Cardiovascular Diagnosis Using Federated Learning

A Project Report

Submitted by

ANN MARIYA	**JEC16CS026**
RAHUL M	**JEC16CS092**
MANEESH MANOJ	**JEC17CS063**
RASHI M	**JEC17CS079**

to

APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology (B.Tech)

in

Computer Science & Engineering

Under the guidance of

Mrs. Namitha T N

CREATING TECHNOLOGY
LEADERS OF TOMORROW
ESTD 2002

Department of Computer Science & Engineering

Jyothi Engineering College

NAAC Accredited College with NBA Accredited Programmes*

Approved by AICTE & affiliated to APJ Abdul Kalam Technological University

A CENTRE OF EXCELLENCE IN SCIENCE & TECHNOLOGY BY THE CATHOLIC ARCHDIOCESE OF TRICHUR

JYOTHI HILLS, VETTIKATTIRI P.O, CHERUTHURUTHY, THRISSUR. PIN-679531 PH : +91- 4884-259000, 274423 FAX : 04884-274777

NBA accredited B.Tech Programmes in Computer Science & Engineering, Electronics & Communication Engineering, Electrical & Electronics Engineering and Mechanical Engineering valid for the academic years 2016-2022. NBA accredited B.Tech Programme in Civil Engineering valid for the academic years 2019-2022.

June 2021

DECLARATION

We the undersigned hereby declare that the project report “Cardiovascular Diagnosis Using Federated Learning”, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Mrs. Namitha T N. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in this submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously used by anybody as a basis for the award of any degree, diploma or similar title of any other University.

Name of Students	**Signature**
ANN MARIYA (JEC16CS026)	
RAHUL M (JEC16CS092)	
MANEESH MANOJ (JEC17CS063)	
RASHI M (JEC17CS079)	

Place:

Date:

Jyothi Engineering College
NAAC Accredited college with NBA Accredited programmes

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JYOTHI ENGINEERING COLLEGE
LEAD KINDLY LIGHT
TET
CREATING TECHNOLOGY
LEADERS OF TOMORROW
ESTD 2002

CERTIFICATE

This is to certify that the report entitled " **CARDIOVASCULAR DIAGNOSIS USING FEDERATED LEARNING** " submitted by ANN MARIYA(JEC16CS026), RAHUL M(JEC16CS092), MANEESH MANOJ (JEC17CS063), and RASHI M(JEC17CS079) to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree in Bachelor of Technology in **Computer Science & Engineering** is a bonafide record of the project work carried out by them under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof. Dr Saju P John

Professor
Internal Supervisor & Head of the Department

ACKNOWLEDGEMENT

We take this opportunity to thank everyone who helped us profusely, for the successful completion of our project work. With prayers, we thank **God Almighty** for his grace and blessings, for without his unseen guidance, this project would have remained only in our dreams.

We thank the **Management** of Jyothi Engineering College and our Principal, **Dr Sunny Joseph Kalayathankal** for providing all the facilities to carry out this project work. We are grateful to the Head of the Department **Prof. Dr Saju P John** for his valuable suggestions and encouragement to carry out this project work. Our sincere thanks to **Dr. Vinith R**, former Head of the Department for permitting us to make use of the facilities available in the department to carry out the project successfully.

We would like to express our whole hearted gratitude to the project guide **Mrs. Namitha T N** for her encouragement, support and guidance in the right direction during the entire project work.

We thank our Project Coordinators **Dr. Swapna B Sasi** & **Mr. Shaiju Paul** for their constant encouragement during the entire project work. We extend our gratefulness to all teaching and non teaching staff members who directly or indirectly involved in the successful completion of this project work.

Finally, we take this opportunity to express our gratitude to the parents for their love, care and support and also to our friends who have been constant sources of support and inspiration for completing this project work.

ANN MARIYA (JEC16CS026)
RAHUL M (JEC16CS092)
MANEESH MANOJ (JEC17CS063)
RASHI M (JEC17CS079)

VISION OF THE INSTITUTE

Creating eminent and ethical leaders through quality professional education with emphasis on holistic excellence.

MISSION OF THE INSTITUTE

- To emerge as an institution par excellence of global standards by imparting quality Engineering and other professional programmes with state-of-the-art facilities.
- To equip the students with appropriate skills for a meaningful career in the global scenario.
- To inculcate ethical values among students and ignite their passion for holistic excellence through social initiatives.
- To participate in the development of society through technology incubation, entrepreneurship and industry interaction.

VISION OF THE DEPARTMENT

Creating eminent and ethical leaders in the domain of computational sciences through quality professional education with a focus on holistic learning and excellence.

MISSION OF THE DEPARTMENT

- To create technically competent and ethically conscious graduates in the field of Computer Science & Engineering by encouraging holistic learning and excellence.
- To prepare students for careers in Industry, Academia and the Government.
- To instill Entrepreneurial Orientation and research motivation among the students of the department.
- To emerge as a leader in education in the region by encouraging teaching, learning,industry and societal connect

PROGRAMME EDUCATIONAL OBJECTIVES

PEO 1: The graduates shall have sound knowledge of Mathematics, Science, Engineering and Management to be able to offer practical software and hardware solutions for the problems of industry and society at large.

PEO 2: The graduates shall be able to establish themselves as practising professionals, researchers or Entrepreneurs in computer science or allied areas and shall also be able to pursue higher education in reputed institutes.

PEO 3: The graduates shall be able to communicate effectively and work in multidisciplinary teams with team spirit demonstrating value driven and ethical leadership.

PROGRAMME SPECIFIC OUTCOMES

Graduate possess -

PSO 1: An ability to apply knowledge of data structures and algorithms appropriate to computational problems.

PSO 2: An ability to apply knowledge of operating systems, programming languages, data management, or networking principles to computational assignments.

PSO 3: An ability to apply design, development, maintenance or evaluation of software engineering principles in the construction of computer and software systems of varying complexity and quality.

PSO 4: An ability to understand concepts involved in modeling and design of computer science applications in a way that demonstrates comprehension of the fundamentals and trade-offs involved in design choices.

PROGRAMME OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

COURSE OUTCOMES

COs	Description
C410.1	The students will be able to analyse a current topic of professional interest and present it before an audience.
C410.2	Students will be able to identify an engineering problem, analyse it and propose a work plan to solve it.
C410.3	Students will have gained thorough knowledge in design, implementations and execution of Computer science related projects.
C410.4	Students will have attained the practical knowledge of what they learned in theory subjects.
C410.5	Students will become familiar with usage of modern tools.
C410.6	Students will have ability to plan and work in a team.

CO MAPPING TO POs

	POs											
COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C410.1	3	2	3	3	3	3	2	3	3	2	3	2
C410.2	3	3	3	3	3	3	2	3	2	3	3	3
C410.3	3	2	2	3	3	3	3	3	2	3	3	3
C410.4	3	3	2	3	3	3	3	2	3	3	3	3
C410.5	2	3	3	3	2	3	3	2	2	3	2	2
C410.6	3	2	3	2	2	3	2	3	2	3	2	2
Average	2.83	2.5	2.67	2.83	2.67	3	2.5	2.67	2.33	2.83	2.67	2.67

CO MAPPING TO PSOs

	PSOs			
COs	PSO1	PSO2	PSO3	PSO4
C410.1	3	3	1	3
C410.2	3	3	2	3
C410.3	3	1	3	2
C410.4	2	1	3	2
C410.5	3	1	2	2
C410.6	3	2	3	3
Average	2.83	1.833	2.33	2.5

ABSTRACT

Cardiovascular diseases are the number one cause of deaths globally as per WHO. The first step of diagnosing such diseases is to auscultate or to listen for any abnormal sound, usually done by a medical practitioner. The effective diagnosis of any such condition depends upon the skill and experience of the physician. The usage of Machine Learning based as a solution for the above problem is hindered due to data privacy restrictions and confidential nature of medical data. This project investigates the employment of a privacy preserving Machine Learning paradigm known as Federated Learning as a solution to the aforementioned problem.

Keywords: Machine Learning, Federated learning, Sequence Neural Network, Cardiovascular disease diagnosis, Spectrogram.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

CNN	: Convolutional Neural Networks
RNN	: Recurrent Neural Networks
DFD	; Data Flow Diagram
FL	: Federated Learning
KNN	: K Nearest Neighbour
ESC	: Environmental Sound Classification
FTL	: Federated Transfer Learning
CVD	: Cardiovascular Disease
SNR	: Signal to Noise Ratio
TF	: TensorFlow

CHAPTER 1

INTRODUCTION

1.1 Overview

Our project is based on the papers Federated Learning for Healthcare Informatics[1] which discusses about how federated learning technologies can improve the machine learning practice especially in the field of healthcare and medicine and create a global improvement; by utilizing the data collected in distributed hospitals without compromising data privacy of its patients by sharing only the inferences of local model training or optimization; not the entire local data. The best part is that this collection of local improvements will, in turn, produce a much efficient global model respecting all the data privacy laws. In this project we try to use Federated Learning technology in Health Analytic purpose, to diagnose cardiovascular conditions. The main constraint while creating Machine Learning model in Health Sector are respectable privacy policies. By Federated Learning, we can perform machine learning on decentralized server or devices (here on local server of the respective hospital) preserving privacy constraints. Then we pass only the learned inference of the respective model trained on the local server to a central server. Similarly, the inferences of multiple servers from different hospitals contribute their inferences and these are aggregated to form a globally improved model. This model is then shared with all the participants, and they can attain a globalized improvised performance.

1.2 Objectives

The main objective of this project is to diagnose cardiovascular conditions using a Federated Learning based system. The usage of FL in this project work improves the accuracy of the classifier due to access to much more data, overcoming privacy restrictions when compared with conventional ML.

1.3 Data Description

The data for this project is taken from an open source challenge dataset known as ”Classification of Heart Sound Recordings - The PhysioNet Computing in Cardiology Challenge 2016”. The dataset is publicly available at https://physionet.org/content/challenge-2016/1.0.0/files. The dataset is divided into abnormal and normal labels, and combined the dataset has 2000+ recordings taken using an digital stethoscope. As is the case with a usual deep learning problem, we would be training the model using training dataset and evaluating the performance with the validation dataset.

1.4 Organization of the project

The report is organised as follow:
• Chapter 1: Introduction- Gives an introduction to “Cardiovascular Diagnosis Using Federated Learning”.
• Chapter 2: Literature Survey- Summarizes the various existing techniques that helps in achieving the desired result.
• Chapter 3:Methodology- Methods which are used in this project
• Chapter 4 Results and Discussion needed for implementation.
• Chapter 5: Conclusion & Future Scope- Concludes with the future scope of implementation.
• References: Includes the references for the project.

CHAPTER 2

LITERATURE SURVEY

2.1 Federated Machine Learning: Concept and Applications

Federated learning is a Machine Learning technique that trains algorithms across multiple decentralized edge devices or services holding data samples, without exchanging them. Federated learning enables multiple actors to build a common, robust machine learning model without sharing data, thus allowing to address critical issues such as data privacy, data security, data access rights and access to heterogeneous data[3].

The concept of Federated Learning was proposed by Google in 2016 as an idea to build machine-learning models based on datasets that are distributed across multiple devices while preventing data leakage.

Federated learning is essentially a training methodology with participating devices holding data without exposing them, collaboratively train a model $\mathbf{M_{FED}}$ with accuracy $\mathbf{V_{FED}}$, as opposed to conventional machine learning training methods producing a model $\mathbf{M_{SUM}}$ with accuracy $\mathbf{V_{SUM}}$ such that the absolute difference is less than Delta[2].

$$\mathbf{V_{FED}} - \mathbf{V_{SUM}} < \mathrm{Delta} \tag{2.1}$$

2.1.1 Categorization of Federated Learning

- Horizontal FL

  Horizontal federated learning, or sample-based federated learning, is introduced in the scenarios in which datasets share the same feature space but different space in samples. This essentially means that Client A and Client B has the same set of features. This version of FL is the most widely used in use cases in the current scenario. A horizontal federated learning system typically assumes honest participants and security against an honest-but-curious server. That is, only the server can compromise the privacy of data participants. At the end of the training, the universal model and all of the model parameters are exposed to all participants.

- Vertical FL

  Vertical federated learning uses different datasets of different feature space to jointly train a global model. Privacy-preserving machine-learning algorithms have been proposed for vertically partitioned data. Vertical federated learning or feature-based federated learning is applicable to the cases in which two datasets share the same sample

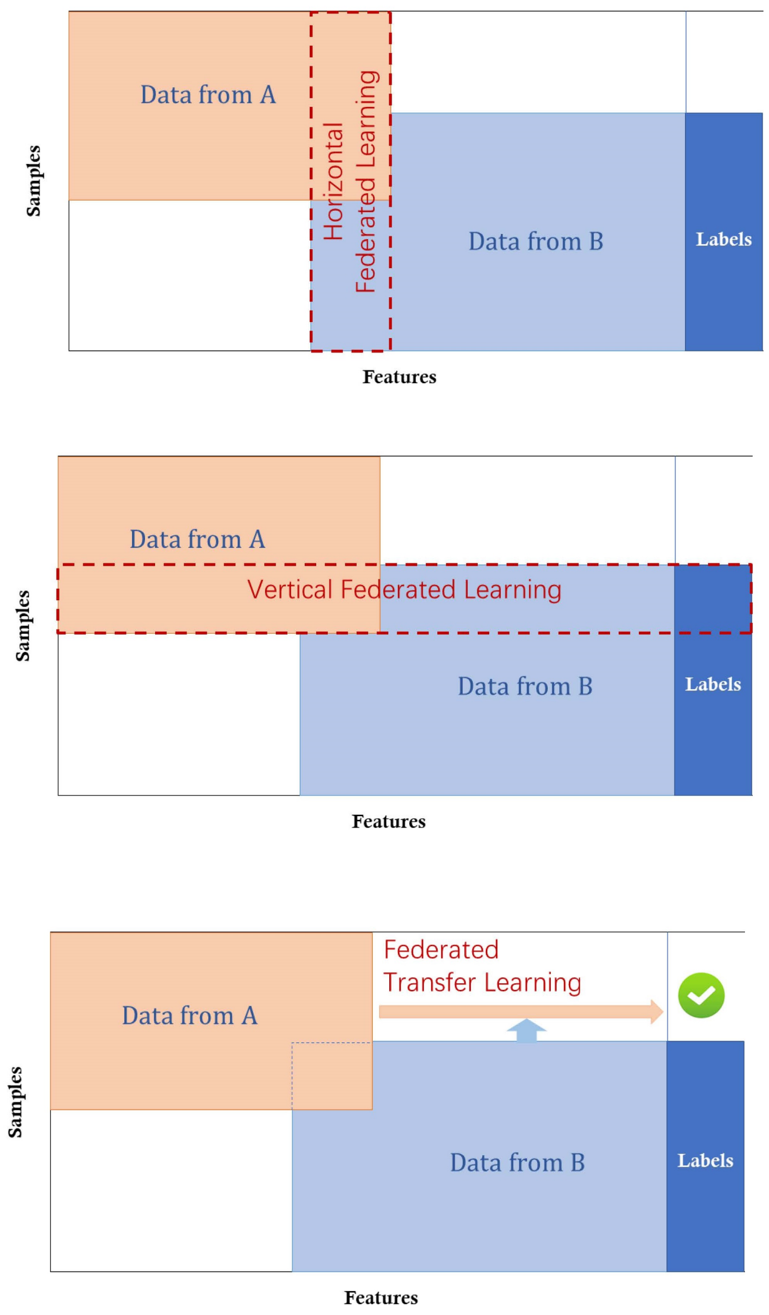


Figure 2.1: Horizontal, Vertical & Transfer FL

ID space but differ in feature space. Vertically federated learning is the process of aggregating these different features and computing the training loss and gradients in a privacy-preserving manner to build a model with data from both parties collaboratively. A vertical federated-learning system typically assumes honest but curious participants.

- Federated Transfer Learning.

Federated transfer learning is vertical federated learning utilized with a pre-trained model that is trained on a similar dataset for solving a different problem. One such example of Federated transfer learning is to train a personalised model e.g. Movie recommendation for the user's past browsing behavior. FTL is an important extension to the existing federated learning systems because it deals with problems exceeding the scope of existing federated learning algorithms.

2.1.2 Applications of FL

FL, being an innovative modeling mechanism that could train a collaborative model on data from multiple devices, without compromising privacy and security of those data, has a promising application in sales, financial, and many other industries in which data cannot be directly aggregated for training machine-learning models owing to factors such as intellectual property rights, privacy protection, and data security. Therefore, FL provides good technical support for us to build a cross-enterprise, cross-data, and cross-domain ecosphere for big data and AI. These are the main reason why FL is now well researched and applications are being developed in areas like:

- Personalised Ads and Commercials
- Healthcare Informatics and Smart Diagnose
- Smart Retail innovations

2.2 Spectral images based environmental sound classification using CNN

Convolutional Neural Networks (CNNs) are conventionally used to map image data to an output variable. Several other use cases of CNNs were discovered and utilized over time with the advent of proper data preprocessing and feature extraction methods. One such use of CNN is to classify audio data which are fed as spectrogram or any other graphical plotting of audio signals.

Environmental Sound Classification (ESC) refers to the common task of classifying audio signals based on the various component signals [4]. Most of the recent implementations of ESC uses CNN as a classifier due to the use of spectral images over audio clips. The spectral images can be viewed as a visible representation of the frequency spectrum for the audio signals. Audio signals are less periodic, weak ambiance, short interval, and the addition of noise on audio signals is much easy as compared with images.

