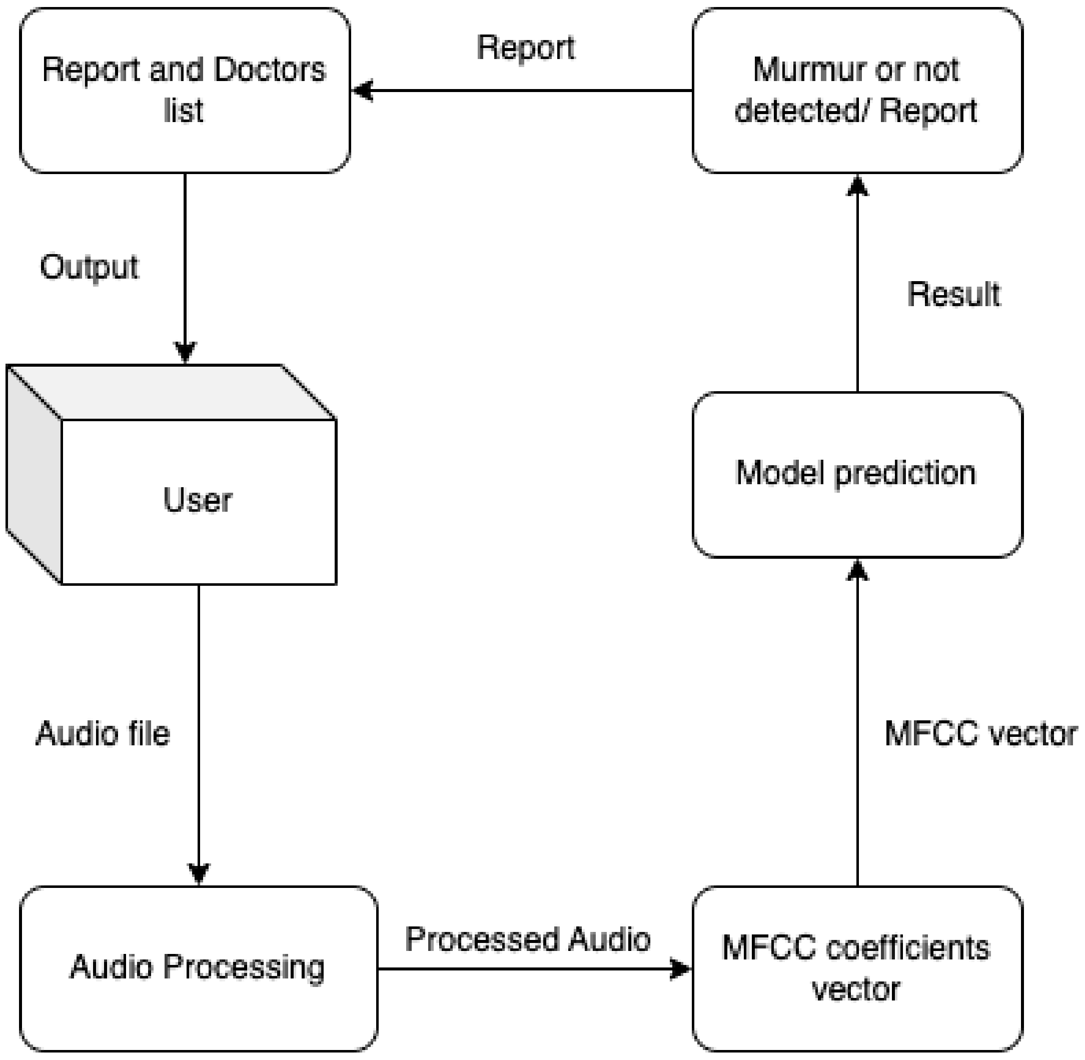


Figure 4.5: DFD Level 2

4.4 Timeline Chart

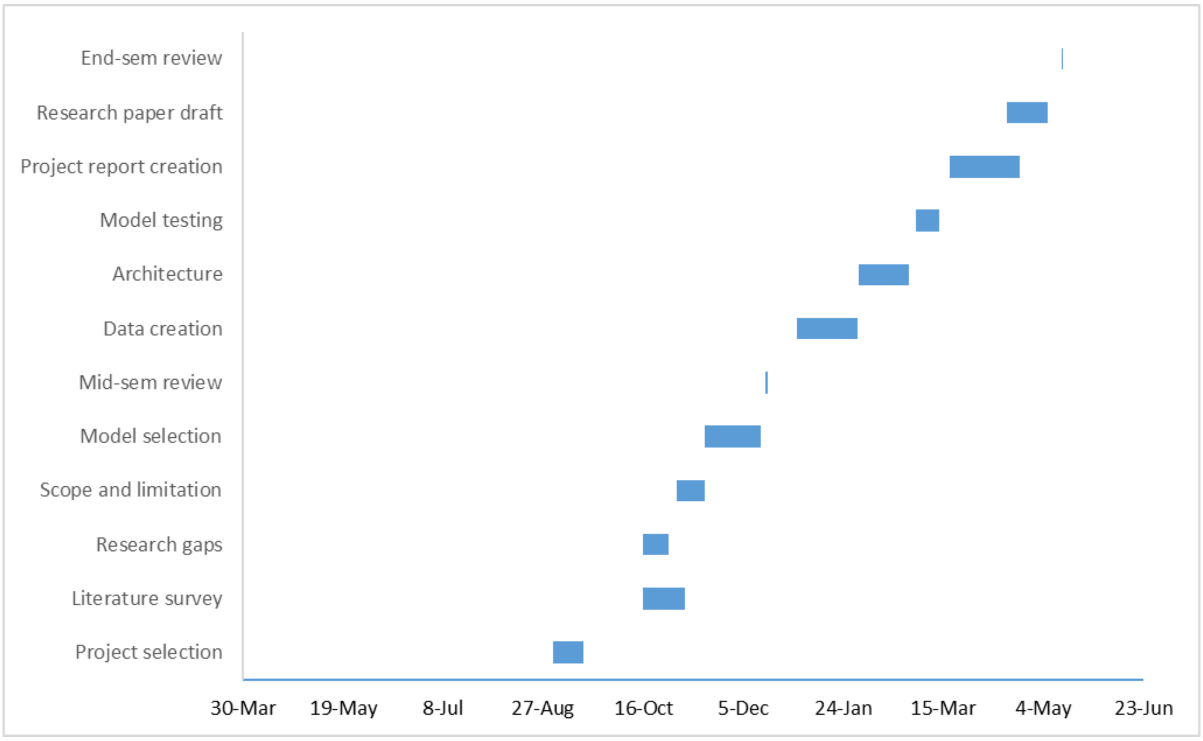


Figure 4.6: Gantt Chart

Task	Date of initiation (dd/mm/yyy)	Duration of activity (Days)
End-sem review	13-05-2023	1
Research paper draft	16-04-2023	20
Project report creation	18-03-2023	35
Model testing	01-03-2023	12
Architecture	01-02-2023	25
Data creation	01-01-2023	30
Mid-sem review	16-12-2022	1
Model selection	16-11-2022	28
Scope and limitation	02-11-2022	14
Research gaps	16-10-2022	13
Literature survey	16-10-2022	21
Project selection	01-09-2022	15

Table 4.1: Timeline of the project

Chapter 5

Design

5.1 Architecture Design

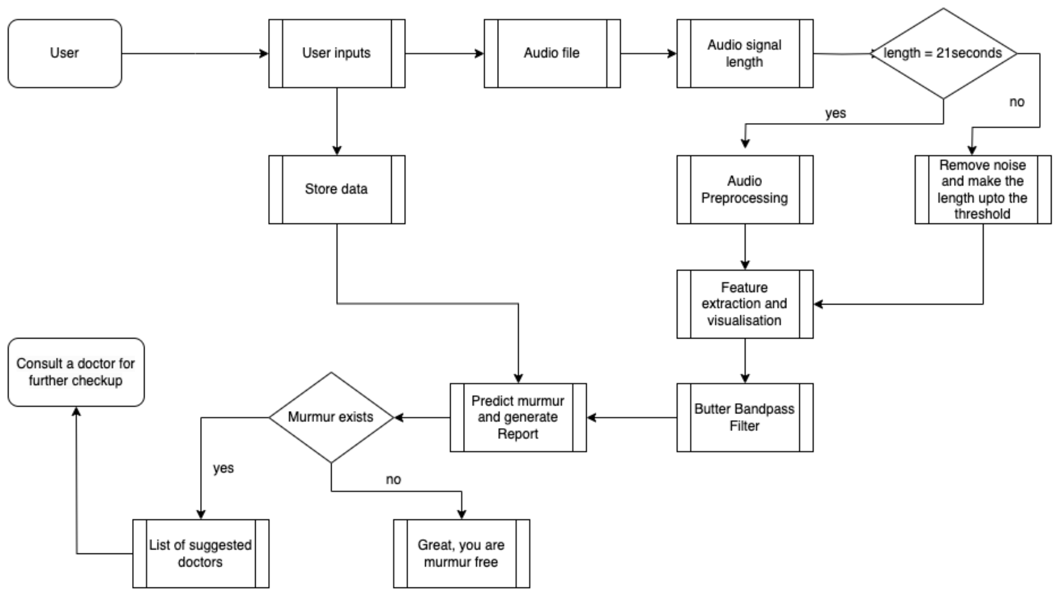

User
User inputs
Audio file
Audio signal length
length = 21seconds
yes
no
Store data
Audio Preprocessing
Remove noise and make the length upto the threshold
Feature extraction and visualisation
Consult a doctor for further checkup
Murmur exists
Predict murmur and generate Report
Butter Bandpass Filter
yes
no
List of suggested doctors
Great, you are murmur free

Figure 5.1: System Architecture

The heart murmur classification system works in the following steps:

1. User uploads the audio signal file along with user details like first name, last name, age, email id, location or address and this generates a user id in the database.
2. This data is stored in a Postgresql database and is used further in report generation.
3. Our model takes a fixed audio length, hence we first check if the audio length of the file is 21 seconds or not. If it is 21 seconds, then we start with audio preprocessing. If not then we remove the noise and tailor the length upto the threshold size and then start the audio preprocessing.
4. In audio preprocessing, we extract the features of the audio signal and visualise the signal features.
5. We have different computations for different types of features of audio signals.
6. We use the butter bandpass filter in audio preprocessing to define the low and high pass filters for the audio signal.
7. The coefficient vector is generated and passed in the model for prediction.

8. We get the prediction whether the murmur is found or not. If it is not found then the patient is murmur free and healthy. If it is found then the user is suggested to consult a doctor.
9. For consultation, a user is suggested a list of doctors nearby as per the users location.
10. The user can visit a doctor for further checkups.
11. For future architectural system, we can filter the doctors list according to the location of user, the speciality of doctor, the fame of the doctor or the number of experience of the doctor.
12. This can help us create a database of doctors and with the help of AI we can automate the appointments for the user.
13. The user activity can help us recommend the most visited doctor and help doctor analyse the number of users and their activities or their checkups.

5.2 User Interface Design

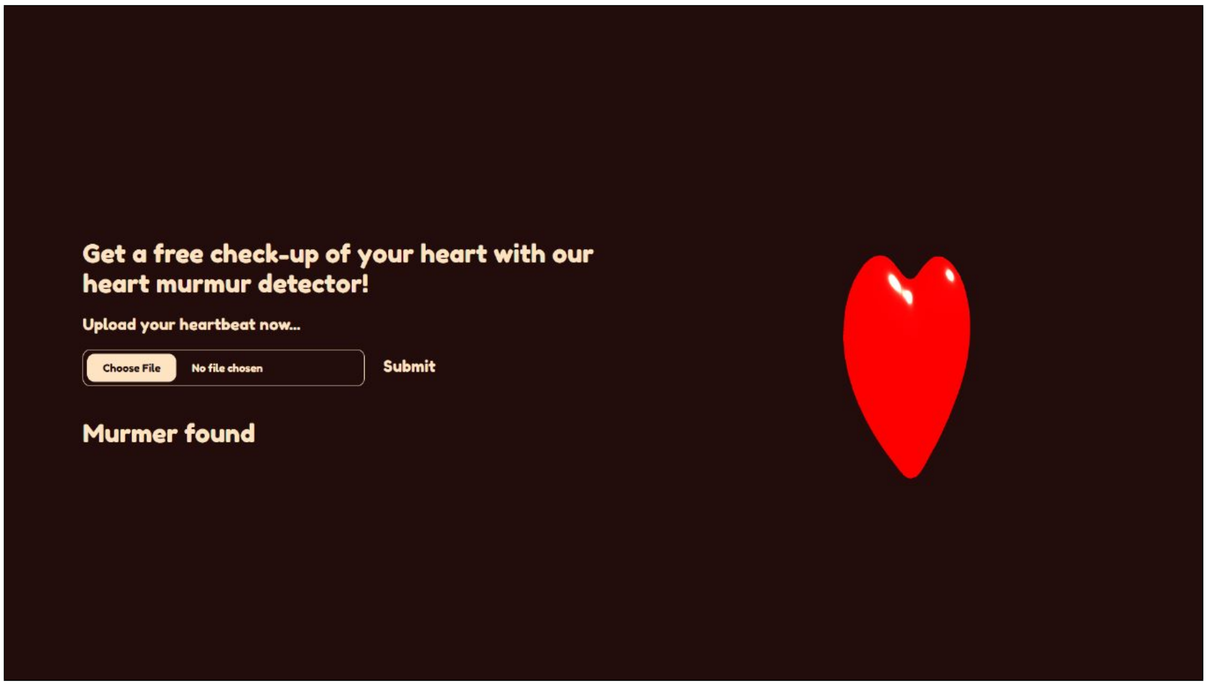

Get a free check-up of your heart with our heart murmur detector!
Upload your heartbeat now...
Choose File
No file chosen
Submit
Murmer found

Figure 5.2: Minimalistic web interface for users

The heart murmur classification system User Interface is as follows:

1. We have used a python framework Django for backend development since it is a very flexible framework and helps in API formation of machine learning models very effortlessly.
2. For frontend, we have used a javascript framework named React which is versatile and has high quality user interface.
3. User uploads the file in the choose file tab, if the file is not chosen or empty, it notifies the user that file is not found after clicking on the submit button.
4. The uploaded file gets stored in a database, here we have used Postgresql database.
5. The backend API processes the file and runs the model to give the prediction of whether the murmur is found or not.
6. If a murmur is found, then it should that the user must consult a doctor. If murmur is not found then the user is murmur free and healthy.
7. The heart gif was created using Three.js for graphics and GLTF model.

Chapter 6

Implementation

6.1 Model Created

6.1.1 Preprocessing

Once the dataset was obtained, analysis and preprocessing were the first steps toward the model classification. Preprocessing and dataset analysis involved designing a Homomorphic envelope [14] using the Butterworth filter [15] based on the Hilbert transform [16], a subset of the process of sound segmentation using homomorphic filtering [24] to obtain a sampled version of the signal, which helped indicate the heart rate of the patient - which is a significant aspect [22] that must be checked while dealing with datasets of heart sounds. This was followed by visualizing the Mel Frequency Cepstral Coefficients [25], the MFCCs [26], a feature visualization technique for audio sound data. The MFCCs and their scaled representations help visualize the data as a spectrum of time versus frequencies. The T-Distributed Stochastic Neighbor Embeddings [27], or the TSNE, is a dimensionality reduction technique that helps visualize the data in two dimensions, either possessing a heart murmur or not possessing them. This was done based on feature extraction where the several features present in the dataset included “mfcc,” “spamp,” “wavelet,” “mfcc-dwt,” or simply raw data. For the “mfcc” and “mfcc-dwt” features, the data was resampled to a frequency of 22050 kHz, the standard sampling rate for low bit rates of MPEG Audio Layer 3. The higher frequency contents are missed, but the intention is resampling to strip the files to their bare essentials. The Python library used to build the Butterworth filter [21] was Scipy with the “butter” function of the third order. The Hilbert transform [23] was built using the “hilbert” library from Scipy. While extracting features, the Butterworth filter built had the following parameters:

- Order of the Butterworth filter: 5 works best when multiple single-order filters are cascaded to form a fifth-order Butterworth filter.
- Lowcut and highcut frequencies: Array of [25, 400] frequencies in kHz, combined with the 5th order Butterworth filter.
- BType: Band - the band filter.