The frequency spectrum of the audio signal is visually represented in the form of spectrogram images. It is a very rare approach to convert audio files into images for classification tasks. As discussed earlier, such conversion can provide a better classification accuracy and a less error rate. This study utilizes the following datasets for model training and validation purposes:

- ESC-10: This dataset consists of 400 short clips recordings with an average time span of 5s each. These small clips involve 10 different classes with a total time duration of 33 min
- ESC-50 The collection of this dataset is 2000 short recordings of 50 separate classes, which are grouped into 5 major categories
- Urbansound8k: The Urbansound8k (Us8k) includes 8732 sound clips. The average period of these short clips is up to four seconds each with a total of 10 classes of various indoor and outdoor environmental sounds.

2.2.1 Workflow of ESC procedure

- Data Preprocessing

  Audio recordings have background noises, very short intervals, and rapid changes in the clips. These noisy signals make it very hard for classification task. This study involves the classification of sounds from the environment after converting the audio clips into spectrogram images, which keeps the effects of noise in signals from affecting the classification procedure to a large extent.

- Classifier

  CNNs are presented in this study as a method of classifying audio signals by converting sound signals into spectrograms.

2.3 Classification of Heart Sounds Using Convolutional Neural Network

Heart sound play an important role in the diagnosis of cardiac conditions but it is problematic and time-consuming even for experts to discriminate different kinds of heart sounds due to the low signal-to-noise ratio(SNR)[5]. In this paper “Classification of Heart Sounds Using Convolutional Neural Network” a conventional feature engineering method is combined with deep learning algorithms to automatically classify normal and abnormal heart sounds.

Statistics show that cardiovascular disease (CVD) is one of the main reasons for mortality in the world. Heart sounds are a kind of mechanical vibration which is caused by the movement of blood in the cardiovascular system and they are considered to be an important indicator in the diagnosis of several CVDs. phonocardiogram (PCG) which is a graphical recording of heart sounds, can be used to diagnose deformation in heart organs and damage to heart valves. The common approaches to heart sound or PCG classification include three steps:

- Feature extraction

- Feature selection
- Classification by a classifier

2.3.1 Materials and Methods

- Dataset

  Dataset contains a total of 3153 PCG recordings and is composed of six subsets from different part of the world. Their sampling frequency is 2000Hz and length of recording varies from 5s to 120s. The dataset includes 2488 normal samples and 665 abnormal samples manually labelled with 1 and -1, respectively. The proposed method includes two major steps—feature extraction and feature selection and classification based on the designed CNN. First, the phonocardiogram (PCG) is preprocessed and

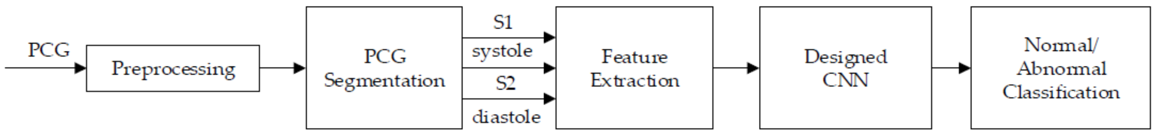


Figure 2.2: Block diagram of proposed method

  segmented. After that, the four states of each PCG recording are used to extract multiple features. Finally, the extracted features are fed into the designed convolutional neural network (CNN) model to classify normal and abnormal heart sound.

- Preprocessing

  Firstly, each PCG recording was filtered by a high-pass filter with a cut-off frequency of 10 Hz to remove baseline drift. Secondly, the spike removal algorithm was applied to the filtered recordings. Thirdly, the recordings were normalized to zero mean and unit standard deviation. Finally, the PCG recordings were segmented using the method of the hidden semi-Markov model (HSMM) segmentation method.

2.3.2 Heart Sound Classification Based on the Designed CNN

The proposed model in this paper[5] is composed of three Conv-blocks, a global average pooling (GAP) layer and a classification layer with the sigmoid function. Each Conv-block includes a convolutional1D (Conv-1D) layer and a maxpooling1D layer, followed by the strategy of dropout to prevent overfitting. The numbers of filters for the Conv1D layers were set to 32, 64 and 128, respectively and each filter had a kernel size of 3 and strides 3. The pooling size was 2 with two strides 2. Relu was adopted as the activation function in each convolution layer. Conv1D denotes a one-dimensional convolution layer. GAP denotes the global average pooling layer.

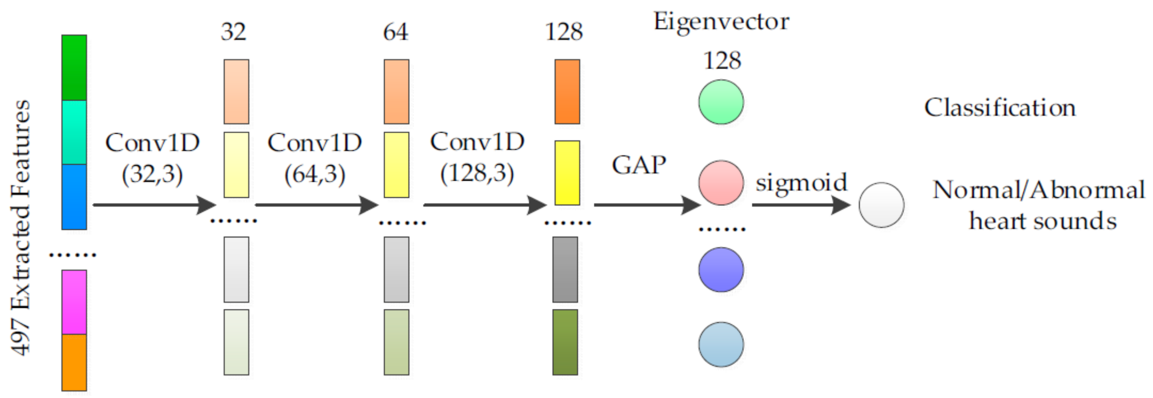


Figure 2.3: Structure of designed CNN

In this proposed classification method, as the input of the model comprises all of the extracted features, we assume that the designed model is used only to select features and classify normal and abnormal heart sounds. Therefore, the designed model is a shallow or compact model. Pooling layers is also used to reduce the feature dimensions. So, the dimensions of the input features gradually decreased from the bottom layers to the top layers of the Conv-blocks. This procedure reorganizes and reduces the dimensionality of the input (497 features in total), which can also be viewed as feature selection in signal processing.

2.4 Learning Image-based Representations for Heart Sound Classification

Heart disease continues to be a leading worldwide health burden. Phonocardiograph is a method of recording the sounds and murmurs made by heartbeats, as well as the associated turbulent blood flow with a stethoscope, over various locations in the chest cavity. Phonocardiogram is widely used in the diagnose of heart disease.

In this paper[6] Image classification convolutional neural network is utilised to process scalogram images of PCG recording for abnormal heart sound detection. Instead of training CNNs from the scratch, the aforementioned pre-trained ImageNet is used to construct robust heart sound classification models.

2.4.1 Scalogram Representation

In this paper, to transform the PCG samples into images which can be processed by an ImageNet, the scalogram images are generated using the morse wavelet transformation with 2 kHz sampling frequency. While creating the images the frequency is represented in kHz on the vertical axis and time is represented in s on the horizontal axis. Viridis colour map, which varies from blue (low range) to green (mid-range) to red (upper range) is used to colour the wavelet coefficient values. Axes and margins are removed and finally, the scalogram images are scaled to 224×224 for compatibility with the VGG16 ImageNet.

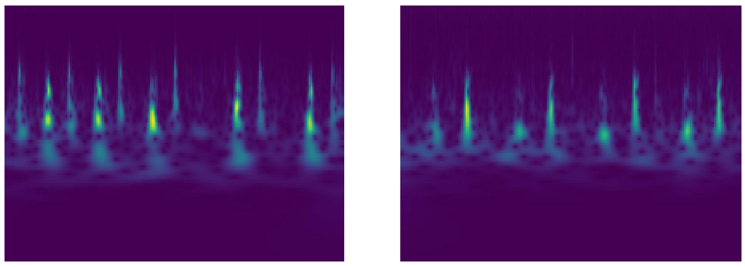


Figure 2.4: Scalogram images of a normal and an abnormal heartbeat

It can even be observed by human eyes that, there are some clear distinctions between the two classes in these images. The scalogram images are extracted from the first 4 s segments of normal/ abnormal heart sounds using the Viridis colour map.

2.4.2 Convolutional Neural Networks

In this paper, ImageNet is used to process the scalogram images for the heart sound classification. VGG16 is constructed from 13 ([2, 2, 3, 3, 3]) convolutional layers, five max-pooling layers, three fully connected layers fc6, fc7, fc and a soft-max layer for 1000 labels according to the image classification task in the ImageNet Challenge. The receptive field size of 3×3 is used in all of the convolutional layers.

2.4.3 Deep PCG Feature Representations

PCG Feature Extraction from ImageNet: The activations of the first fully connected layer fc6 of VGG16 are employed as the feature representations. Essentially, scalogram images are feed into VGG16 and then the deep PCG feature representations of 4096 attributes are extracted as the activations of all neurons in the first fully connected layer fc6.

PCG Feature Extraction from adapted ImageNet: Transfer learning methodology is used to adapt the parameters of VGG16 to better suit the task of abnormal heart sound detection. After the adaptation, the scalogram images are fed into the updated CNN model and a new set of deep representations are extracted from the first fully connected layer fc6.

2.4.4 End-to-end ImageNet based Classification

The parameter of VGG16 is adapted on the heart sound data by transfer learning. To construct a robust end-to-end heart sound CNN classifier, two different approaches are used.

- Learning Classifier of ImageNet ImageNet classifier is created by freezing the parameter of the convolutional layers and fc6, and updating the parameters of the final two fully connected layers and the soft-max layer for classification
- Learning ImageNet In this method, replace the last fully connected layer with a new one which has 2 neurons and a soft-max layer in order to achieve the 2-class classification

task. Then update the entire network so that all VGG16 parameters are adapted to the heart sound data. This method represents a faster way to achieve a full CNN based classification than training an entire CNN from scratch with random initialisation of parameters.

2.5 The future of digital health with federated learning

Data-driven machine learning (ML) has emerged as a promising approach for building accurate and robust statistical models from medical data, which is collected in huge volumes by modern healthcare systems. Research on artificial intelligence (AI), and particularly the advances in machine learning (ML) and deep learning (DL) have led to disruptive innovations in radiology, pathology, genomics and other fields. Modern DL models feature millions of parameters that need to be learned from sufficiently large curated data sets in order to achieve clinical-grade accuracy, while being safe, fair, equitable and generalising well to unseen data[7].

Federated learning (FL) is a learning paradigm seeking to address the problem of data governance and privacy by training algorithms collaboratively without exchanging the data itself. Originally developed for different domains, such as mobile and edge device use cases, it recently gained traction for healthcare applications. Recent research has shown that models trained by FL can achieve performance levels comparable to ones trained on centrally hosted data sets and superior to models that only see isolated single-institutional data.

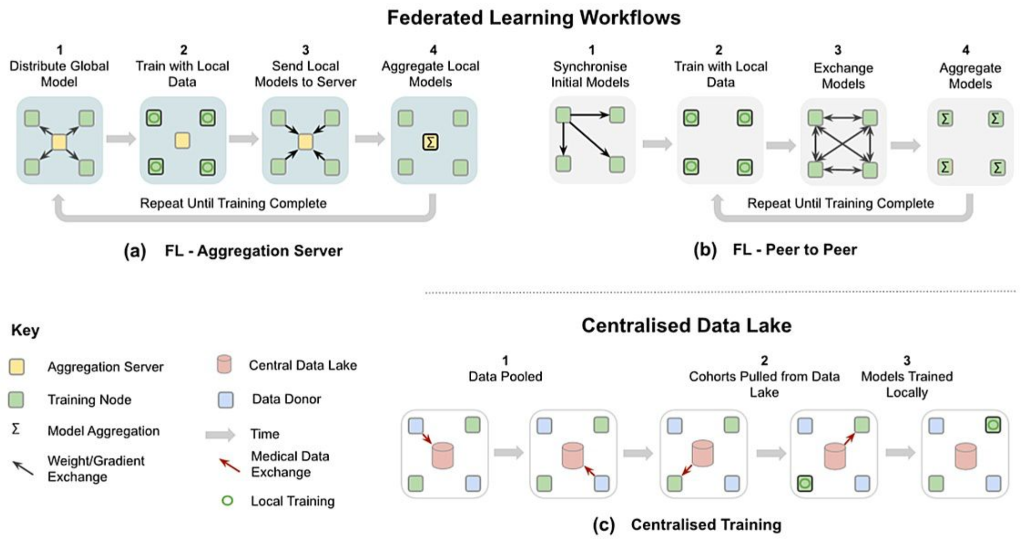


Figure 2.5: FL workflows and comparison with centralised architecture

2.5.1 The promise of Federated Efforts

The promise of FL is simple to address privacy and data governance challenges by enabling ML from non-co-located data. In a FL setting, each data controller not only defines its own governance processes and associated privacy policies, but also controls data access and has the ability to revoke it. This includes both the training, as well as the validation phase. FL implicitly offers a certain degree of privacy, as FL participants never directly access data from other institutions and only receive model parameters that are aggregated over several participants. In a FL workflow with aggregation server, the participating institutions can even remain unknown to each other.However,it has been shown that models.themselves can,under certain conditions, memorise information Therefore, mechanisms such as differential privacy or learning from encrypted data have been proposed to further enhance privacy in a FL setting.

2.5.2 Current FL efforts in Digital Health

Since FL is a general learning paradigm that removes the data pooling requirement for AI model development, the application range of FL spans the whole of AI for healthcare. By providing an opportunity to capture larger data variability and to analyse patients across different demographics, FL may enable disruptive innovations for the future but is also being employed right now.

The applicability and advantages of FL have also been demonstrated in the field of medical imaging, for whole-brain segmentation in MRI, as well as brain tumour segmentation. By linking healthcare institutions, not restricted to research centres, FL can have direct clinical impact.

2.5.3 Challenges and Considerations

A successful model training still depends on factors like data quality, bias and standardisation. These issues have to be solved for both federated and unfederated learning efforts via appropriate measures, such as careful study design, common protocols for data acquisition, structured reporting and sophisticated methodologies for discovering bias and hidden stratification.

- Data heterogeneity

  Medical data is particularly diverse not only because of the variety of modalities, dimensionality and characteristics in general, but even within a specific protocol due to factors such as acquisition differences, brand of the medical device or local demographics.

- Privacy and security

  Healthcare data is highly sensitive and must be protected accordingly, following appropriate confidentiality procedures. Therefore, some of the key considerations are the trade-offs, strategies and remaining risks regarding the privacy preserving potential of FL.

- Traceability and accountability

  As per all safety-critical applications, the reproducibility of a system is important for FL in healthcare. In contrast to centralised training, FL requires multiparty computations in environments that exhibit considerable variety in terms of hardware, software and networks. Traceability of all system assets including data access history, training configurations, and hyperparameter tuning throughout the training processes is thus mandatory.

2.5.4 System architecture

Healthcare institutional participants are equipped with relatively powerful computational resources and reliable, higher-throughput networks enabling training of larger models with many more local training steps, and sharing more model information between nodes.

The administration of such a federation can be realised in different ways. In situations requiring the most stringent data privacy between parties, training may operate via some sort of "honest broker" system, in which a trusted third party acts as the intermediary and facilitates access to data. This setup requires an independent entity controlling the overall system, which may not always be desirable, since it could involve additional cost and procedural viscosity. Additionally, in a trustless based architecture the platform operator may be cryptographically locked into being honest by means of a secure protocol, but this may introduce significant computational overheads.

ML, and particularly DL, has led to a wide range of innovations in the area of digital healthcare. By enabling multiple parties to train collaboratively without the need to exchange or centralise data sets, FL neatly addresses issues related to egress of sensitive medical data. As a consequence, it may open novel research and business avenues and has the potential to improve patient care globally.

2.6 Lung and Heart Sounds Analysis: State-of-the-Art and Future Trends

In this paper[8], we explore a variety of techniques and open questions that address the challenge of analysing heart sound disease more efficiently and effectively. We are analyzing a global system in which smartphones are used for monitoring, diagnosis, and giving medical support and assistance which is based on a large database.The current technology in treatment has brought improvements in quality of life and survival to heart-failure patients. On the other hand, improved survival and is accompanied by some of the additional health problems. Recent studies show that a substantial number of heart-failure patients die from causes other than the cardiac disease.

2.6.1 Heart Sound Characteristics

Some of the mechanisms used by detecting the heart sounds are generated includes opening or closing heart valves, flow of blood through the valve orifice,flow of blood into the ventricular chambers, and rubbing of cardiac surfaces.