If the feature was “spamp,” the one-dimensional Discrete Fourier Transform (DFT) modulus was obtained to obtain straightforward numerical values for real inputs using the np.fft.rfft function of the Numpy library. A discrete wavelet transform with a ‘db1’ wavelet was output along with an MFCC resampled to 22 kHz for the “mfcc-dwt” feature. The feature-extracted dataset was then treated to the t-distributed stochastic embedding technique to obtain a scatterplot containing the various files represented as plots that either possessed murmurs or didn’t.

6.1.2 The AlexNet

This set the stage for building the deep-learning model [36] for the classification. To work with large audio datasets for binary classification, a simple AlexNet [20] provides a baseline for the deep learning model. The original AlexNet [35] comprises five convolutional layers and 3 fully connected layers. It was introduced for the task of image classification but was later transposed to audio classification as well. The AlexNet's specialty lay in its usage of the Rectified Linear Unit (ReLu) function as the activation function as opposed to the sigmoid activation function [19], which strips only the positive part from the entire argument. The ReLu function [42] is performed on the output of the preceding convolutional layer. ReLu activation ensures that positive-valued neurons are maintained while the negative-valued ones are zeroed. ReLu removes the associativity of successive convolutional layers and introduces non-linearity in the network.

Batch Normalization [39] [40] standardizes and normalizes inputs by transforming a batch of input data with a mean of zero and a standard deviation of one, which is also used in the model for the project. However, the original AlexNet employs the LRN or local response normalization [38]. LRN maximizes neighboring neurons' activity. Neighboring neurons refer to neurons with the same spatial position on different feature maps. LRN is not widely utilized because other, more effective normalization methods exist. AlexNet also possesses overlapping pooling layers. Pooling layers in CNNs reflect the general information from the original set of pixels while projecting information within a set of pixels or values in a feature map onto a smaller grid. Finally, the Dropout layers [41] come into the picture. The Dropout layers are present in the AlexNet to prevent overfitting. These layers add a probability to the neurons, which randomly charges the neurons in the process of backpropagation during the feed-forward step. Due to this, each neuron stops depending on neighboring neurons and learns more useful features. It, however, increases the network converging time and, thus, the training time. The AlexNet has the first five convolutional layers arranged sequentially, with a max pooling layer after each convolution layer, followed by three dense, fully connected layers with a Dropout layer succeeding each dense layer, followed by the output layer.

6.2 Working of the project

Initial attempts at constructing the MurmurNet involved modifying the AlexNet as follows:

- The kernel size was set at 2.
- The pool size was set at 2.
- A convolutional layer with 16 as the number of filters or the dimensionality of the output space with a max-pooling layer [37] follows this layer.
- Another convolutional layer with 32 filters follows with another max-pooling layer.
- Three sequential convolutional layers with 64, 128, and 256 filters, respectively, with a third max-pooling layer, are stacked on it.
- Next comes a global average pooling layer. Flatten preserves all values in a tensor while transforming it into a one-dimensional tensor with the sample's dimension added. GlobalAveragePooling2D uses average pooling on the spatial dimensions until all are one and the other dimensions remain unchanged.
- Next comes the dense output layer with softmax activation and a dimensionality of 2 for binary output.
- The dropout layers were struck off because they reduced accuracy and took up more training time.

While this particular model performed well, boasting a 91% validation accuracy, using the adaptive moment estimation (ADAM) optimizer and the stochastic gradient descent (SGD) optimizer also produced an accuracy of 91%, the final model was better optimized, more robust and had greater accuracy. The model worked best with the "binary crossentropy" loss function.

It had a more remarkable resemblance to the AlexNet, which wasn't a requirement but validated the credibility of the MurmurNet, and provided a more expansive exposure to different optimizers with the experimentation providing much better accuracies.

6.2.1 Architecture

The architecture of the final model was as follows:

1. First Convolutional Layer: The first convolutional layer with 96 filters, a kernel size of (11,11), strides of (4,4), the batch normalization layer, and a max pooling layer with a pool size of (2,2) and strides of (2,2) and ReLu activation.

2. Second Convolutional Layer: The second convolutional layer with 256 filters, a kernel size of 2, strides of (1,1), the batch normalization layer, and a max pooling layer with a pool size of (2,2) and strides of (2,2) and ReLu activation.
3. Third Convolutional Layer: The third convolutional layer with 384 filters, a kernel size of (3,3), strides of (1,1), the batch normalization layer, and ReLu activation. Note the absence of a max pooling layer.
4. Fourth Convolutional Layer: The fourth convolutional layer with 384 filters, a kernel size of (3,3), strides of (1,1), the batch normalization layer, and a max pooling layer with a pool size of (2,2) and strides of (2,2), and ReLu activation.
5. Fifth Convolutional Layer: The fifth convolutional layer with 384 filters, a kernel size of (3,3), strides of (1,1), the batch normalization layer, and a max pooling layer with a pool size of (2,2) and strides of (2,2), and ReLu activation.
6. Flatten: The flattening layer connects these convolutional layers to a dense, fully connected layer.
7. First Fully Connected Layer: The first fully connected layer has 4096 units as the dimensionality of the output space, with batch normalization and ReLu activation. A Dropout layer, with a probability of 0.4, was added.
8. Second Fully Connected Layer: Similar to the first fully connected layer, the second fully connected layer has 4096 units as the dimensionality of the output space, with batch normalization and ReLu activation. A Dropout layer, with a probability of 0.4, was added.
9. Third Fully Connected Layer: The third fully connected layer has 1000 units as the dimensionality of the output space, with batch normalization and ReLu activation. A Dropout layer, with a probability of 0.4, was added.
10. Output Layer: The output layer has a dimensionality of 2 for a binary outcome, batch normalization, and the softmax activation layer.

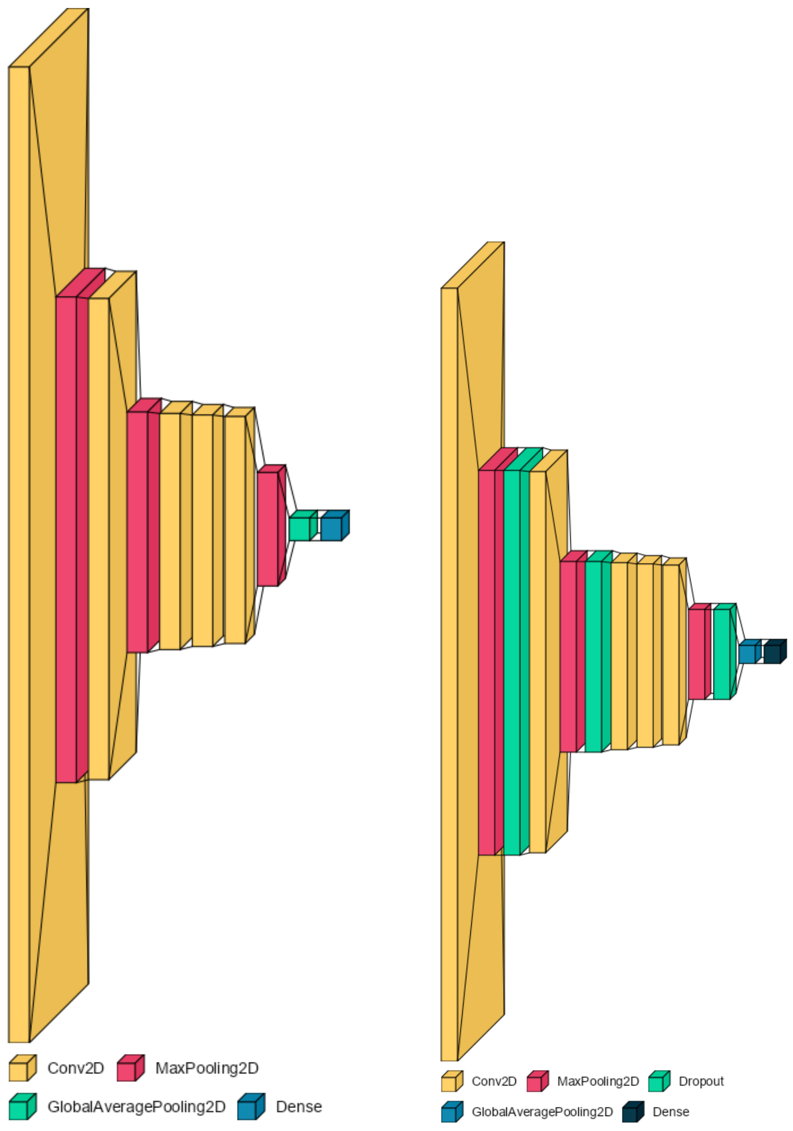


Figure 6.1: Initial Models

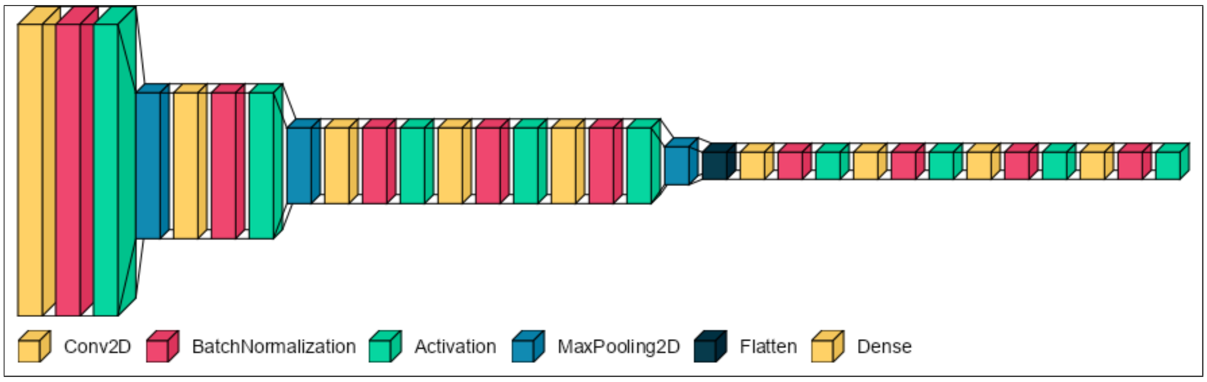


Figure 6.2: Final Model without dropout layers

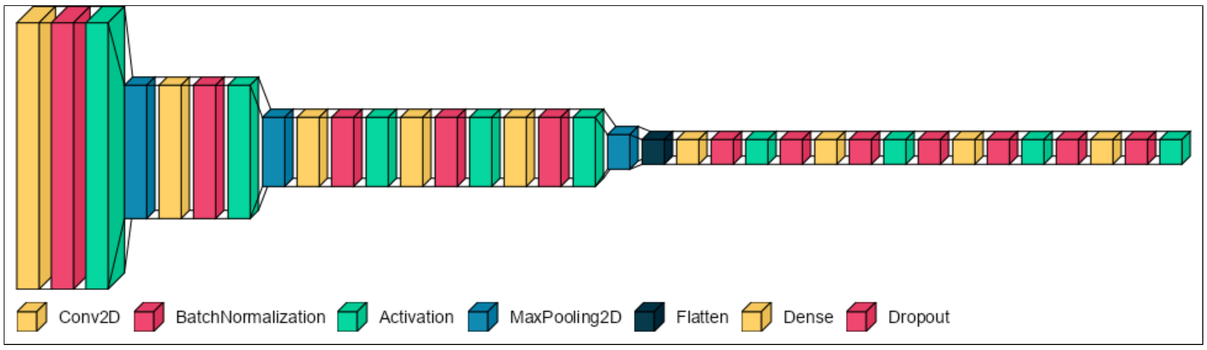


Figure 6.3: Final Model with dropout layers

6.2.2 Python Modules Used

```python
import os
import tqdm
import pywt
import zipfile
import librosa
import warnings
import numpy as np
import pandas as pd
import IPython.display
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,
    precision_recall_fscore_support, confusion_matrix,
    classification_report
from scipy.signal import hilbert, filtfilt, butter, lfilter
from keras.models import Sequential
from keras.callbacks import Callback
from keras.utils import to_categorical, plot_model
from keras.layers import Dense, Conv2D, MaxPooling2D,
    GlobalAveragePooling2D, Dropout, Activation, BatchNormalization

%matplotlib inline
warnings.filterwarnings('ignore')
```

6.2.3 Heart Rate Determination

```python
def heart_rate(y, fs):
    def homomorphic_envelope(y, fs, f_LPF=8, order=3):
        b, a = butter(order, 2 * f_LPF / fs, 'low')
        he = np.exp(filtfilt(b, a, np.log(np.abs(hilbert(y)))))
        return he

    he = homomorphic_envelope(y, fs)
    x = he - np.mean(he);
    corr = np.correlate(x, x, mode='full')
    corr = corr[int(corr.size/2):]
    min_index = int(0.5*fs)
    max_index = int(2*fs)
    index = np.argmax(corr[min_index:max_index])
    true_index = index+min_index
    heartRate = 60/(true_index/fs)
    return heartRate
```

6.2.4 MFCC Visualization

```python
# Visualize MFCCs of a normal and an abnormal audio
def get_mfcc(path, title1, title2):
    """ Get MFCCs of a normal signal and display """
    x, fs = librosa.load(path)
    mfccs = librosa.feature.mfcc(y=x, sr=fs)
    fig, ax = plt.subplots(nrows=1, ncols=2, sharex=True)
    fig.tight_layout(pad=5.0)
    scaled_mfccs = scale(mfccs, axis=1)
    img1 = librosa.display.specshow(mfccs, sr=fs, x_axis='time', ax=
   ax[0])
    fig.colorbar(img1, ax=ax[0])
    ax[0].set(title=title1)
    img2 = librosa.display.specshow(scaled_mfccs, sr=fs, x_axis='time
   ')
    fig.colorbar(img2, ax=ax[1])
    ax[1].set(title=title2)
    return
```

6.2.5 Feature Extraction for TSNE Visualization

```python
def extract_feature(file,feat='mfcc',flt=True,nMFCC=96,flatten=True,mean=False):
    # nMFCC, flatten, mean only apply when feat='mfcc'

    def butter_bandpass_filter(data, fs, lowcut=25, highcut=400, order=5):
        nyq = 0.5*fs
        low = lowcut/nyq
        high = highcut/nyq
        b, a = butter(order, [low, high], btype='band')
        y = lfilter(b, a, data)
        return y

    # Load data and pre-process
    data, rate = librosa.load(file, sr=None)
    data = data[:5*rate]

    if flt:
        data = butter_bandpass_filter(data, rate)

    if feat == 'mfcc':
        if flatten:
            return librosa.feature.mfcc(y=data,sr=rate,n_mfcc=nMFCC).flatten()
        else:
            data = librosa.resample(y=data, orig_sr=rate, target_sr=22050)
            mfcc = librosa.feature.mfcc(y=data, n_mfcc=nMFCC)
            if mean:
                return np.mean(mfcc.T,axis=0)
            else:
                return mfcc

    elif feat == 'spamp':
        return abs(np.fft.rfft(data))

    elif feat == 'wavelet':
        cA, cD = pywt.dwt(data, 'db1')
        return cD

    elif feat == 'mfcc-dwt':
        data = librosa.resample(data, rate, 22050)
        mfcc = np.mean((librosa.feature.mfcc(y=data, n_mfcc=nMFCC)).T
    ,axis=0)
        cA, cD = pywt.dwt(data, 'db1')
        return np.concatenate((mfcc, cD))

    elif feat == 'raw':
        return data

    else:
        raise ValueError('Invalid second argument')
```

6.2.6 Code Snippet For Initial Model With SGD Optimizer

```
# AlexNet
model402 = Sequential([
    Conv2D(16, kernel_size=2, activation='relu', input_shape=(
   xtrain402.shape[1],xtrain402.shape[2],1)),
    MaxPooling2D(pool_size=2),
    Conv2D(32, kernel_size=2, activation='relu'),
    MaxPooling2D(pool_size=2),
    Conv2D(64, kernel_size=2, activation='relu'),
    Conv2D(128, kernel_size=2, activation='relu'),
    Conv2D(256, kernel_size=2, activation='relu'),
    MaxPooling2D(pool_size=2),
    GlobalAveragePooling2D(),
    Dense(2, activation='softmax')
])

# compile
model402.compile(optimizer='SGD',loss='binary_crossentropy',metrics=[
   'accuracy'])
model402.summary()

# train model
metrics402 = Metrics(validation_data=(xvalid402, yvalid402))
history402 = model402.fit(xtrain402, ytrain402, validation_data=(
    xvalid402, yvalid402),
                          epochs=30, callbacks=[metrics402], verbose=2)

plot_history(history402, 'CNN-402')
plot_metrics(metrics402, 'CNN-402')

print(classification_report(np.argmax(ytest402,axis=1), np.argmax(
    model402.predict(xtest402),axis=1)))
```