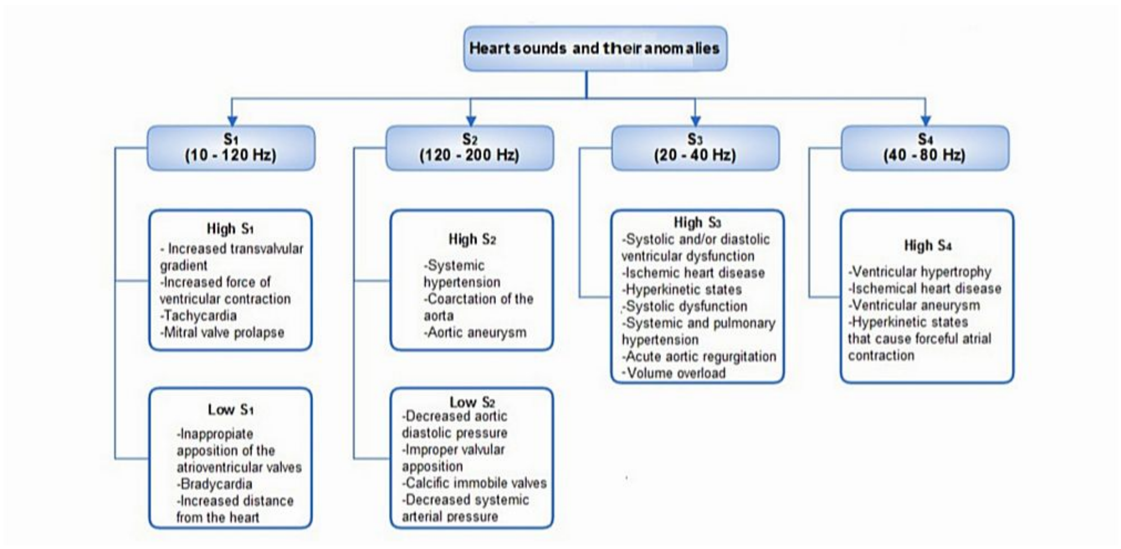

Heart sounds and their anomalies
S1 (10 - 120 Hz)
S2 (120 - 200 Hz)
S3 (20 - 40 Hz)
S4 (40 - 80 Hz)
High S1 - Increased transvalvular gradient -Increased force of ventricular contraction -Tachycardia -Mitral valve prolapse
Low S1 -Inappropiate apposition of the atrioventricular valves -Bradycardia -Increased distance from the heart
High S2 -Systemic hypertension -Coarctation of the aorta -Aortic aneurysm
Low S2 -Decreased aortic diastolic pressure -Improper valvular apposition -Calcific immobile valves -Decreased systemic arterial pressure
High S3 -Systolic and/or diastolic ventricular dysfunction -Ischemic heart disease -Hyperkinetic states -Systolic dysfunction -Systemic and pulmonary hypertension -Acute aortic regurgitation -Volume overload
High S4 -Ventricular hypertrophy -Ischemical heart disease -Ventricular aneurysm -Hyperkinetic states that cause forceful atrial contraction

Figure 2.6: Heart Sound Characteristics

2.6.2 Auscultation

Auscultation is a rapid,easy, effective, and noninvasive technique and it is used by trained physicians to diagnose respiratory and cardiac diseases.In recent stethoscopes it can be recorded and send the recorded sounds to a personal computer for processing and analysis. Auscultation with a stethoscope is a highly subjective process and depends on several factors including experience and skill of healthcare professionals and their ability to recognize different sound patterns. Computerized analysing methods will enables a systematic approach to the diagnosis of different respiratory or cardiac diseases.The computerized heart sound analysis takes place in different stages.

2.6.3 Heart Sound Databases

The heart sound databases are developed for the learning tools and enhance problem-solving skills in the area of respiratory medicine. The effort which is needed to document data for storage and sharing in a semi permanent manner is rarely available at the close of a research project.During the past few years, some websites contain rich educational material on heart sound databases which have been developed and can be used to train health-care professionals. They incorporate user manuals, listening tips, respiratory and cardiac sound recordings, waveforms, exercises for diagnosis training, and quizzes. These include Easy Auscultation Training, Practical Clinical Skills, The Auscultation Assistant, and SoundCloud. In addition to the online repositories and several books for understanding heart sounds and murmurs are available. Computerized heart sound analysis represents the advance technologies in diagnosing, monitoring, treatment, and visualization signaling of respiratory and cardiac pathologies. However, due to a lack of published guidelines, significant differences exist among various laboratories, such as the use of different ways of sensors to acquire signals and signal-processing techniques.

2.7 Attack Detection using Federated Learning in medical cyber-physical systems

Cyber-Physical Systems (MCPS) are networked systems of medical devices that provide seamless integration of physical and computation components in healthcare environments to deliver high quality care by enabling continuous monitoring and treatment. As MCPS store sensitive medical data and personal health data, security breaches and unauthorized access to this information can lead to severe repercussions for both the patient and hospital in the form of loss of privacy, abuse, physical harm and liability. The heterogeneity of devices involved in these systems (such as body sensor nodes and mobile devices) introduce large attack surfaces and hence necessitate the design of effective security solutions for these environment.MCPS helps constantly monitor and analyze in- formation gathered from medical devices, infer the patient's health condition for diagnosis, and provides timely treatment either through direct feedback from healthcare providers or through automated treatments using medical actuators.

2.7.1 Network Architecture

The MCPS network consists of medical devices that are basically wireless body sensor nodes placed on the patient's body; mobile devices that acts as a local gateway to the medical devices and a back-end server at the hospital. The sensor nodes are used to collect patient vitals and administer drugs, such as insulin or anesthetics. The mobile device acts as a gateway for the medical devices. The sensor nodes communicate with the mobile device wirelessly using a short-range communication protocol, such as Bluetooth or Zigbee. The server is also connected to the Internet via a wired connection to the hospital's gateway.

The server is responsible for handling messages transmitted from the mobile device as well as relaying messages back to patient's mobile devices. mobile device collects, aggregates, and keeps a history of node measurements, such as blood pressure.The system follows a client-server topology between patient's mobile devices and the hospital server. This ensures scalability as adding more or new mobile devices in the hospital network increases message traffic and logic at the server linearly.

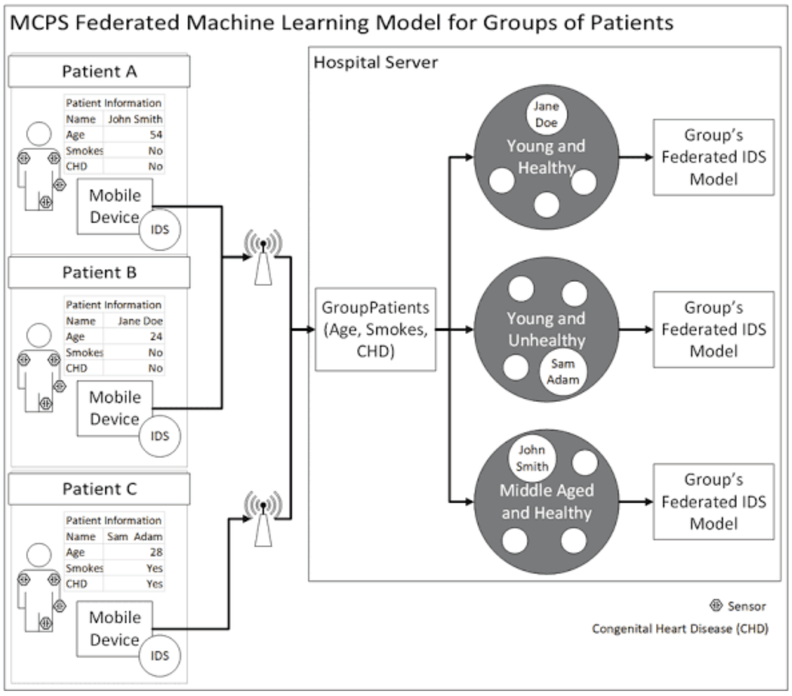


Figure 2.7: FL model for a group of patients

2.7.2 Clustering of Patients

The clustering process occurs during registration of a mobile device with the hospital sever (Figure 2). After a mobile device has been assigned to a group, it only receives and contributes to that groups model.The attack detection process begins with a mobile device registering with the server. Devices are then clustered into different groups based on their patient history. Determining the correct number of clusters will depend on several factors including the number of mobile devices in network and the number of parameters used for clustering process. Each group has a federated model stored on the server. The mobile device then downloads the federated model from the server and continues to learn and update a new model using the patient's data.

2.7.3 Updating the Model

Federated Learning is a distributed machine learning algorithm that builds a global model by averaging weights w across many devices over several communication rounds t. Each neuron in the hidden layer has a transfer function, denoted by f, that takes each feature in a sample (In1...Ini) and multiplies it by its weight (IW1,1...IWi,1) plus a bias (B1). The weights are modified during training.when a patient registers with the server they are clustered into a group who share a single IDS model. After determining the number of patients to use, the server selects patients from the group at random and without replacement. The mobile devices of the subset are then sent a message by the server to send their model's weights to the server. The server keeps a record of each mobile device's weights and at an end of a communication round calculates the next federated model w t+1.

2.7.4 Experimental Setup

The MIMIC dataset from PhysioNet is used for evaluation of the proposed system. This dataset has six features, which are typically displayed on an ICU monitor, including elapsed time, arterial blood pressure, heart rate, pulse, respiratory rate, and blood oxygen concentration. The dataset has 121 records with each record having about 35-40 hours of monitored activity. The ma- chine learning was executed using Sci-kit Learn's Multi-Layer Perceptron running on Raspberry Pi's. Each Raspberry Pi is associated with a patient who is generating data for the device to train the IDS. All attack were simulated using new patient data; data the ML model has not been trained on. Half of the data samples are modified to simulate attacks and half remain unperturbed. We also simulated the system using MATLAB by following the same process as above, where the Raspberry Pi's were replaced with MATLAB objects to represent each device. As a result, we can conclude that a federated learning based IDS can train on more data, increasing accuracy and lowering FPR, while decreasing the amount of time and computation needed of an individual mobile device.

2.8 An open access database for the evaluation of heart sound algorithms

Cardiovascular diseases (CVDs) continue to be the leading cause of morbidity and mortality worldwide. One of the first steps in evaluating the cardiovascular system in clinical practice is physical examination. Auscultation of the heart sounds is an essential part of the physical examination and may reveal many pathologic cardiac conditions such as arrhythmias, valve disease, heart failure, and more. Heart sounds provide important initial clues in disease evaluation, serve as a guide for further diagnostic examination, and thus play an important role in the early detection for CVDs. During the cardiac cycle, the heart first experiences electrical activation, which then leads to mechanical activity in the form of atrial and ventricular in which the valves can be best heard in the following locations:

- Aortic area—centred at the second right intercostal space.
- Pulmonic area—in the second intercostal space along the left sternal border.
- Tricuspid area—in the fourth intercostal space along the left sternal edge.
- Mitral area—at the cardiac apex, in the fifth intercostal space on the midclavicular line.

Fundamental heart sounds (FHSs) usually include the first (S1) and second (S2) heart sounds. S1 occurs at the beginning of isovolumetric ventricular contraction, when already closed mitral and tricuspid valves suddenly reach their elastic limit due to the rapid increase

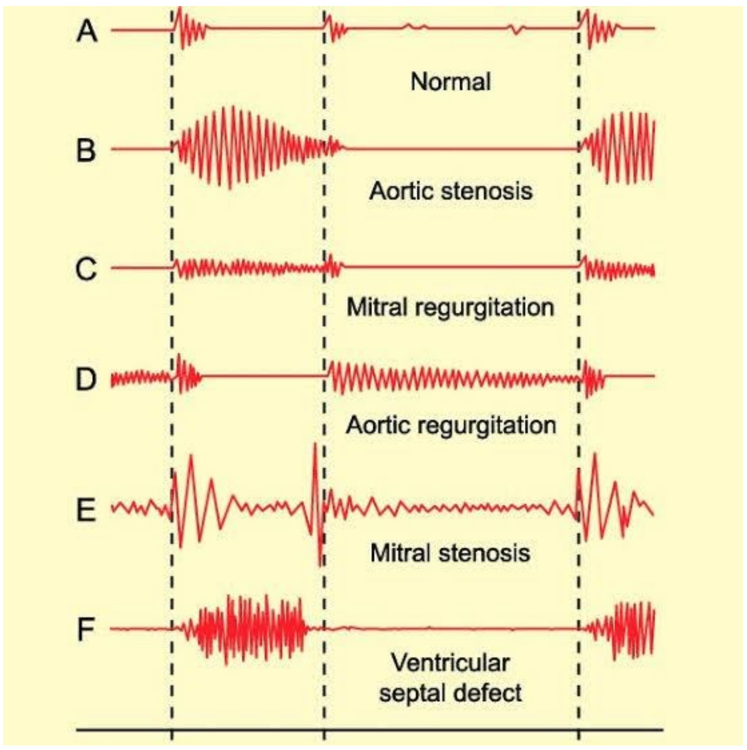


Figure 2.8: Normal vs Abnormal PCGs

in pressure within the ventricles. S2 occurs at the beginning of diastole with the closure of the aortic and pulmonic valves. While the FHSs are the most recognizable sounds of the heart cycle, the mechanical activity of the heart may also cause other audible sounds, such as the third heart sound (S3), the fourth heart sound (S4), systolic ejection, mid-systolic click (MC), the diastolic sound or opening snap (OS), as well as heart murmurs caused by turbulent, high-velocity flow of blood.

2.8.1 Classification procedure of heart sounds

2.8.2 Step 1 Pre-processing

- To assess the signal quality
- To filter out baseline changes and high frequency noises
- To extract relevant features

Step 2 Segmentation

- To delineate the start and end of each phase of the heart beat (S1, systolic, S2, diastolic, etc.)

Step 3 Classification then Clinical

- To map the features for each segmented phase of the beat to a known phase or sound, or the entire recording to a pathology. Sometimes quality classifications are made on sections or entire recordings.

These typical three steps for automated analysis of heart sound in clinical applications.

2.8.3 Potential benefits of Public Heart sound data

The public release of the heart sound database has many potential benefits to a wide range of users. First, those who lack access to well-characterized real clinical signals may benefit from access to these data for developing prototype algorithms. The availability of these data can encourage researchers from a variety of backgrounds to develop innovative methods to tackle problems in heart sound signal processing that they might not otherwise have attempted. An additional benefit is that the data can be re-evaluated with new advances in machine learning and signal processing as they become available.These databases have value in medical and biomedical engineering education by providing well-documented heart sound recordings from both healthy subjects and patients with a variety of clinically significant diseases.

CHAPTER 3

METHODOLOGY

3.1 Modules

3.1.1 Pre-Processing

Our project intends to classify the heart condition of the patients by recording their heartbeat sound. The input to our system is from an electronic stethoscope in audio format. Our system takes these audio signals as the input and classifies them as either normal or abnormal[12].

Audio machine learning applications used to depend on traditional digital signal processing techniques to extract features. For instance, to understand human speech, audio signals could be analyzed using phonetics concepts to extract elements like phonemes. All of this required a lot of domain-specific expertise to solve these problems and tune the system for better performance. In recent years, Deep Learning has become more and more ubiquitous and with deep learning, the traditional audio processing techniques are no longer needed, and we can rely on standard data preparation without requiring a lot of manual and custom generation of features. So with Deep Learning, we don't actually need to deal with audio data in its raw form. Instead, the audio data can be converted to images and then we can use a standard CNN architecture to process those images.

The main function of the preprocessing module is to convert the audio signals coming from the electronic stethoscope to images. This is done by generating Spectrograms from the audio. The Spectrum is the set of frequencies that are combined together to produce a signal. The Spectrum plots all of the frequencies that are present in the signal along with the strength or amplitude of each frequency.

A Spectrogram of a signal plots its Spectrum over time and is like a 'photograph' of the signal. It plots Time on the x-axis and frequency on the y-axis. It is as though we took the Spectrum again and again at different instances in time, and then joined them all together into a single plot. It uses different colours to indicate the Amplitude or strength of each frequency. The brighter the colour the higher the energy of the signal. Each vertical 'slice' of the Spectrogram is essentially the Spectrum of the signal at that instant in time and shows how the signal strength is distributed in every frequency found in the signal at that instant. The first picture displays the signal in the Time domain i.e, Amplitude vs Time. It gives us a sense of how loud or quiet a clip is at any point in time, but it gives us very little information about which frequencies are present. The second picture is the Spectrogram and displays the signal in the frequency domain.

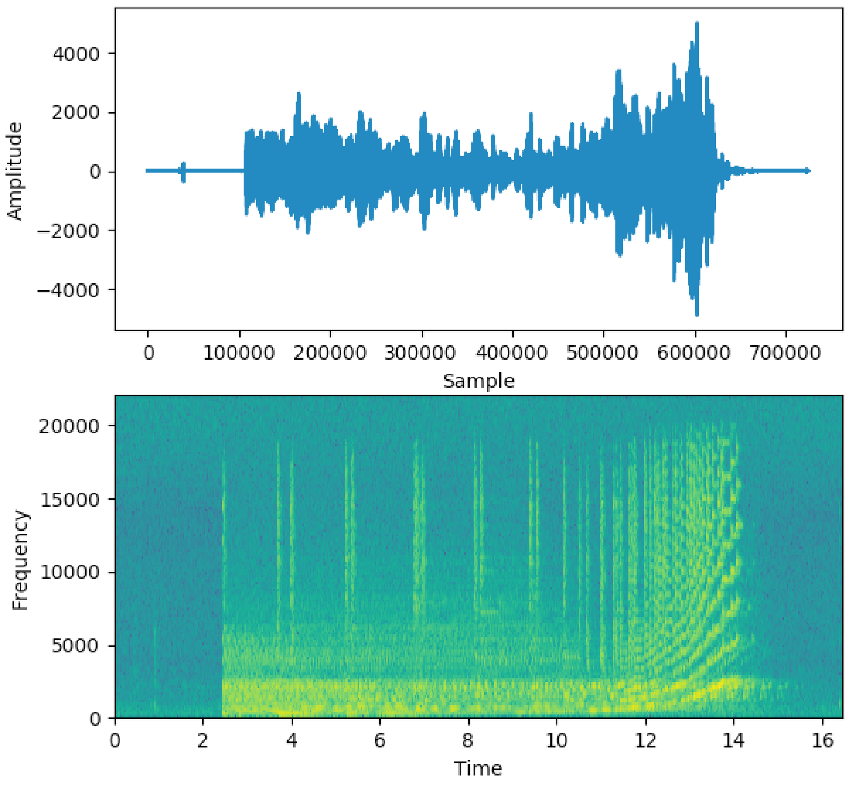


Figure 3.1: Audio signal & generated Spectrogram

In our project, we are using Mel Spectrograms. A Mel Spectrogram makes two important changes relative to a regular Spectrogram that plots Frequency vs Time.

- It uses the Mel Scale instead of Frequency on the y-axis.
- It uses the Decibel Scale instead of Amplitude to indicate colours

The second function in the preprocessing module is to convert the generated Mel spectrogram to grayscale. The Mel spectrogram image generated on the first stage of the preprocessing module is a 3 channel image but a grayscale spectrogram contains all of the relevant information in its pixel intensities. So by converting it to grayscale we are reducing the size of the input by 60% and also improve the performance of the system.