6.2.7 Final Model Without Dropouts - Adam Optimizer

```
MurmurNet = Sequential([
     # First Convolutional Layer
     Conv2D(filters=96, input_shape=(xtrain402.shape[1],xtrain402.
    shape[2],1), kernel_size=(11,11), strides=(4,4), padding='same'),
     BatchNormalization(),
     Activation('relu'),
     MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),

     # Second Convolutional Layer
     Conv2D(filters=256, kernel_size=2, strides=(1,1), padding='same')
    ,
     BatchNormalization(),
     Activation('relu'),
     MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),

     # Third Convolutional Layer
     Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='
    same'),
     BatchNormalization(),
     Activation('relu'),

     # Fourth Convolutional Layer
     Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='
    same'),
     BatchNormalization(),
     Activation('relu'),

     # Fifth Convolutional Layer
     Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='same'),
     BatchNormalization(),
     Activation('relu'),
     MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),

     # Passing it to a fully connected layer
     Flatten(),

     # First Fully Connected Layer
     Dense(4096, input_shape=(xtrain402.shape[1],xtrain402.shape[2],1)),
     BatchNormalization(),
     Activation('relu'),

     # Second Fully Connected Layer
     Dense(4096),
     BatchNormalization(),
     Activation('relu'),

     # Third Fully Connected Layer
     Dense(1000),
     BatchNormalization(),
     Activation('relu'),

     #Output Layer
     Dense(2),
     BatchNormalization(),
     Activation('softmax')
 ])

 #Model Summary
 MurmurNet.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
 MurmurNet.summary()

 # train model
 metrics402 = Metrics(validation_data=(xvalid402, yvalid402))
 history402 = MurmurNet.fit(xtrain402, ytrain402, validation_data=(
    xvalid402, yvalid402),
                            epochs=30, callbacks=[metrics402], verbose=2)

 plot_history(history402, 'CNN-402')
 plot_metrics(metrics402, 'CNN-402')

 print(classification_report(np.argmax(ytest402,axis=1), np.argmax(
    MurmurNet.predict(xtest402),axis=1)))
```

6.2.8 Compilation With Adagrad Optimizer

```
#Model Summary
MurmurNet.compile(optimizer='adagrad',loss='binary_crossentropy',
    metrics=['accuracy'])
MurmurNet.summary()
```

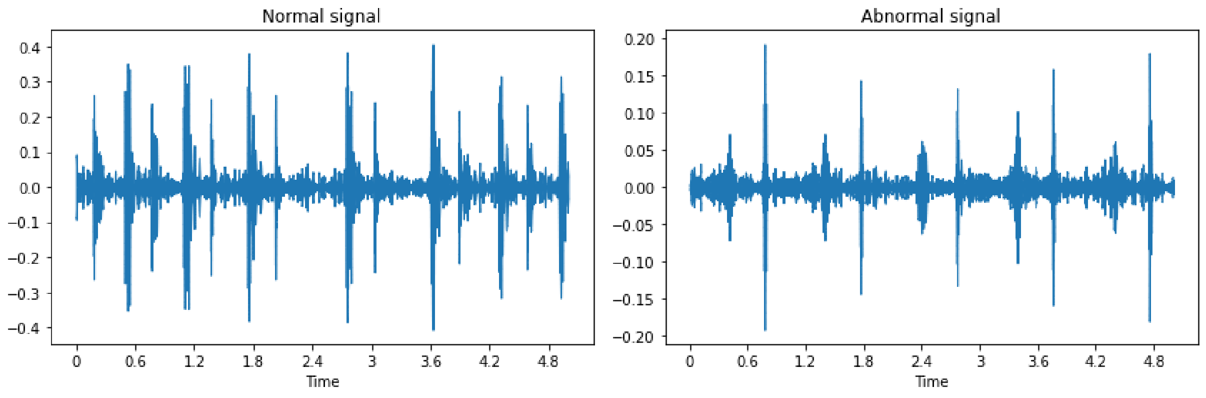


Figure 6.4: Visualization of a Normal and Abnormal Heart Signal

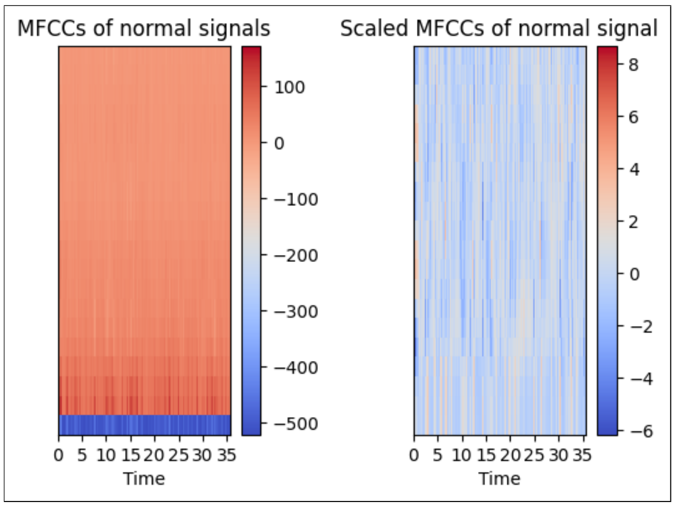


Figure 6.5: MFCC of a Normal heart sound

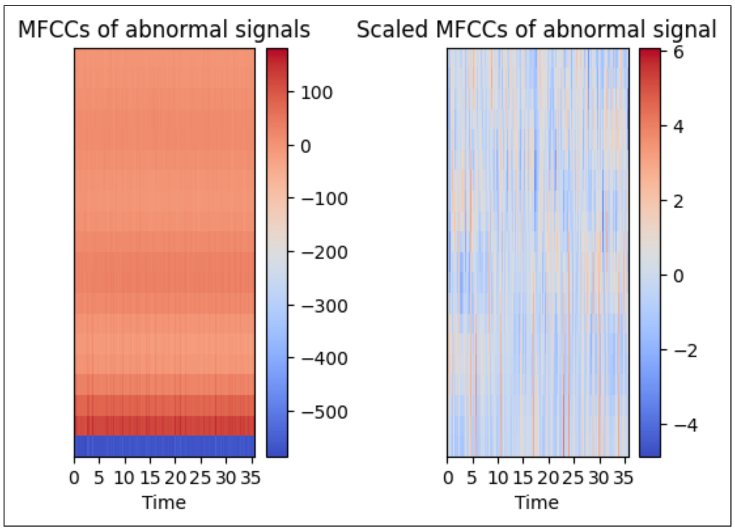


Figure 6.6: MFCC of an Abnormal heart sound

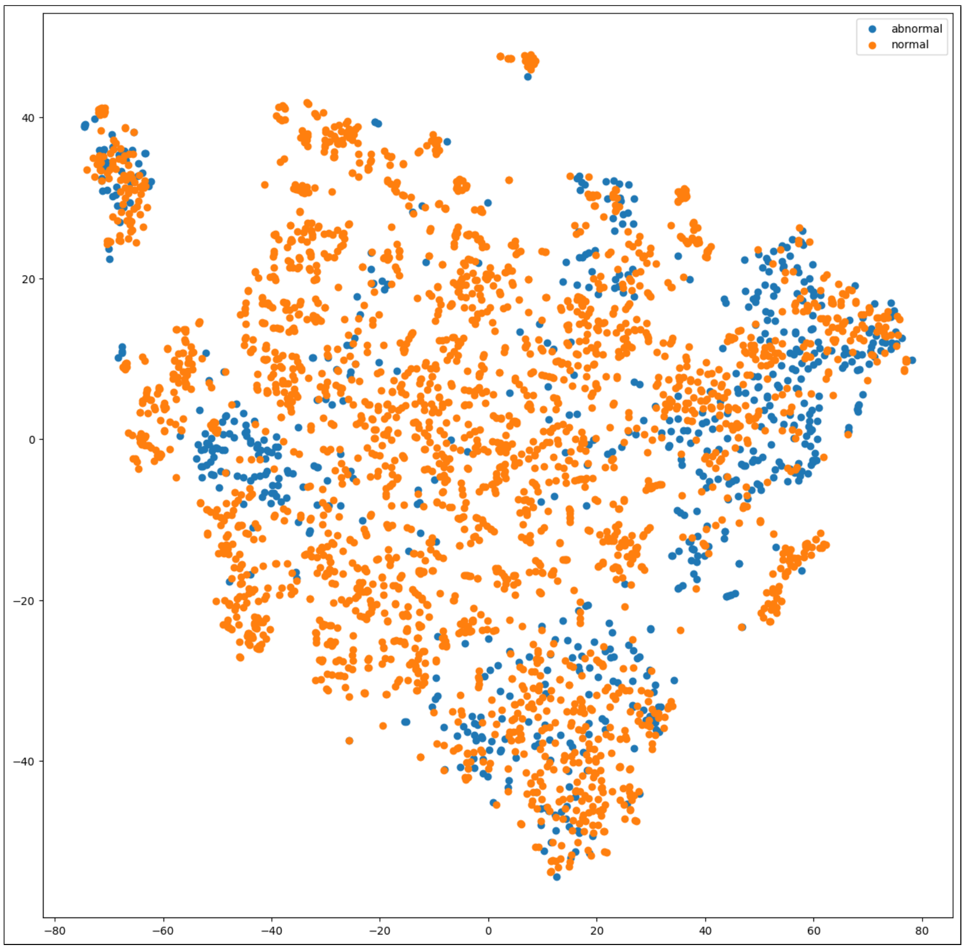


Figure 6.7: T-distributed stochastic neighbor embedding (TSNE)

Chapter 7

Results and Discussions

7.1 Optimizers And Accuracies

The final model provided a chance to try out different parameters and adjust layers to fine-tune the model's accuracy. In some cases, the dropout layer was removed to observe the performance, and the outcomes were recorded when the model was trained with different optimizers [17] while keeping the loss function as binary_crossentropy. In one case, specifically, the Adagrad optimizer [28], the mean squared function was tried to reduce the variation between the accuracies and losses seen on the training and validation datasets.

The optimizers tested in the final model were the Adam optimizer [18], essentially the RMSProp optimizer [29] with momentum, and the SGD or the Stochastic Gradient Descent optimizer [33], the most common choices for Binary classifications. The Nadam optimizer [30], which is the Adam optimizer with Nesterov momentum [31] [34], was also tried. The Nesterov component [32] is used for the updation of the gradient by the Nadam Optimizer. The RMSProp stands for Root Mean Squared Propagation which ensures constant movement in the average calculation of the square of gradients.

The results of the initial attempt are summarized in table 7.1:

Optimizer	Loss Function	Accuracy	Observations
Adam	Binary Cross Entropy	90%	With dropout layers. Significant difference between validation and training accuracies and losses.
Adam	Binary Cross Entropy	91%	Without dropout layers. Significant difference between validation and training accuracies and losses.
SGD	Binary Cross Entropy	89%	With dropout layers. A significant difference between validation and training accuracies, but similar losses.
SGD	Binary Cross Entropy	91%	Without dropouts. No significant variation between validation and training accuracies and losses.

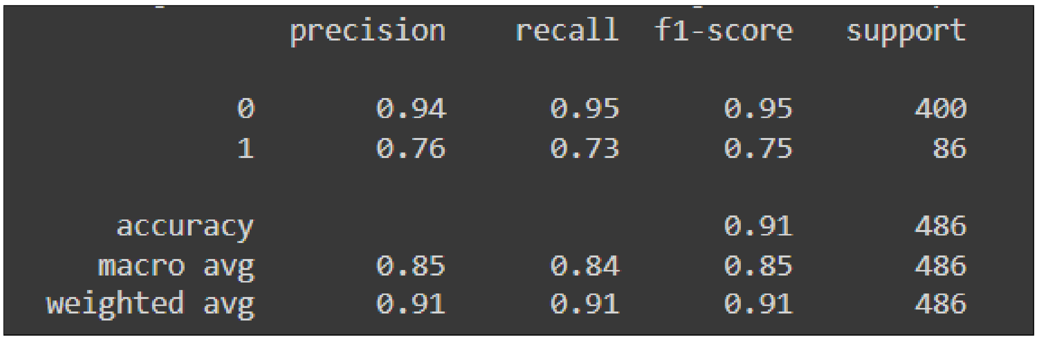
Table 7.1: Optimizers and Accuracies

Optimizer	Loss Function	Accuracy	Observations
Adam	Binary Cross Entropy	90%	With dropout layers. No significant difference between validation and training accuracies and losses.
Adam	Binary Cross Entropy	92%	Without dropout layers. No significant difference between validation and training accuracies and losses.
SGD	Binary Cross Entropy	89%	With dropout layers. Validation accuracy is observed to be greater than training accuracy.
SGD	Binary Cross Entropy	91%	Without dropouts. Significant variation between validation and training accuracies and losses.
Nadam	Binary Cross Entropy	90%	Without dropout layers. No significant difference between validation and training accuracies and losses.
Adagrad	Binary Cross Entropy	91%	Without dropout layers. Significant variation between validation and training accuracies and losses. 99% training accuracy recorded.
Adagrad	Root Mean Squared	89%	Without dropout layers. Significant variation between validation and training accuracies and losses. 99% training accuracy recorded.
RMSProp Optimizer	Binary Cross Entropy	91%	Without dropout layers. No significant difference between validation and training accuracies and losses.

Table 7.2: Results of final model

The results for the final model are summarized in table 7.2:

7.1.1 Initial Model Adam Optimizer Without Dropouts Evaluation Metrics



```
              precision    recall  f1-score   support

           0       0.94      0.95      0.95       400
           1       0.76      0.73      0.75        86

    accuracy                           0.91       486
   macro avg       0.85      0.84      0.85       486
weighted avg       0.91      0.91      0.91       486
```

Figure 7.1: Initial Model Adam Optimizer Metrics

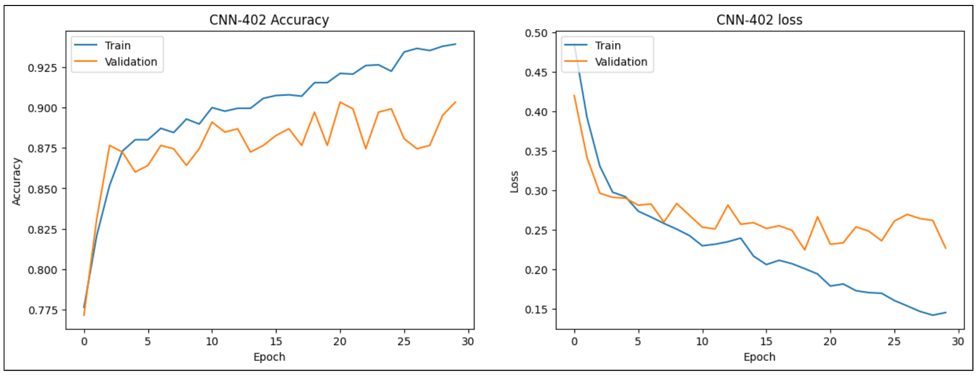
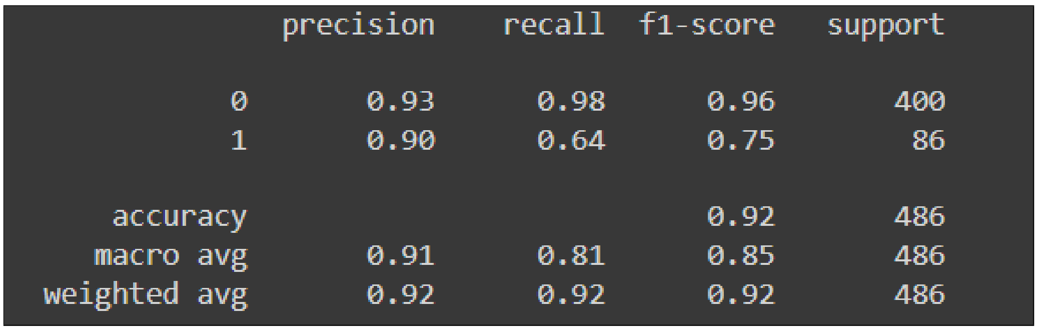