The third and last stage of the preprocessing module consists of the resizing function. The audio files coming to the system will be different in length. So while generating the spectrogram, the width representing the X-axis (Time) of the spectrogram depends on the length of the audio. In this stage, the greyscale Mel-spectrogram is resized to a predefined size that is compatible with the Deep Learning model architecture.

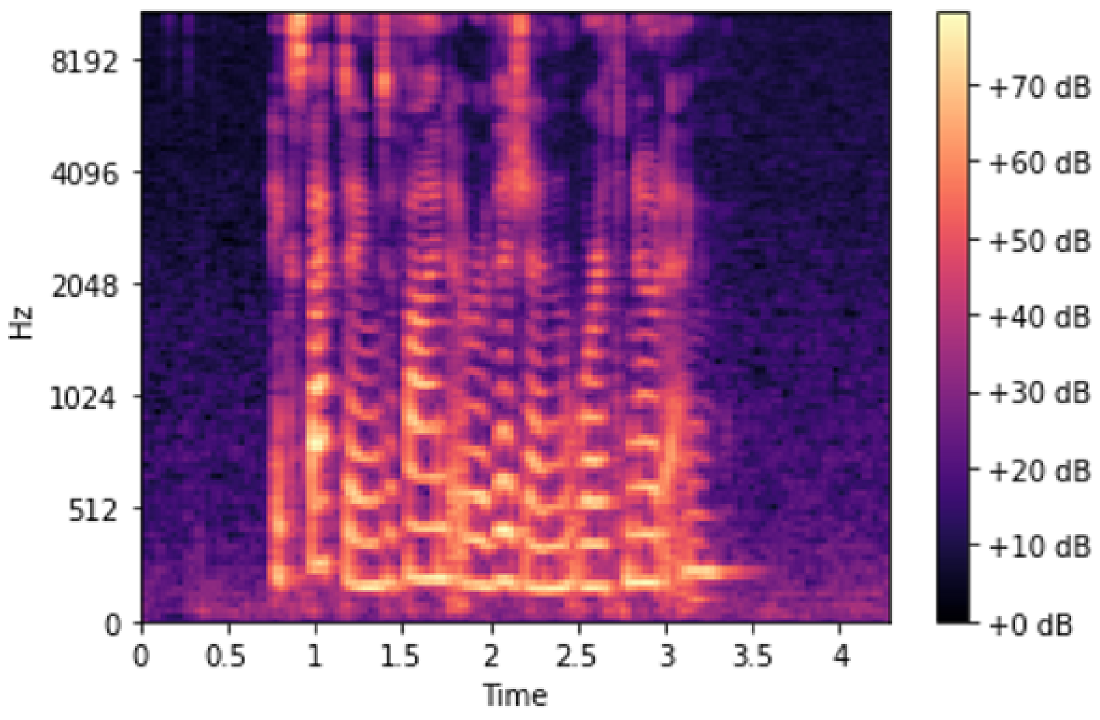


Figure 3.2: Mel-Spectrogram

3.1.2 Deep Learning Model

We have used a Convolutional Neural Network(CNN) to classify the pre-processed Mel-spectrogram images of heartbeat sounds. A CNN is a class of deep neural network, most commonly applied to analyze visual imagery. This neural network applies a series of convolutions on input image data to reduce the input size and obtain feature maps. The input image to this layer are Mel-spectrograms that were pre-processed as described in the above section. The output produced is a tensor of values that depict the classifier's confidence about the image being classified to the corresponding class label.

3.1.3 Federated Learning

Federated learning is a machine learning technique that trains an algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging them. The word 'Federate' describes the organisation or grouping of multiple entities with similar motives. This meaning is apparent in the way of implementation of Federated Learning in our system

The concept of FL was introduced by Google in 2016[11] to solve the various complications that arose due to standard Machine Learning methodologies mainly in the area of Privacy & Confidentiality. Traditionally the data used for ML required to be stored in a Central Server to

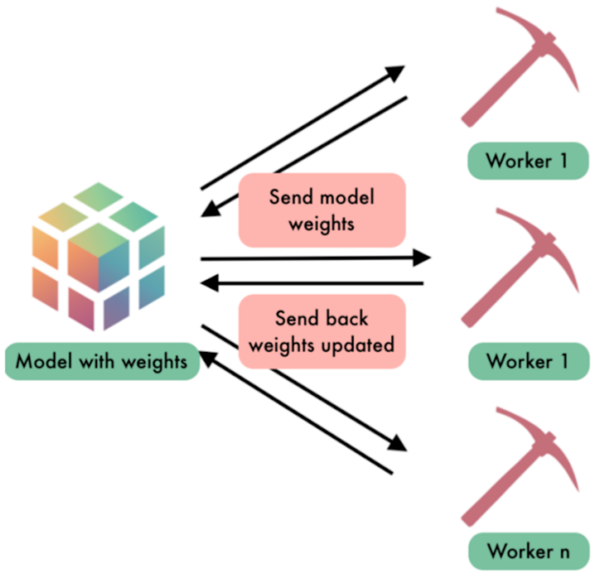


Figure 3.3: Federated learning

facilitate the creation or training of a model that could then summarize the data and provide predictions for future unseen data. This process has a requirement to create a central dataset for model training purposes. This requirement makes it impractical to develop an ML model using privacy-sensitive data like Medical Data, as in our case. The usage of Medical and other privacy-sensitive data is done by following certain guidelines like “Patient Safety: Achieving a New Standard for Care”[10] that restricts the exchange of patient data.

The issues regarding the utilization of Privacy sensitive data in ML was initially addressed by the introduction of Distributed Learning. This enables the involvement of multiple nodes for the training process with their own subset of data. The models participate in the training process that is orchestrated by a central server, which receives gradients computed after each iteration and creates an updated model. This method of learning was initially accepted for usage with medical data as ‘Distributed-mini-batch’[9] algorithm.

The above-mentioned architecture had many issues in continuous usage because of the many problems that arose due to the high requirement of paralleling of data for gradient computation and synchronization of gradient exchange.

Federated Learning was introduced to address the problems stated above giving more importance to data privacy and confidentiality. It uses less communication and is more resilient than distributed mini-batching. Moreover, this approach is highly privacy-preserving, making it suitable for applying in the medical domain.

The Federated Learning Module implemented in our system uses Weighted Averaging to average or combine multiple model parameters and feed the aggregated main model. The entire process of Federated Learning requires a central entity that admits clients to the procedure and manages the entire process. Our system implements Model-Centric FL, i.e the Model is hosted centrally and the data is available locally with clients. This implementation preserves the privacy of patients as there is no exchange of patient data.

The architecture & implementation of Federated Learning is discussed much more deeply in the implementation part.

3.1.4 User Interface

User Interface is one of the most important modules in our system. This is the module that is visible to the outside world where users can interact with our system. We have two different groups of users in our system. The first group is the doctors, who are the primary users of the system. The second group consists of the patients.

We have designed and developed two separate UIs for the users by considering the two different use-cases. For doctors, we have developed a Web-application that has all the functionalities needed. This Web-application includes the Deep Learning module which we have developed for classifying the heart condition of the patients. Doctors can add new patients to the application and record their heartbeat sound directly inside the Web application. This recording is used to predict the heart condition of the patients. This prediction plus the details of the patient will be stored on a database. In addition to that, Doctors can also prescribe medications and provide remarks through the web application.

The second UI we developed is for the patients. It is a simple mobile application. Patients can use this application to view their medical data. This data includes the predicted heart condition of the patient, medication prescribed by the doctor and remarks from the doctor. Each user will be having a unique ‘Prescription ID’ which is used to access their profile on the app. As a part of security, we have also included the date of consultation for the authentication of the user. So users can simply input the unique “Prescription ID” and the date of consultation and view their data.

3.2 System Requirements & Specifications

3.2.1 Introduction

This document describes the functionality and basic idea behind the project “Cardio vascular diagnosis using Federated Learning”.

3.2.2 Purpose

Cardiovascular diseases (CVDs) continue to be the leading cause of morbidity and mortality worldwide according to WHO. Automated classification of cardiovascular sounds has the potential to detect abnormalities in the early stages of a cardiovascular dysfunction and thus enhance the effectiveness of decision making. One of the first steps in evaluating the cardiovascular system in clinical practice is physical examination, which is dependant on the level of expertise of the physician. Also, because of data-privacy restrictions, conventional Machine Learning based technologies are not able to achieve the level of accuracy, comparable to a human counterpart. Our system, which uses Federated Learning methodology proves to be a solution to most of the currently prevailing restrictions.

3.2.3 Description

An Electronic stethoscope is used to auscultate a patients heartbeat sound and obtain the recorded sample of the same. This recorded sample is passed to our system to obtain a diagnosis of the patient’s heart condition. Federated Learning training methodology is utilized to obtain inferences from many such implementations of the system to improve the overall efficiency of the system over time without sharing private patient data.

3.2.4 Functional requirements

1. Detect abnormal heart sound using heart sound recording of patient
2. Produce a diagnosis within a 60s time

3.2.5 Non Functional Requirements

1. Privacy The system should respect the confidentiality of medical data and should maintain privacy
2. Security

   The system should maintain the security of data used.

3.2.6 Technical requirements

- Linux OS

Linux is a family of open-source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged in a Linux distribution.

- Python 3.6

  Python is the most popular dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support by providing a smooth platform for integration with other languages and tools, and also comes with extensive standard libraries. Its simplicity makes it possible to be learned in a few days. It encourages higher quality, more maintainable code.

- PyTorch

  PyTorch is an open-source library for Python, based on Torch, mainly focussing on machine learning, used for applications such as natural language processing. Facebook's artificial intelligence research group developed PyTorch and Uber's ”Pyro” software for probabilistic programming is built on it. PyTorch primarily provides two high-level features: Tensor computation (like NumPy) along with strong GPU acceleration Deep Neural Networks.

- Google Colaboratory (Colab)

  Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

- Jupyter Notebook

  The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

- OpenMined PyGrid

  PyGrid is a peer-to-peer network of data owners and data scientists who can collectively train AI models using PySyft. PyGrid is also the central server for conducting both model-centric and data-centric federated learning.

- OpenMined PySyft

  OpenMined PySyft is an open-source framework for Machine Learning and other computations on decentralized data PySyft decouples private data from model training, using Federated Learning Encrypted Computation within the main Deep Learning frameworks like PyTorch.

- Docker

  Docker is a popular open-source project written in go and developed by Dotcloud (A PaaS Company).Docker has well-defined wrapper components that make packaging applications easy. It is basically a container engine that uses the Linux Kernel features like namespaces and control groups to create containers on top of an operating system. We used Docker to host PyGrid deployed a PyTorch Deep Learning(DL) model in it.

- SCIKIT learn

  Scikit-learn is a platform that provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. The library is built upon and supported by the SciPy (Scientific Python) that must be installed before you can use sci-kit-learn.
  This stack that includes:

  - SciPy: Fundamental library for scientific computing
  - Matplotlib: Comprehensive 2D/3D plotting
  - IPython: Enhanced interactive console
  - Sympy: Symbolic mathematics
  - Pandas: Data structures and analysis

  The extensions or modules for SciPy are conventionally named SciKits.

3.3 Data Flow Diagrams

3.3.1 Data Flow Diagram- Level 0

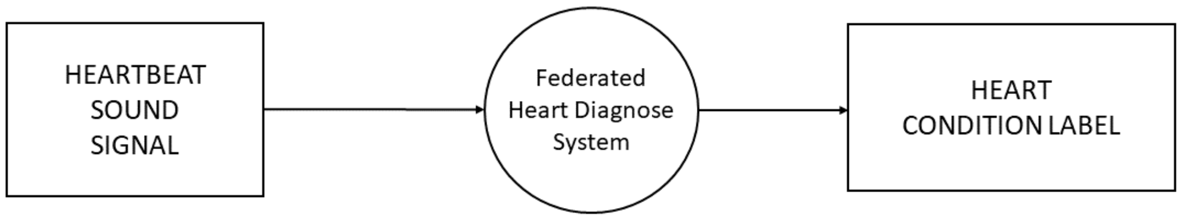


Figure 3.4: DFD- Level 0

3.3.2 Data Flow Diagram- Level 1

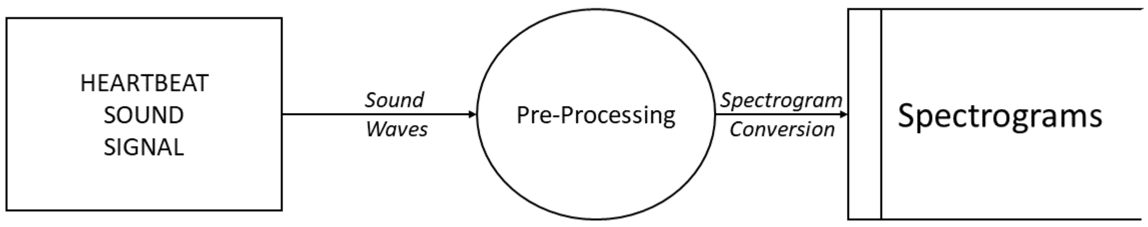


Figure 3.5: DFD- Level 1: Pre-Processing

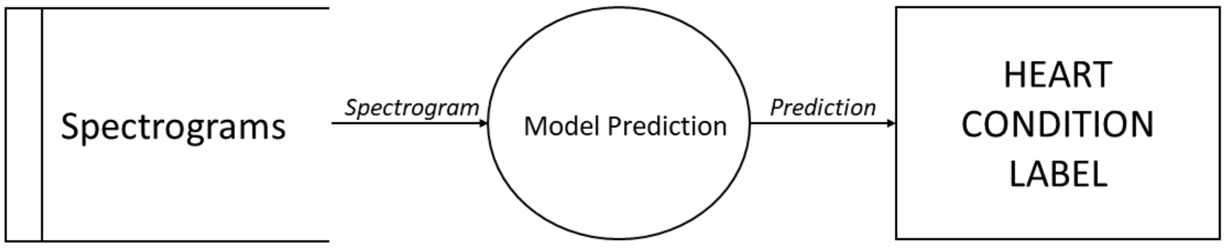


Figure 3.6: DFD- Level 1: Prediction

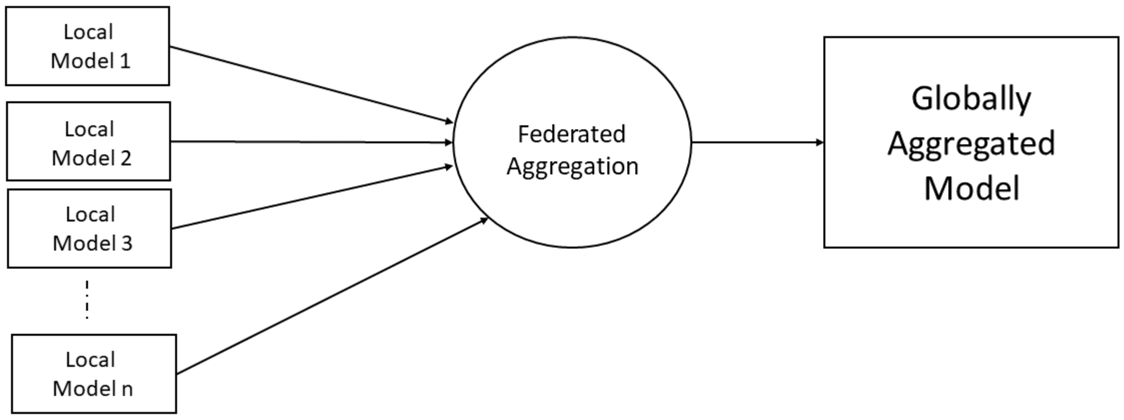


Figure 3.7: DFD- Level 1: Federated Aggregation

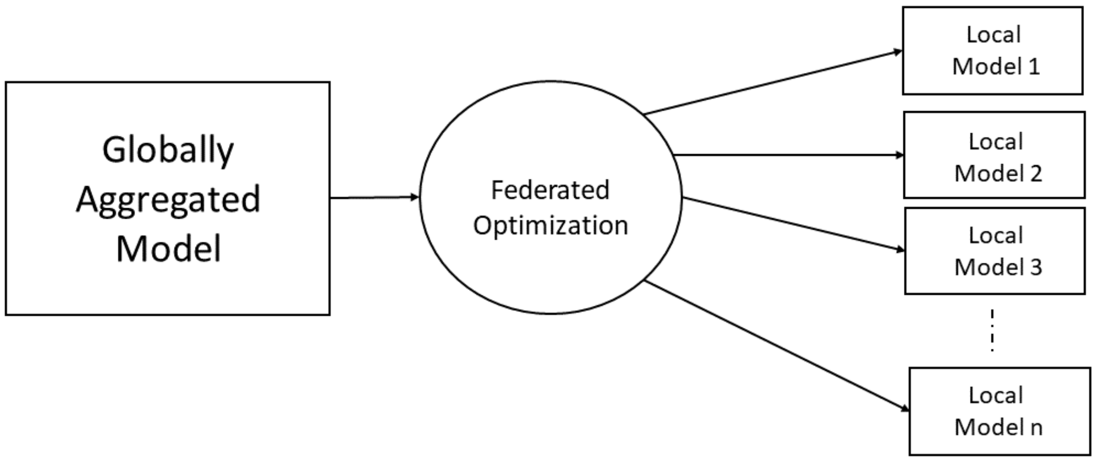


Figure 3.8: DFD- Level 1: Federated Optimization

3.3.3 Data Flow Diagram- Level 2

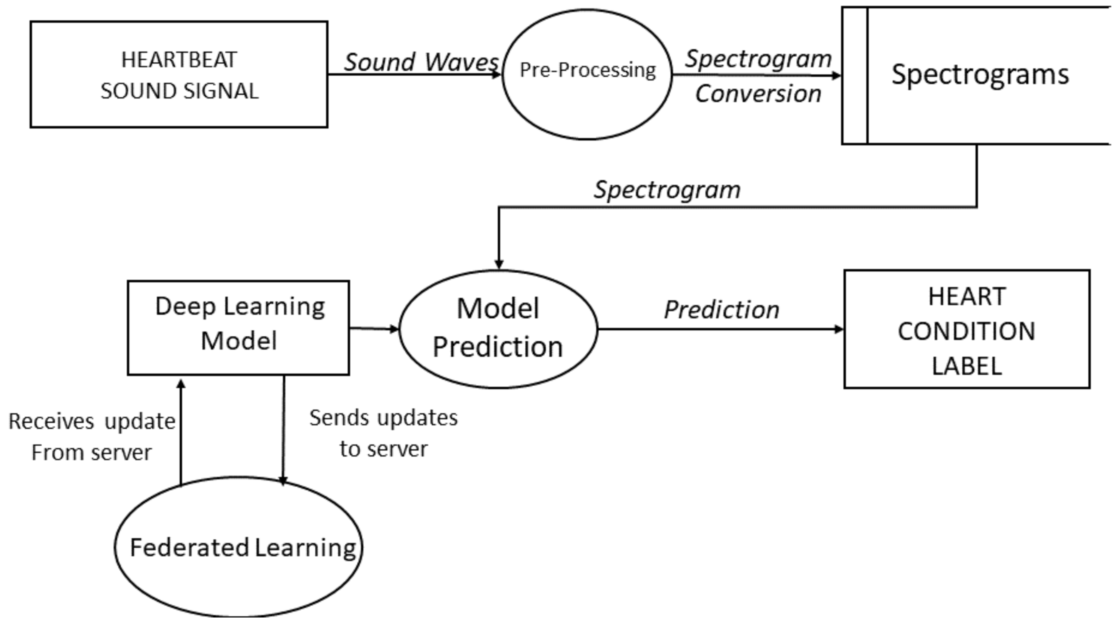


Figure 3.9: DFD- Level 2: System

3.4 Architecture

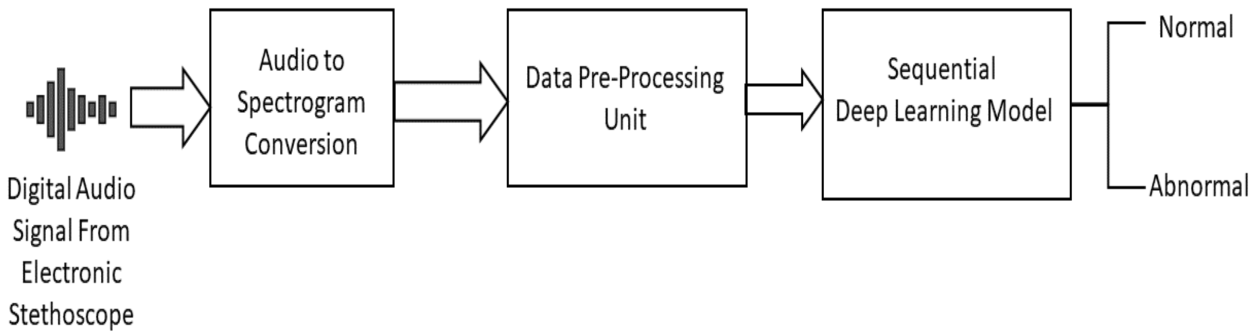

Digital Audio Signal From Electronic Stethoscope
Audio to Spectrogram Conversion
Data Pre-Processing Unit
Sequential Deep Learning Model
Normal
Abnormal

Figure 3.10: Distributed Device Architecture

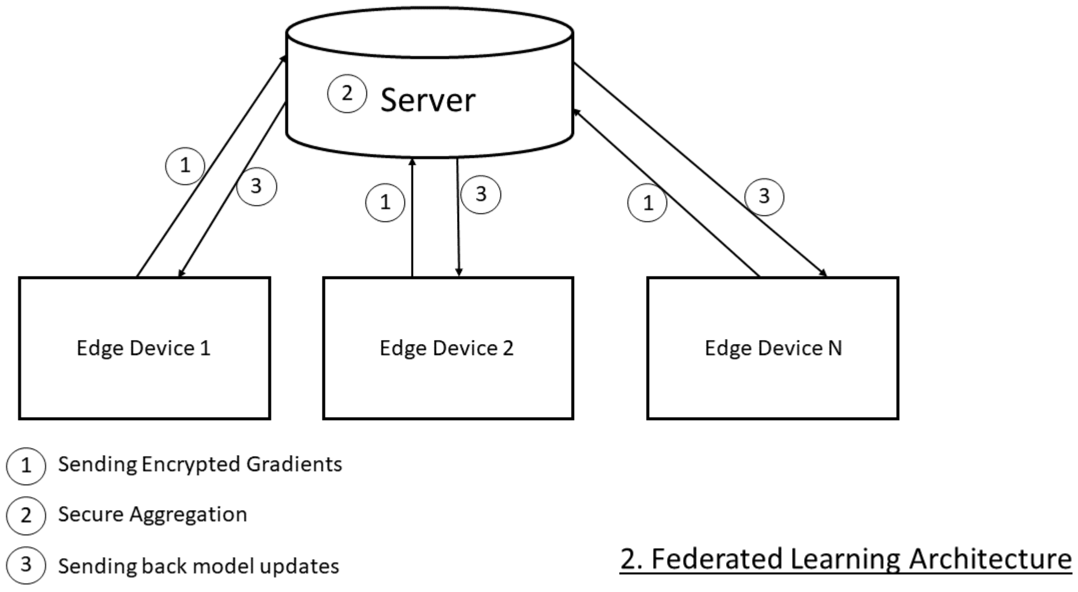

2 Server
1
3
1
3
1
3
Edge Device 1
Edge Device 2
Edge Device N
1 Sending Encrypted Gradients
2 Secure Aggregation
3 Sending back model updates
2. Federated Learning Architecture

Figure 3.11: Federated Learning Architecture

3.5 Use Case Diagram

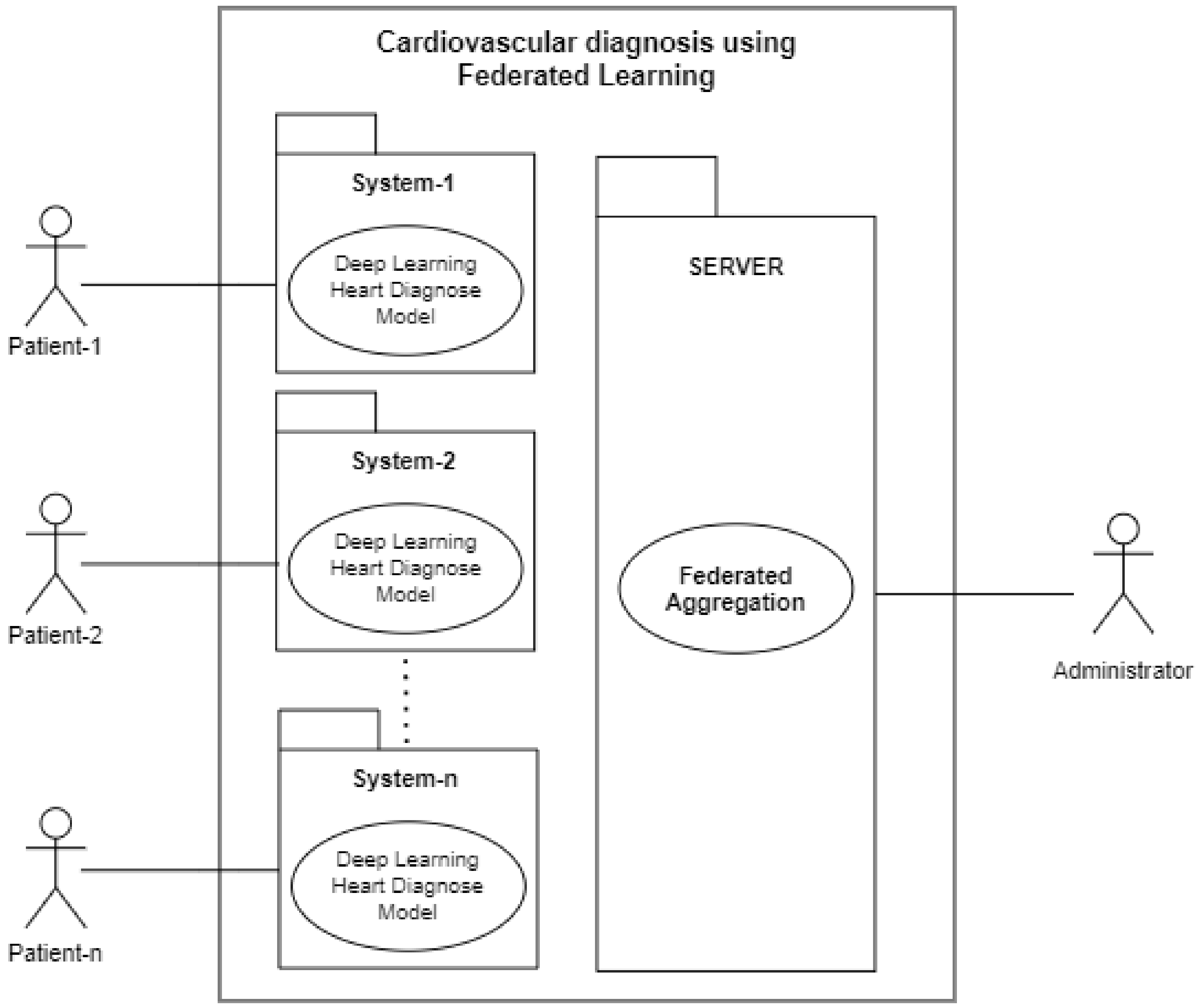

Cardiovascular diagnosis using Federated Learning
System-1
Deep Learning Heart Diagnose Model
System-2
Deep Learning Heart Diagnose Model
System-n
Deep Learning Heart Diagnose Model
SERVER
Federated Aggregation
Patient-1
Patient-2
Patient-n
Administrator

Figure 3.12: Use Case Diagram

3.6 Implementation

3.6.1 Pre-Processing

For the implementation of the preprocessing module, we are using different python libraries like Librosa for audio analysis and spectrogram generation, Skimage for resizing and saving the generated spectrogram images and Matplotlib for plotting the generated spectrogram.

The function "Mel-spectrogram" on the above screenshot contains all the functions of the preprocessing module. Initially, we are loading the audio file using the Librosa python library this audio signal is in the Time domain i.e. Amplitude vs Time this audio signal is the time domain is plotted below.

Pre-processing

```
import librosa
import numpy
import skimage.io
from skimage.transform import resize


def mel_spectrogram(wav_file, save_path):
  signal,sr = librosa.load(wav_file)
  spec_gram = librosa.feature.melspectrogram(signal,sr, fmax=2000)
  fig, ax = plt.subplots()
  S_dB = librosa.power_to_db(spec_gram, ref=np.max)

  img = librosa.display.specshow(S_dB)
  plt.axis('off')
  plt.savefig(save_path, bbox_inches='tight',pad_inches=0)

  #Resize to requirement
  img = Image.open(save_path)
  img = img.resize((698, 234))
  img.save(save_path)
```

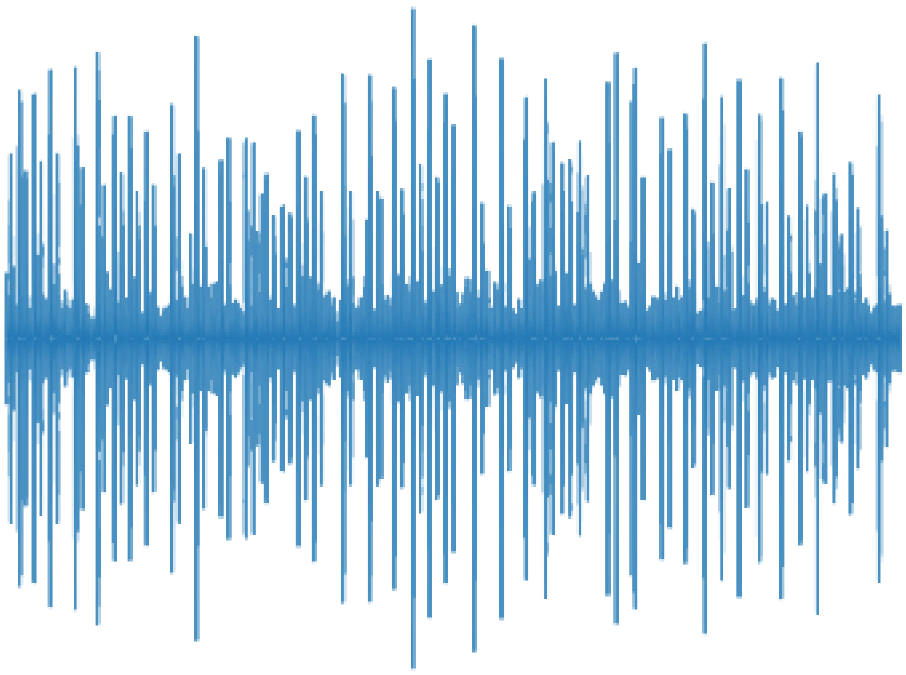


Figure 3.13: Audio signals in time domain.

The audio signal is converted to a Mel-spectrogram using the Librosa library. This Mel-spectrogram is plotted below.

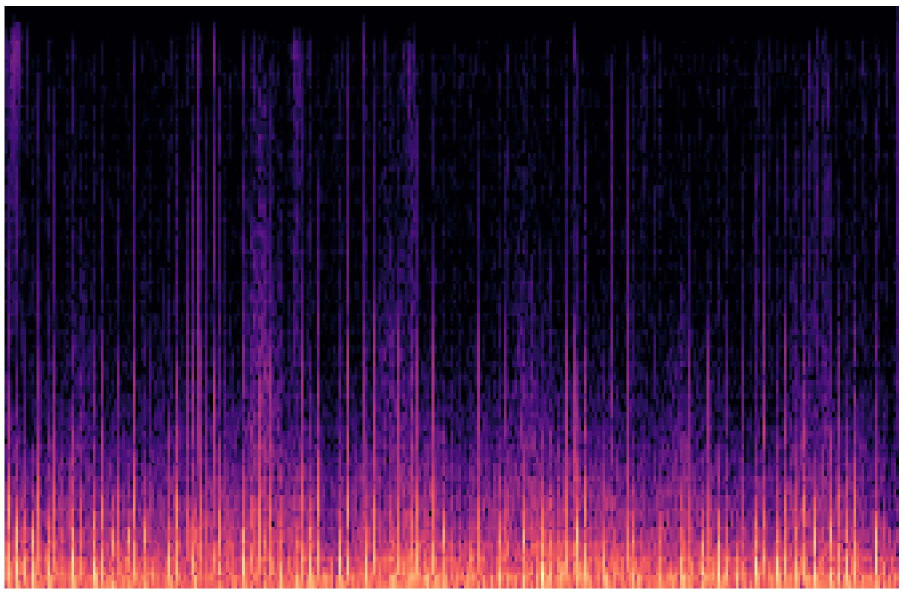


Figure 3.14: Mel-Spectrogram of corresponding audio.

The current Mel-spectrogram is a 3 channel RGB image. This RGB image is plotted below after converting to a gray-scale image.

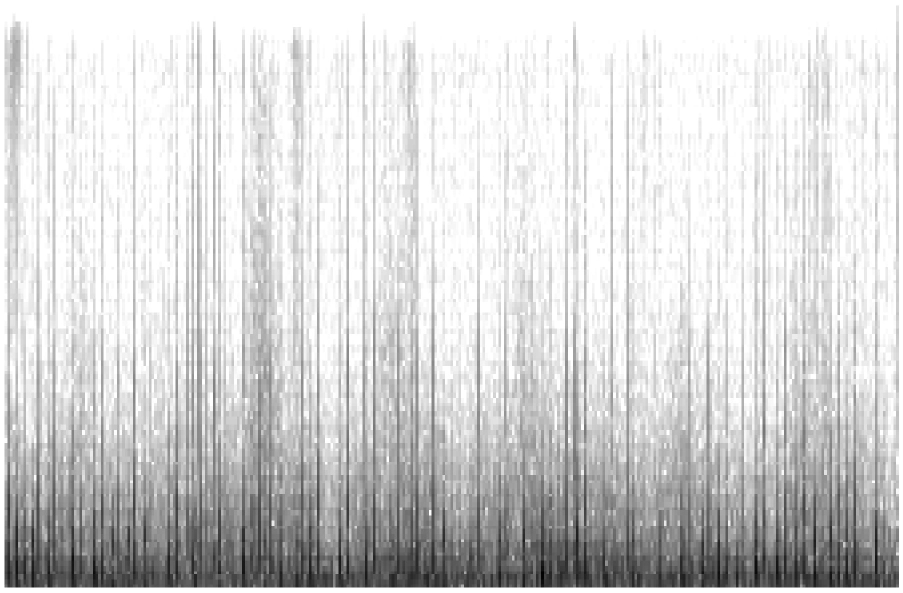


Figure 3.15: Gray-scale Mel-spectrogram.

The size of the generated gray-scale Mel-spectrogram image will depend on the length of the audio signal. This image is converted to a standard size that is compatible for the model.