Figure 7.2: Initial Model Adam Optimizer Graphs - Instability Observed

7.1.2 Adam Optimizer Without Dropouts Evaluation Metrics



```
              precision    recall  f1-score   support

           0       0.93      0.98      0.96       400
           1       0.90      0.64      0.75        86

    accuracy                           0.92       486
   macro avg       0.91      0.81      0.85       486
weighted avg       0.92      0.92      0.92       486
```

Figure 7.3: Adam Optimizer Metrics

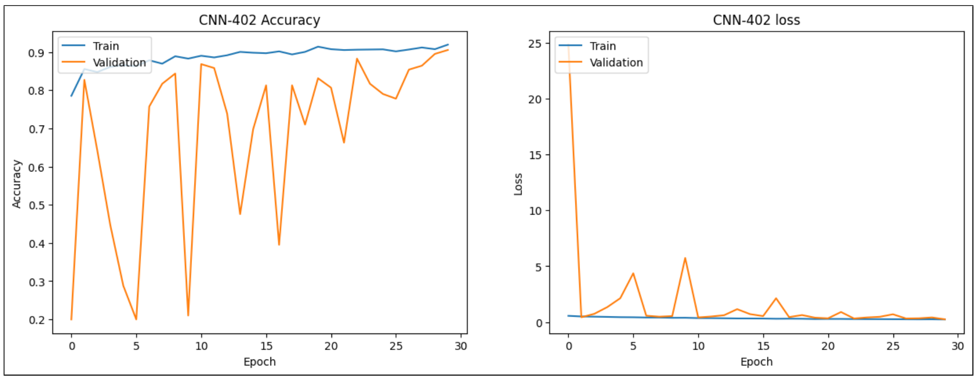
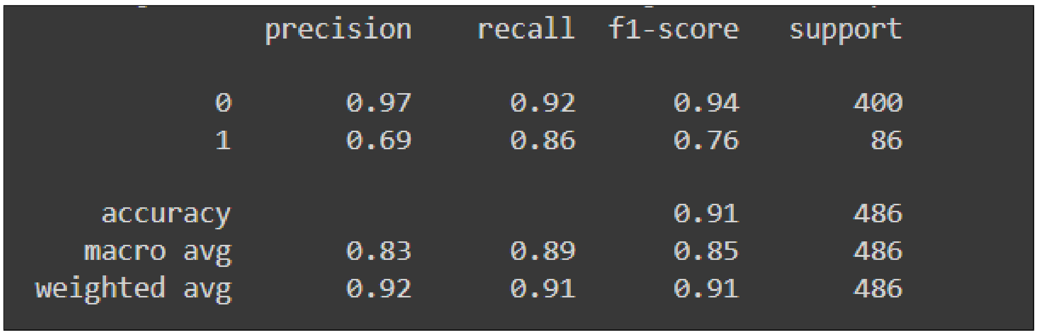


Figure 7.4: Adam Optimizer Graphs

7.1.3 Adagrad Optimizer Without Dropouts - Training Accuracy of 99% - Overfitting



```
              precision    recall  f1-score   support

           0       0.97      0.92      0.94       400
           1       0.69      0.86      0.76        86

    accuracy                           0.91       486
   macro avg       0.83      0.89      0.85       486
weighted avg       0.92      0.91      0.91       486
```

Figure 7.5: Adagrad Optimizer Metrics

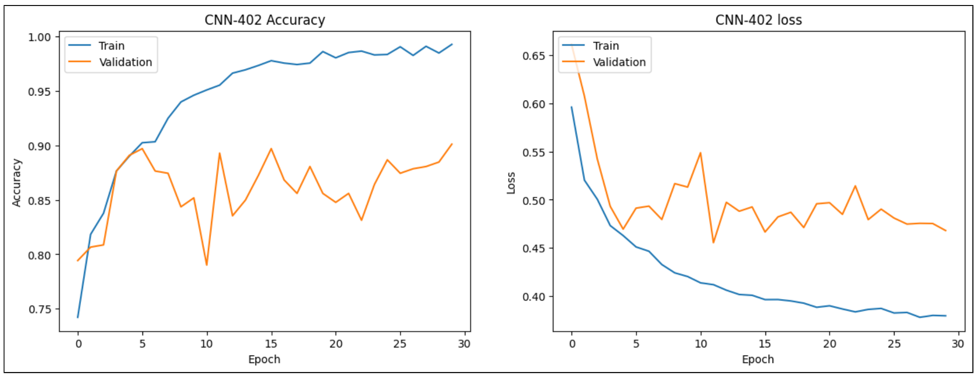


Figure 7.6: Adagrad Optimizer Graphs

Chapter 8

Conclusion & Future Scope

A three-step approach towards multi-document comprehension provides a viable solution with appropriate tradeoffs with efficiency. The implemented network provides equivalent metrics compared to other methods. The end-to-end model does not suffer from loss of context between multiple stages of a pipelined model and provides better overall performance. It does, however, require substantial computation. The model is not generative and thus cannot process more complex natural language queries. Future work can include improving the expensive transformer network to improve efficiency, along with work on the rerank network.

We have trained on 3000 PCGs and tested the remaining data. We preprocessed the audio files using the Butterworth filter. Then we visualized it using Mel frequency cepstral coefficients. We trained it using the AlexNet, which has the first five convolutional layers arranged sequentially, with a max pooling layer after each convolution layer, followed by three dense, fully connected layers with a Dropout layer succeeding each dense layer, followed by the output layer. We also applied hyperparameter tuning, which helped us get a maximum accuracy of 92%. To continue training the pre-trained models, we tried various approaches such as freezing all layers except the last one, training for a few/many epochs, and high or low learning rate. We experimented with various optimizers relevant to the model and worked with different loss functions as well, seeing how they performed, noting their stabilities and the differences that they displayed on the training and validation sets in the accuracies as well as the losses. An accuracy of 99% was observed using the Adagrad optimizer but the possibility of its sustainability was ruled out by the huge difference in the training and validation accuracies.

However, there is always further scope for improvement in all situations as is present in this particular situation too. More research is needed to explain how these CNNs interpret the PCGs comprehensively. A greater emphasis on the explainability of these models with the inclusion of explainable AI to see the accountability of the features in their contribution to the model could yield interesting results with clinical implications.

One approach to improve the learning and, eventually, the model's output is to enhance the dataset and make it more comprehensive. The mentioned idea can be done by making the dataset more representative. Since, for the first version, the target audience is people in rural areas with fewer doctors, we can further extend it to people in cities. The software must be scalable for a full-scale setup in multiple checkup centers if people face problems testing at home because although doctors are present, with the recent skyrocketing reliance on technology, an artificial intelligence model for murmur detections would be useful not only at home but also in clinics, nursing homes, hospitals, and various medical facilities. We can improve the user interface to make it more responsive and user-friendly such that even children and native people can use it effortlessly.

Chapter 9

Acknowledgements

We are extremely thankful to the college administration, our principal Dr. Hari Vasudevan, the Department of Computer Engineering, our Head of Department Dr. Meera Narvekar and last but not least our project guide Prof. Sudhir Bagul for providing us with this unique opportunity to exhibit and hone our skillset.

PAPER NAME

BE_Project_Report.pdf

WORD COUNT	CHARACTER COUNT
11321 Words	63007 Characters
PAGE COUNT	FILE SIZE
51 Pages	1005.6KB
SUBMISSION DATE	REPORT DATE
May 10, 2023 1:32 PM GMT+5:30	May 10, 2023 1:33 PM GMT+5:30

● 4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 3% Internet database
- Crossref database
- 2% Submitted Works database
- 1% Publications database
- Crossref Posted Content database

● Excluded from Similarity Report

- Bibliographic material
- Cited material
- Manually excluded text blocks
- Quoted material
- Small Matches (Less then 10 words)

References

[1] Singstad, B. J., Gitau, A. M., Johnsen, M. K., Ravn, J., Ailo, L., Bongo, H. S., & Medsensio, A. S. Phonocardiogram Classification Using 1-Dimensional Inception Time Convolutional Neural Networks.

[2] Chorba, J. S., Shapiro, A. M., Le, L., Maidens, J., Prince, J., Pham, S., ... & Thomas, J. D. (2021). Deep learning algorithm for automated cardiac murmur detection via a digital stethoscope platform. Journal of the American Heart Association, 10(9), e019905.