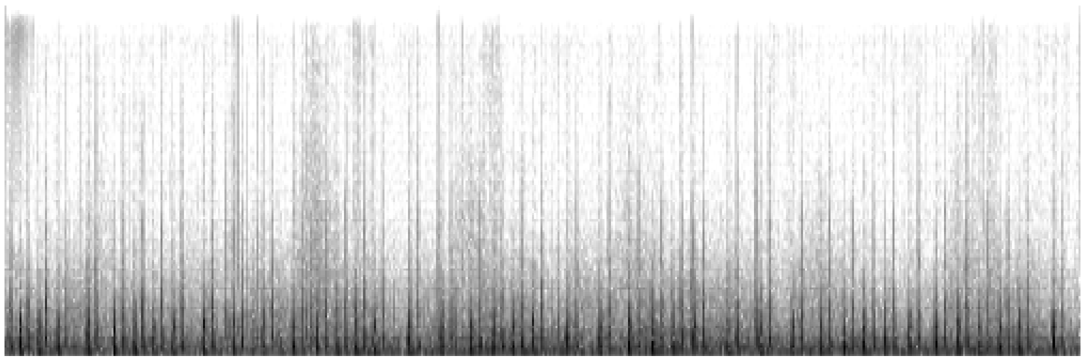


Figure 3.16: Resized gray-scale Mel-spectrogram.

3.6.2 Deep Learning Model

The CNN we built contain the following layers:

- Convolutional Layer

  This is the basic component of a CNN that creates feature maps. It is used to map features using various filters. It also reduces the overall complexity of the input by simplifying or reducing the dimensions of it.

- ReLU layer

  This is an activation layer that applies the 'Rectified Linear Unit' activation on the input. The basic process that happens is that the negative inputs get mapped to zero and the positive inputs remain the same. This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the convolutional layer.

- BatchNormalization Layer

  This layer standardizes the inputs to a layer, for each mini-batch. This has the effect of stabilizing the learning process and reducing the number of training epochs required to train deep networks.

- Flatten Layer

  This layer converts the entire set of matrices to a one dimensional array of values, i.e it flattens the matrix

- Linear Layer

  This is the output layer that maps the network's output to the desired number of classes. In our case the number of neurons is 2, because we have two classes, normal & abnormal.

Deep Learning Model - Architecture

```
class model_HeartCNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.Conv2d_1 = nn.Conv2d(in_channels=1,
        out_channels=20,kernel_size=5, stride=2)
        self.ReLU_1 = nn.ReLU()
        self.BatchNorm2d_1 = nn.BatchNorm2d(20)

        self.Conv2d_2 = nn.Conv2d(in_channels=20,
        out_channels=20,kernel_size=5, stride=2)
        self.ReLU_2 = nn.ReLU()
        self.BatchNorm2d_2 = nn.BatchNorm2d(20)

        self.Flatten_1 = nn.Flatten()
        self.Linear_4=nn.Linear(in_features=192640,
        out_features=2)
        def forward(self, x):
            x_reshaped = torch.reshape(x,(-1, 1, 234, 698))
            x = self.Conv2d_1(x_reshaped)
            x = self.ReLU_1(x)
            x = self.BatchNorm2d_1(x)

            x = self.Conv2d_2(x)
            x = self.ReLU_1(x)
            x = self.BatchNorm2d_2(x)

            x = self.Flatten_1(x)
            output = self.Linear_4(x)
            return output
```

Training & Testing

Training the model is the process where the weights and biases for the internal layers are decided. We used a DataLoader to feed the dataset into the model for training. The Dataloader receives a dataset that has been transformed from the integer array of pixel values to corresponding 'Tensor' formats for processing. The transformations also include the resize function where the images get resized to match the Model's architecture definitions.

```
training_dataset_path = "/content/dataset_2/training"
validation_dataset_path = "/content/dataset_2/validation"
img_transforms = transforms.Compose([transforms.Grayscale
(num_output_channels=1),
transforms.Resize((234, 698)), transforms.ToTensor(),
transforms.Normalize((0.0,), (0.5,))])
train_set = torchvision.datasets.ImageFolder
(root = training_dataset_path, transform=img_transforms)
test_set = torchvision.datasets.ImageFolder
(root = validation_dataset_path, transform=img_transforms)
```

The DataLoader also has a weighted sampler that samples the images according to the number of images in the dataset. Various configurations including batchsize are defined in the loader. The network is then trained for 10 epochs using the data. Back-propagation is done during the training process to correct and minimize the error. The model obtained is then tested for accuracy and other evaluation metrics. The metrics are calculated using the normal formula using the parameters like True Positives(correctly predicted positives), True Negatives(correctly predicted negatives), False Positives(mispredicted positives) and False Negatives(mispredicted negatives).

```
train_loader = DataLoader(dataset=train_set, shuffle=False,
batch_size=12, sampler=weighted_sampler)
test_loader = DataLoader(dataset=test_set, shuffle=False,
batch_size=12)

def TrainModel(model,epochs):
    optimizer = optim.Adam(model.parameters(), lr=0.0015)
    # defining the loss function
    criterion = nn.CrossEntropyLoss()
    # checking if GPU is available
    if torch.cuda.is_available():
        model = model.cuda()
        criterion = criterion.cuda()

    for i in range(epochs):
        running_loss = 0
        for images, labels in train_loader:

            if torch.cuda.is_available():
                model.cuda()
            images = images.cuda()
            labels = labels.cuda()
            # Training pass
            optimizer.zero_grad()
            output = model(images)
            loss = criterion(output, labels)

            #This is where the model learns by backpropagating
            loss.backward()

            #And optimizes its weights here
            optimizer.step()

            running_loss += loss.item()
    return model


def plotConfusion(model):
  if torch.cuda.is_available():
        model.cuda()
  @torch.no_grad()
  def get_all_preds(model, loader):
      all_preds = torch.tensor([])
      if torch.cuda.is_available():
        all_preds = all_preds.cuda()
      for batch in loader:
          images, labels = batch
          if torch.cuda.is_available():
            model.cuda()
            images = images.cuda()
            labels = labels.cuda()
          preds = model(images).detach()
          all_preds = torch.cat(
              (all_preds, preds)
              ,dim=0
          )
      return all_preds
```

3.6.3 Federated Learning

The implementation of Federated Learning(FL) was based on the OpenMined PySyft library, which is a Python library for secure and private Deep Learning. PySyft enabled us to develop a FL system based on a PyTorch Deep Learning model and it facilitated the inclusion of Encrypted computation & exchange of parameters inside the system, which in turn contributed to making the entire system Privacy-Preserving.

The implementation consists of two separate entities:

- Central Server

  The entire process of Federated Learning is governed by the Central Server. The Central Server contains the Global ML model and the training plans required to set up a new client in the system. It also has plans to average the updates received from client devices to improve the model performance. It also handles setting up the keys required for encrypted communication between the client and the server.

- Worker Clients

  These are the devices that participate in the process. These client devices initially request the Central Server to join the FL workflow. If accepted the clients also receive the Model Architecture with global parameters and the preset training plans. The clients then train the model on their locally available data and pass the updated model parameters to the Server whenever the model improves over the global model.

Implementation of Central Server

We implemented the Central Server using the OpenMined PyGrid environment. PyGrid is a peer-to-peer network of data owners and data scientists who can collectively train AI models using PySyft. The environment was created inside a Docker Container. The Docker container we created contains all the dependencies to start an instance of Central Server, known as PyGrid-Domain. The entire process of FL starts with the administrator or the designer developing a ML model along with a suitable training and averaging plan and hosting it on PyGrid-Domain. The domain works as an interface for the clients to perform actions. The environment has an SQLite database that contains the list of clients and their corresponding configurations as well as the ML model's architecture. The PyGrid-Domain that we implemented can be used to host multiple models and training plans depending on the use case.

Training Plan

```
def set_params(model, params):
    for p, p_new in zip(model.parameters(), params):
        p.data = p_new.data


def cross_entropy_loss(logits, targets, batch_size):
    norm_logits = logits - logits.max()
    log_probs = norm_logits -
    norm_logits.exp().sum(dim=1, keepdim=True).log()
    return -(targets * log_probs).sum() / batch_size


def sgd_step(model, lr=0.1):
    with ROOT_CLIENT.torch.no_grad():
        for p in model.parameters():
            p.data = p.data - lr * p.grad
            p.grad = th.zeros_like(p.grad.get())


@make_plan
```

```
def train(
    xs=th.rand([64 * 3, 1, 28, 28]),
    ys=th.randint(0, 10, [64 * 3, 10]),
    params=List(local_model.parameters()),
):


    model = local_model.send(ROOT_CLIENT)
    set_params(model, params)
    for i in range(1):
        indices = th.tensor(range(64 * i, 64 * (i + 1)))
        x, y = xs.index_select(0, indices),
        ys.index_select(0, indices)
        out = model(x)
        loss = cross_entropy_loss(out, y, 64)
        loss.backward()
        sgd_step(model)

    return model.parameters()
```

Averaging Plan

```
@make_plan
def avg_plan(
    avg=List(local_model.parameters()),
    item=List(local_model.parameters()),
    num=Int(0)
):
    new_avg = []
    for i, param in enumerate(avg):
        new_avg.append((avg[i] * num + item[i]) / (num + 1))
    return new_avg
```

Central Server Configuration

```
name = "mnist"
version = "1.0.0"

client_config = {
    "name": name,
    "version": version,
    "batch_size": 64,
    "lr": 0.1,
    "max_updates": 1,
}

server_config = {
    "min_workers": 1,
#     "min_workers": 2,
    "max_workers": 2,
    "pool_selection": "random",
    "do_not_reuse_workers_until_cycle": 6,
    "cycle_length": 28800,  # max cycle length in seconds
    "num_cycles": 30,  # max number of cycles
    "max_diffs": 1,  # number of diffs to collect before avg
    "minimum_upload_speed": 0,
    "minimum_download_speed": 0,
    "iterative_plan": True,
}
```

Implementation of Worker Client

The clients are the entities that represent themselves as data containers, i.e they have data locally that they use to train the model and update the model parameters. The implementation of a client was done using a Python environment. The client requests the server to participate in the FL workflow and the server accepts the client along with it, the server also provides configuration and training details. The client entity as well as the entire process was defined using PySyft functions.

Client creation & Cycle

```
def create_client_and_run_cycle():
    client = FLClient(url=gridAddress,
    auth_token=auth_token, secure=False)
    client.worker_id = client.grid_worker.authenticate(
        client.auth_token, model_name, model_version
    )["data"]["worker_id"]
    job = client.new_job(model_name, model_version)
```

```
# Set event handlers
job.add_listener(job.EVENT_ACCEPTED, on_accepted)
job.add_listener(job.EVENT_REJECTED, on_rejected)
job.add_listener(job.EVENT_ERROR, on_error)

# Shoot!
job.start()
```

3.6.4 User Interface

Web Application for Doctors

For developing the web application, We used Flask. Flask is a micro web framework written in Python. The main reason for choosing this framework is because it is written in Python language. Other modules in our system like Pre-processing, Federated Learning and Deep Learning model are also developed using the Python language. So by using Flask, all these modules can be integrated and run under the same environment. We have three different pages on our web application. The first page is for the login, the second one for entering the patient's data and prediction of the heart condition and the third one is for entering the prescription details and remarks for the patient.

SQLite database is used for storing the data of both doctors and patients. There are three tables in our database. The first one is for storing the details of the doctor like name, username and password. The second table contains the details of the patients like Prescription ID, name, gender, heart condition and remarks. The third table contains the medication details of each Prescription ID.

Mobile Application for Patients

Flutter SDK is used to develop the mobile application. Flutter SDK is Google's UI toolkit for crafting natively compiled applications for mobile, web, and desktop from a single codebase. The main reason for choosing flutter is because It can be used to develop cross-platform applications from a single codebase.

The mobile application has two windows. The first window is for the login or authentication of the user and the second one is for displaying the medical details of the user. We have designed and developed an API as a part of our web application for connecting the mobile application with our system.

CHAPTER 4

RESULTS & DISCUSSION

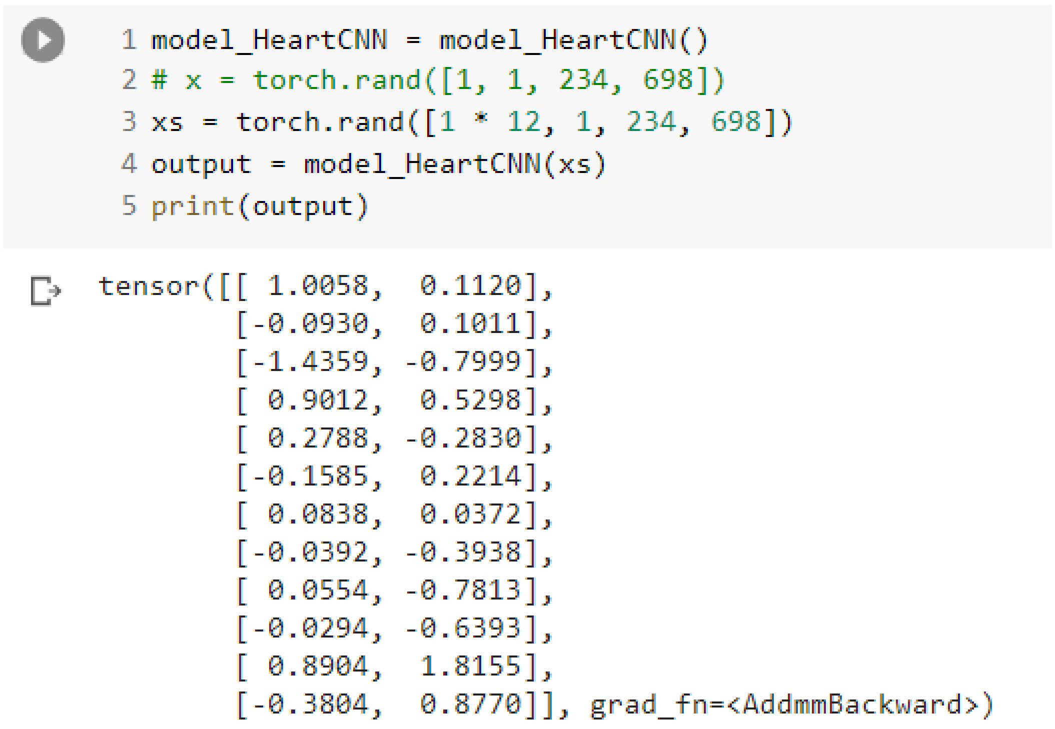
4.1 Federated Learning Heart Sound Classification Results

Federated Learning allows to execute privacy-preserving collaborative learning on Heart sound data of patients, usually these data are protected under privacy guidelines & therefore cannot be shared.

Using Federated Learning a globally improvised model can be formed without sharing data between non-affiliated hospitals yet utilising the data by learning inferences from them. Which increases the prediction accuracy.

4.1.1 Classification in Client Device

Deep Learning Model in client device will output probability of each classes i.e, Abnormal & Normal

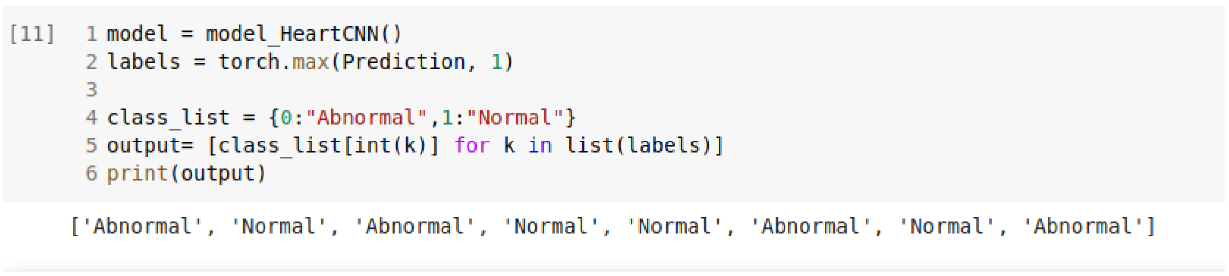


```
model_HeartCNN = model_HeartCNN()
# x = torch.rand([1, 1, 234, 698])
xs = torch.rand([1 * 12, 1, 234, 698])
output = model_HeartCNN(xs)
print(output)
```

```
tensor([[ 1.0058,  0.1120],
        [-0.0930,  0.1011],
        [-1.4359, -0.7999],
        [ 0.9012,  0.5298],
        [ 0.2788, -0.2830],
        [-0.1585,  0.2214],
        [ 0.0838,  0.0372],
        [-0.0392, -0.3938],
        [ 0.0554, -0.7813],
        [-0.0294, -0.6393],
        [ 0.8904,  1.8155],
        [-0.3804,  0.8770]], grad_fn=<AddmmBackward>)
```

Figure 4.1: Training Data With Batch Size 12

The corresponding prediction classes are mapped here to make it easier for us to understand

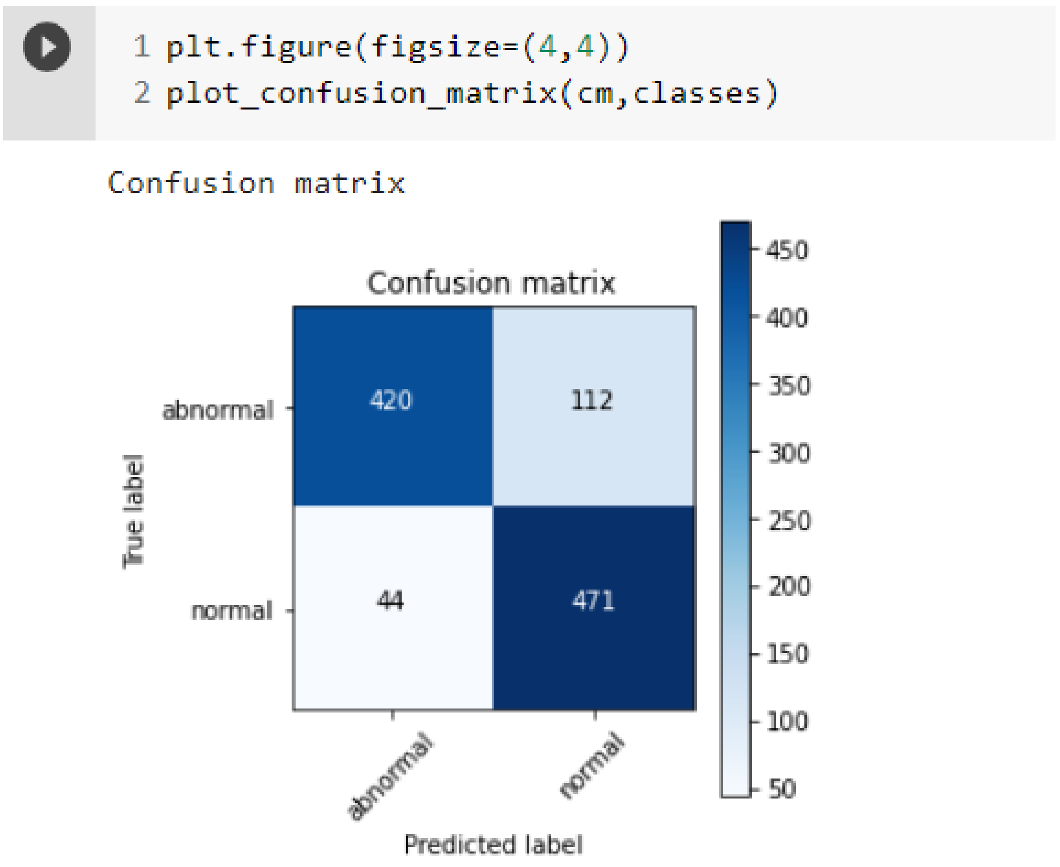


```
[11] model = model_HeartCNN()
     labels = torch.max(Prediction, 1)

     class_list = {0:"Abnormal",1:"Normal"}
     output= [class_list[int(k)] for k in list(labels)]
     print(output)

   ['Abnormal', 'Normal', 'Abnormal', 'Normal', 'Normal', 'Abnormal', 'Normal', 'Abnormal']
```

Figure 4.2: Predictions mapped to respective classes

Confusion Matrix of Classifier

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. Confusion Matrix of the classifier in the client device is shown below



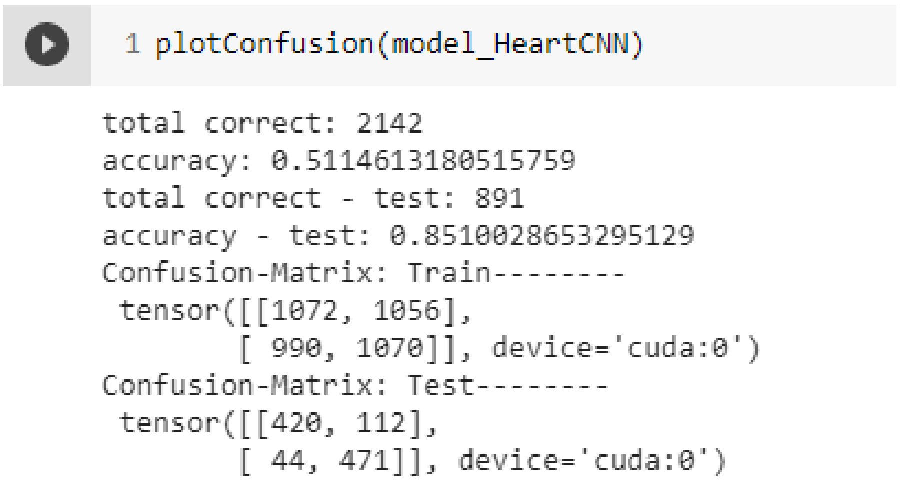
```
plt.figure(figsize=(4,4))
plot_confusion_matrix(cm,classes)
```

```
Confusion matrix
```

Figure 4.3: Confusion Matrix

Analysis of Confusion Matrix

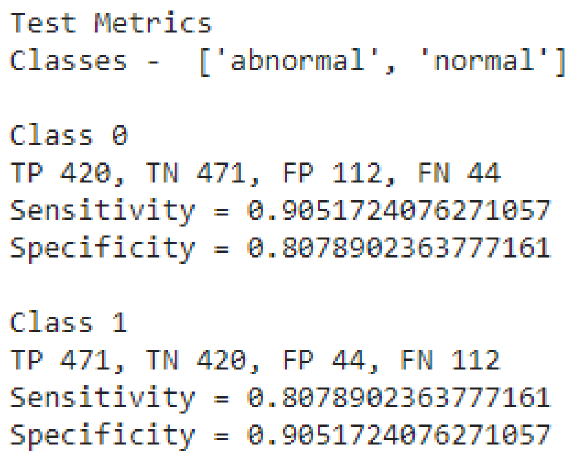
The classifier produced an accuracy 85% on average. The sensitivity & Specificity of the respective classes are also listed. Sensitivity is the ability of a test to correctly identify patients with a disease. Specificity is the ability of a test to correctly identify people without the disease



```
plotConfusion(model_HeartCNN)
```

```
total correct: 2142
accuracy: 0.5114613180515759
total correct - test: 891
accuracy - test: 0.8510028653295129
Confusion-Matrix: Train--------
 tensor([[1072, 1056],
        [ 990, 1070]], device='cuda:0')
Confusion-Matrix: Test--------
 tensor([[420, 112],
        [ 44, 471]], device='cuda:0')
```

Figure 4.4: Sensitivity & Specificity



```
Test Metrics
Classes -  ['abnormal', 'normal']

Class 0
TP 420, TN 471, FP 112, FN 44
Sensitivity = 0.9051724076271057
Specificity = 0.8078902363777161

Class 1
TP 471, TN 420, FP 44, FN 112
Sensitivity = 0.8078902363777161
Specificity = 0.905172407627105 7
```

Figure 4.5: Sensitivity & Specificity

CHAPTER 5

CONCLUSION & FUTURE SCOPE

5.1 Conclusion

Machine Learning, and particularly Deep Learning, has led to a wide range of innovations in the area of digital healthcare. As all ML methods benefit greatly from the ability to access data that approximates the true global distribution, Federated Learning is a promising approach to obtain powerful, accurate, safe, robust and unbiased models. By enabling multiple parties to train collaboratively without the need to exchange or centralise data sets, FL neatly addresses issues related to egress of sensitive medical data.

We use Federated learning to diagnose abnormalities in the Cardiovascular system overcoming the limitations of the existing machine learning models. Federated learning is used to provide an efficient, secure and privacy preserving model. We have designed the system, which aims to identify features that can best diagnose these abnormalities.

5.2 Future Scope

In future, we plan to create highly scalable system that can accommodate more clients hence increasing the efficiency, accuracy and usability of the system.

We also aim to create an improved Machine Learning model with performance improvements. The security of the patient's medical records also need to be addressed, so we intend to develop a Block chain based Medical Record system that is highly secure.

We are also working on deploying the entire system by collaborating with hospitals, to explore the true potential of Federated Learning.

Bibliography

[1] Xu, Jie and Glicksberg, Benjamin S and Su, Chang and Walker, Peter and Bian, Jiang and Wang, Fei. *”Federated learning for healthcare informatics”* Journal of Healthcare Informatics Research, pages 1–19. Springer, 2020.

[2] Yang, Qiang and Liu, Yang and Chen, Tianjian and Tong, Yongxin. *”Federated machine learning: Concept and applications”*. ACM Transactions on Intelligent Systems and Technology (TIST, 2019(10):60– 64, 2017.

[3] Kelvin *”Introduction to Federated Learning and Challenges”*. https://towardsdatascience.com/introduction-to-federated-learning-and-challenges-ea7e02 f260ca (Medium), November, 2020.

[4] Mushtaq, Zohaib and Su, Shun-Feng and Tran, Quoc-Viet. *”Spectral images based environmental sound classification using CNN with meaningful data augmentation”*. . In 2016 17th Applied Acoustics, pages 107581, vol 172. Elsevier, 2020

[5] Li, Fan and Tang, Hong and Shang, Shang and Mathiak, Klaus and Cong, Fengyu *”Classification of Heart Sounds Using Convolutional Neural Network”*. In Applied Sciences,2020 Multidisciplinary Digital Publishing Institute volume 10,11,3956

[6] Ren, Zhao and Cummins, Nicholas and Pandit, Vedhas and Han, Jing and Qian, Kun and Schuller, Björn *”Learning image-based representations for heart sound classification”*. .In 2018 Proceedings of the 2018 International Conference on Digital Health 143–147.

[7] Rieke, Nicola and Hancox, Jonny and Li, Wenqi and Milletari, Fausto and Roth, Holger and Albarqouni, Shadi and Bakas, Spyridon and Galtier, Mathieu N and Landman, Bennett and Maier-Hein, Klaus and others *”arXiv preprint arXiv:2003.08119”*. 2020

[8] Padilla-Ortiz, Ana L and Ibarra, David *”Lung and heart sounds analysis: state-of-the-art and future trends”*. Critical Reviews™ in Biomedical Engineering, vol 46, no 1. Begel House Inc. 2018

[9] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, Lin Xiao Kelvin *”Optimal Distributed Online Prediction Using Mini-Batches”*. https://www.jmlr.org/papers/volume13/dekel12a/dekel12a.pdf (Medium), Journal of Machine Learning Research 13 (2012) 165-202

[10] Aspden P, Corrigan JM, Wolcott J, et al *”Patient Safety: Achieving a New Standard for Care.”*. https://www.ncbi.nlm.nih.gov/books/NBK216088/ (NCBI), Institute of Medicine (US) Committee on Data Standards for Patient Safety, 2004.

[11] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Aguera y Arcas *"Communication-Efficient Learning of Deep Networks from Decentralized Data"*. Proceedings of Machine Learning Research, Volume 54

[12] Ketan Doshi *"Audio Deep Learning Made Simple (Part 1): State-of-the-Art Techniques"*. https://towardsdatascience.com/audio-deep-learning-made-simple-part-1-state-of-the-art-techniques-da1d3dff2504 (Towardsdatascience.com), Feb, 2021.

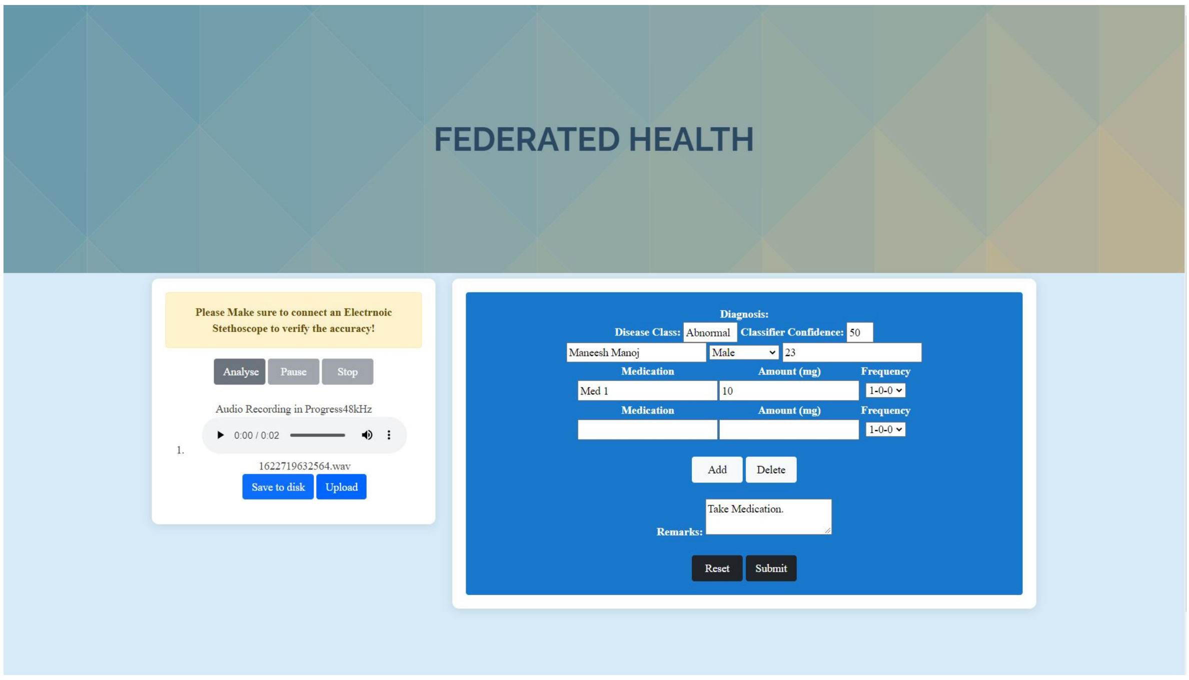
APPENDIX A
SUMMARY OF THE RESULTS

	Sensitivity	Specificity
Abnormal	90.51%	80.781%
Precision	80.781	90.51

Table A.1: Summarization Table

APPENDIX B
SCREENSHOTS

User Interface



Please Make sure to connect an Electrnoic Stethoscope to verify the accuracy!

Audio Recording in Progress48kHz

1. 0:00 / 0:02

1622719632564.wav

Diagnosis:

Disease Class: Abnormal Classifier Confidence: 50

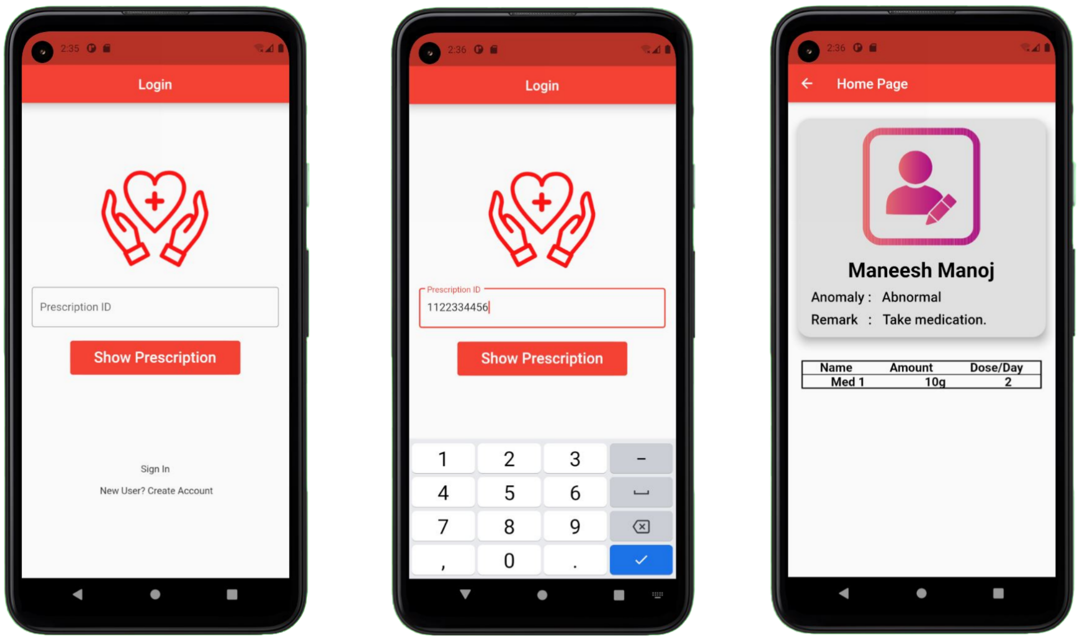
Maneesh Manoj Male 23

Medication	Amount (mg)	Frequency
Med 1	10	1-0-0

Medication	Amount (mg)	Frequency
		1-0-0

Remarks: Take Medication.

Figure B.1: Web Application User Interface



Prescription ID 1122334456

Maneesh Manoj

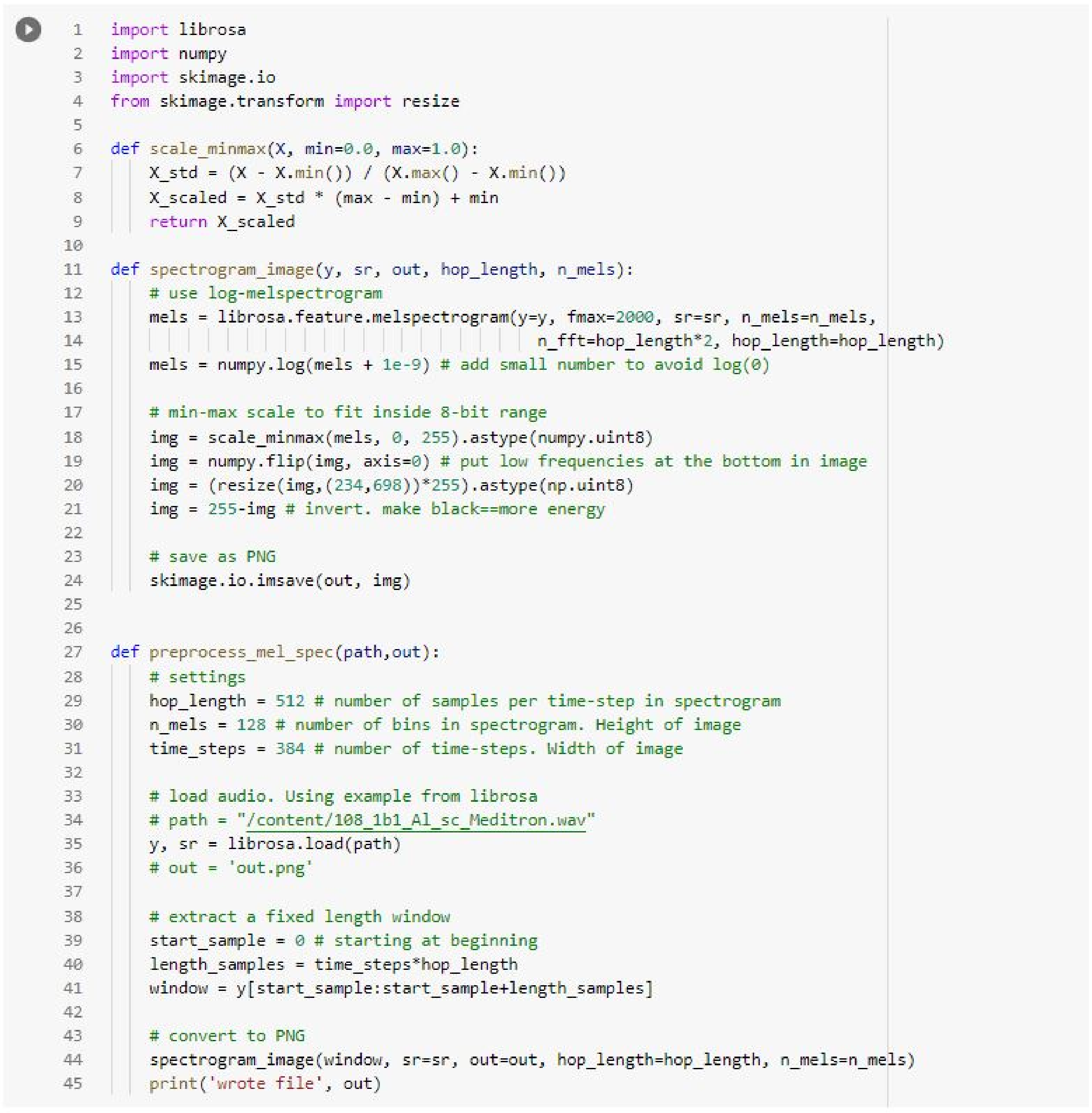
Anomaly : Abnormal

Remark : Take medication.