[3] Renna, F., Oliveira, J., & Coimbra, M. T. (2019). Deep convolutional neural networks for heart sound segmentation. IEEE journal of biomedical and health informatics, 23(6), 2435-2445.

[4] Summerton, S., Wood, D., Murphy, D., Redfern, O., Benatan, M., Kaisti, M., & Wong, D. C. (2022, August). Two-stage Classification for Detecting Murmurs from Phonocardiograms Using Deep and Expert Features. In Computing in Cardiology 2022: 49th Computing in Cardiology Conference.

[5] Elola, A., Aramendi, E., Oliveira, J., Renna, F., Coimbra, M. T., Reyna, M. A., ... & Rad, A. B. (2022). Beyond Heart Murmur Detection: Automatic Murmur Grading from Phonocardiogram. arXiv preprint arXiv:2209.13385.

[6] Grzywalski, T., Maciaszek, A., Belluzzo, R., Szarzyński, K., Piecuch, M., & Hafke-Dys, H. (2020, July). Joint Heart Sounds Segmentation and Murmur Detection with Masked Loss Function. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-6). IEEE.

[7] Wang, J., You, T., Yi, K., Gong, Y., Xie, Q., Qu, F., ... & He, Z. (2020). Intelligent diagnosis of heart murmurs in children with congenital heart disease. Journal of healthcare engineering, 2020.

[8] Draper, J., Subbiah, S., Bailey, R., & Chambers, J. B. (2019). Murmur clinic: validation of a new model for detecting heart valve disease. Heart, 105(1), 56-59.

[9] Petrolis, R., Paukstaitiene, R., Rudokaite, G., Macas, A., Grigaliunas, A., & Krisciukaitis, A. Convolutional neural network approach for heart MurMur detection in auscultation signals using wavelet transform based features.

[10] Imran, Z., Grooby, E., Malgi, V. V., Sitaula, C., Aryal, S., & Marzbanrad, F. A Fusion of Handcrafted Feature-Based and Deep Learning Classifiers for Heart Murmur Detection.

[11] Moulana, S. H., Luqman, A., Abdulghafor, R., Wani, S., & Ibrahim, A. A. (2022). Heart Murmur Detection using Supervised Machine Learning. International Journal on Perceptive and Cognitive Computing, 8(2), 25-29.

[12] Jalali, K., Saket, M. A., & Noorzadeh, S. Heart Murmur Detection and Clinical Outcome Prediction using Multilayer Perceptron Classifier.

[13] Fernando, T., Ghaemmaghami, H., Denman, S., Sridharan, S., Hussain, N., & Fookes, C. (2019). Heart sound segmentation using bidirectional LSTMs with attention. IEEE journal of biomedical and health informatics, 24(6), 1601-1609.

[14] Rezek, Iead, and Stephen J. Roberts. ”Envelope extraction via complex homomorphic filtering.” Technical Report TR-98-9 Technical report (1998).

[15] Mahata, Shibendu, et al. ”Revisiting the use of squared magnitude function for the optimal approximation of (1+ α)-order Butterworth filter.” AEU-International Journal of Electronics and Communications 110 (2019): 152826.

[16] Johansson, Mathias. ”The hilbert transform.” Mathematics Master's Thesis. Växjö University, Suecia. Disponible en internet: http://w3. msi. vxu. se/exarb/mj_ex. pdf, consultado el 19 (1999).

[17] Manataki, M., Vafidis, A., & Sarris, A. (2021, December). Comparing Adam and SGD optimizers to train AlexNet for classifying GPR C-scans featuring ancient structures. In 2021 11th International Workshop on Advanced Ground Penetrating Radar (IWAGPR) (pp. 1-6). IEEE.

[18] Jais, I. K. M., Ismail, A. R., & Nisa, S. Q. (2019). Adam optimization algorithm for wide and deep neural network. Knowledge Engineering and Data Science, 2(1), 41-46.

[19] Waoo, A. A., & Soni, B. K. (2021). Performance Analysis of Sigmoid and Relu Activation Functions in Deep Neural Network. In Intelligent Systems: Proceedings of SCIS 2021 (pp. 39-52). Springer Singapore.

[20] Dhar, P., Dutta, S., & Mukherjee, V. (2021). Cross-wavelet assisted convolution neural network (AlexNet) approach for phonocardiogram signals classification. Biomedical Signal Processing and Control, 63, 102142.

[21] Zhang, X., & Jiang, S. (2021, April). Application of fourier transform and butterworth filter in signal denoising. In 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP) (pp. 1277-1281). IEEE.

[22] Siedenburg, K., Fujinaga, I., & McAdams, S. (2016). A comparison of approaches to timbre descriptors in music information retrieval and music psychology. Journal of New Music Research, 45(1), 27-41.

[23] Rao, K. S., Prasanna, S. M., & Yegnanarayana, B. (2007). Determination of instants of significant excitation in speech using Hilbert envelope and group delay function. IEEE Signal Processing Letters, 14(10), 762-765.

[24] Hassani, K., Bajelani, K., Navidbakhsh, M., Doyle, D. J., & Taherian, F. (2014). Heart sound segmentation based on homomorphic filtering. Perfusion, 29(4), 351-359.

[25] Zheng, Fang, Guoliang Zhang, and Zhanjiang Song. ”Comparison of different implementations of MFCC.” Journal of Computer science and Technology 16 (2001): 582-589.

[26] Gupta, Shikha, et al. ”Feature extraction using MFCC.” Signal & Image Processing: An International Journal 4.4 (2013): 101-108.

[27] Van der Maaten, Laurens, and Geoffrey Hinton. ”Visualizing data using t-SNE.” Journal of machine learning research 9.11 (2008).

[28] Lydia, Agnes, and Sagayaraj Francis. ”Adagrad—an optimizer for stochastic gradient descent.” Int. J. Inf. Comput. Sci 6.5 (2019): 566-568.

[29] Kurbiel, Thomas, and Shahrzad Khaleghian. ”Training of deep neural networks based on distance measures using RMSProp.” arXiv preprint arXiv:1708.01911 (2017).

[30] Halgamuge, Malka N., Eshan Daminda, and Ampalavanapillai Nirmalathas. ”Best optimizer selection for predicting bushfire occurrences using deep learning.” Natural Hazards 103.1 (2020): 845-860.

[31] Dozat, Timothy. ”Incorporating nesterov momentum into adam.” (2016).

[32] Xie, Xingyu, et al. ”Adan: Adaptive Nesterov Momentum Algorithm for Faster Optimizing Deep Models.” arXiv preprint arXiv:2208.06677 (2022).

[33] Keskar, Nitish Shirish, and Richard Socher. ”Improving generalization performance by switching from adam to sgd.” arXiv preprint arXiv:1712.07628 (2017).

[34] Landro, Nicola, Ignazio Gallo, and Riccardo La Grassa. ”Mixing Adam and SGD: a combined optimization method.” arXiv preprint arXiv:2011.08042 (2020).

[35] Alaskar, Haya, et al. ”The implementation of pretrained AlexNet on PCG classification.” Intelligent Computing Methodologies: 15th International Conference, ICIC 2019, Nanchang, China, August 3–6, 2019, Proceedings, Part III 15. Springer International Publishing, 2019.

[36] Demir, Fatih, Daban Abdulsalam Abdullah, and Abdulkadir Sengur. ”A new deep CNN model for environmental sound classification.” IEEE Access 8 (2020): 66529-66537.

[37] Christlein, Vincent, et al. ”Deep generalized max pooling.” 2019 International conference on document analysis and recognition (ICDAR). IEEE, 2019.

[38] Lee, Kangbae, et al. ”Verification of normalization effects through comparison of CNN models.” 2019 International Conference on Multimedia Analysis and Pattern Recognition (MAPR). IEEE, 2019.

[39] Santurkar, Shibani, et al. ”How does batch normalization help optimization?.” Advances in neural information processing systems 31 (2018).

[40] Bjorck, Nils, et al. ”Understanding batch normalization.” Advances in neural information processing systems 31 (2018).

[41] Baldi, Pierre, and Peter J. Sadowski. ”Understanding dropout.” Advances in neural information processing systems 26 (2013).

[42] Lin, Guifang, and Wei Shen. ”Research on convolutional neural network based on improved Relu piecewise activation function.” Procedia computer science 131 (2018): 977-984.