Name	Amount	Dose/Day
Med 1	10g	2

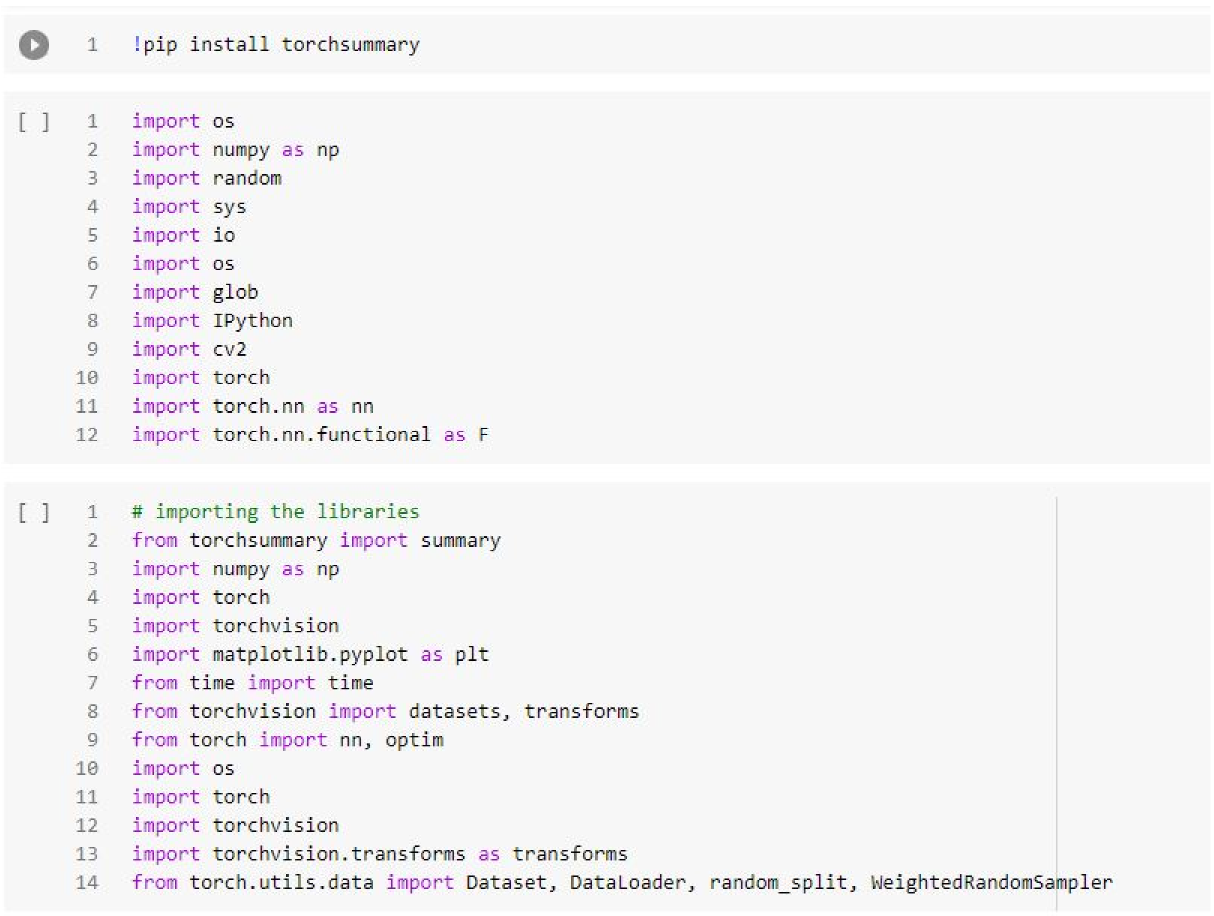
Figure B.2: Mobile Application User Interface

Google Colab Screenshots



```python
import librosa
import numpy
import skimage.io
from skimage.transform import resize

def scale_minmax(X, min=0.0, max=1.0):
    X_std = (X - X.min()) / (X.max() - X.min())
    X_scaled = X_std * (max - min) + min
    return X_scaled

def spectrogram_image(y, sr, out, hop_length, n_mels):
    # use log-melspectrogram
    mels = librosa.feature.melspectrogram(y=y, fmax=2000, sr=sr, n_mels=n_mels,
                                          n_fft=hop_length*2, hop_length=hop_length)
    mels = numpy.log(mels + 1e-9) # add small number to avoid log(0)

    # min-max scale to fit inside 8-bit range
    img = scale_minmax(mels, 0, 255).astype(numpy.uint8)
    img = numpy.flip(img, axis=0) # put low frequencies at the bottom in image
    img = (resize(img,(234,698))*255).astype(np.uint8)
    img = 255-img # invert. make black==more energy

    # save as PNG
    skimage.io.imsave(out, img)


def preprocess_mel_spec(path,out):
    # settings
    hop_length = 512 # number of samples per time-step in spectrogram
    n_mels = 128 # number of bins in spectrogram. Height of image
    time_steps = 384 # number of time-steps. Width of image

    # load audio. Using example from librosa
    # path = "/content/108_1b1_Al_sc_Meditron.wav"
    y, sr = librosa.load(path)
    # out = 'out.png'

    # extract a fixed length window
    start_sample = 0 # starting at beginning
    length_samples = time_steps*hop_length
    window = y[start_sample:start_sample+length_samples]

    # convert to PNG
    spectrogram_image(window, sr=sr, out=out, hop_length=hop_length, n_mels=n_mels)
    print('wrote file', out)
```

Figure B.3: Colab - Spectrogram Creation

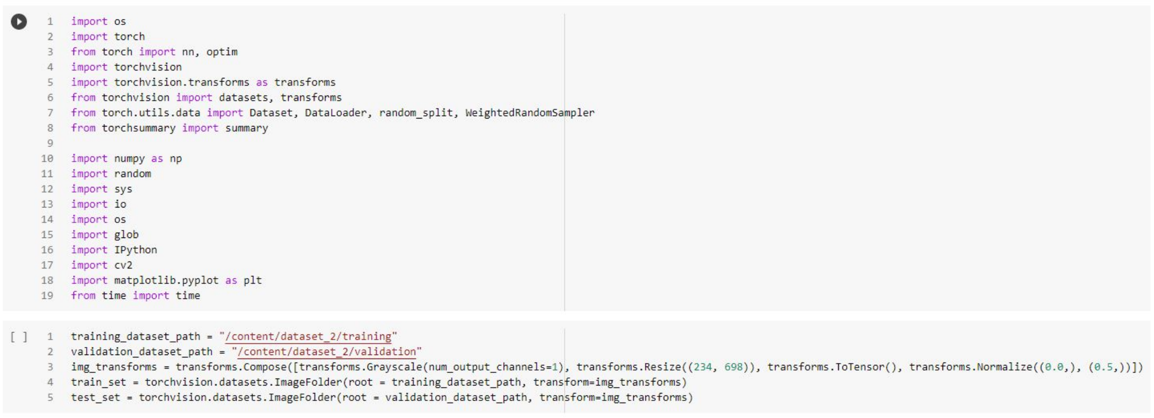


```
!pip install torchsummary
```

```
import os
import numpy as np
import random
import sys
import io
import os
import glob
import IPython
import cv2
import torch
import torch.nn as nn
import torch.nn.functional as F
```

```
# importing the libraries
from torchsummary import summary
import numpy as np
import torch
import torchvision
import matplotlib.pyplot as plt
from time import time
from torchvision import datasets, transforms
from torch import nn, optim
import os
import torch
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import Dataset, DataLoader, random_split, WeightedRandomSampler
```

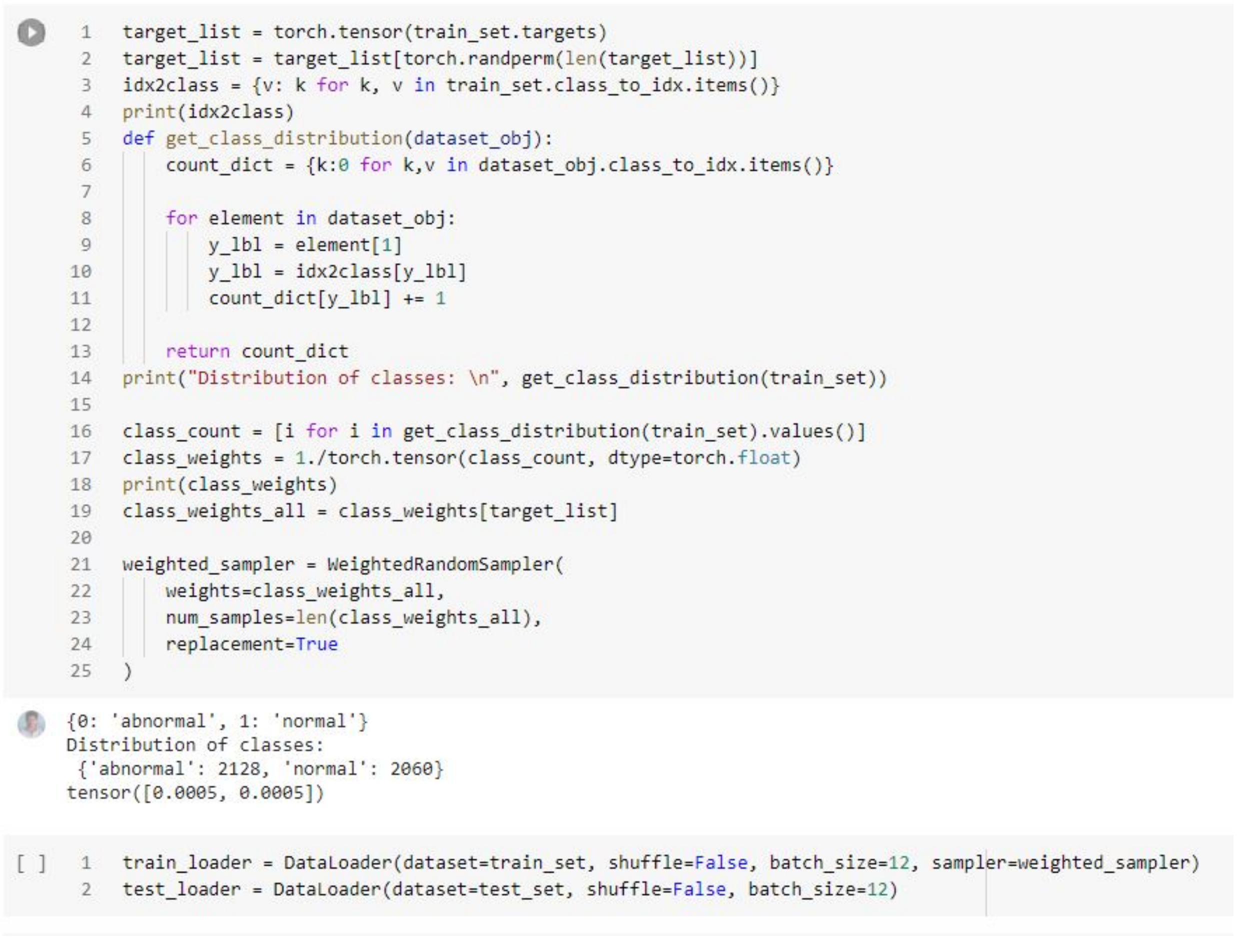
Figure B.4: Colab - Library Imports



```
import os
import torch
from torch import nn, optim
import torchvision
import torchvision.transforms as transforms
from torchvision import datasets, transforms
from torch.utils.data import Dataset, DataLoader, random_split, WeightedRandomSampler
from torchsummary import summary

import numpy as np
import random
import sys
import io
import os
import glob
import IPython
import cv2
import matplotlib.pyplot as plt
from time import time
```

```
training_dataset_path = "/content/dataset_2/training"
validation_dataset_path = "/content/dataset_2/validation"
img_transforms = transforms.Compose([transforms.Grayscale(num_output_channels=1), transforms.Resize((234, 698)), transforms.ToTensor(), transforms.Normalize((0.0,), (0.5,))])
train_set = torchvision.datasets.ImageFolder(root = training_dataset_path, transform=img_transforms)
test_set = torchvision.datasets.ImageFolder(root = validation_dataset_path, transform=img_transforms)
```

Figure B.5: Colab - DataLoader

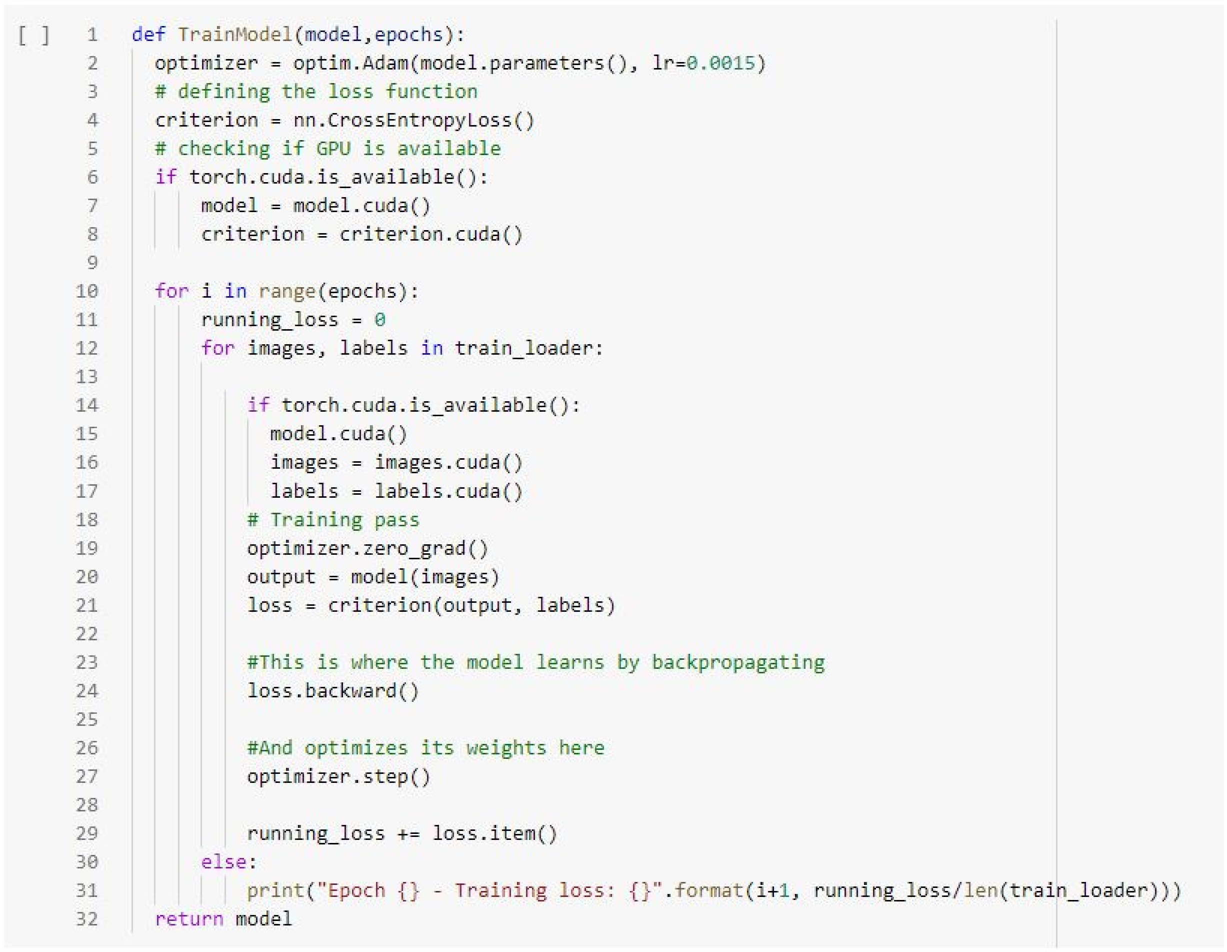


```
target_list = torch.tensor(train_set.targets)
target_list = target_list[torch.randperm(len(target_list))]
idx2class = {v: k for k, v in train_set.class_to_idx.items()}
print(idx2class)
def get_class_distribution(dataset_obj):
    count_dict = {k:0 for k,v in dataset_obj.class_to_idx.items()}

    for element in dataset_obj:
        y_lbl = element[1]
        y_lbl = idx2class[y_lbl]
        count_dict[y_lbl] += 1

    return count_dict
print("Distribution of classes: \n", get_class_distribution(train_set))

class_count = [i for i in get_class_distribution(train_set).values()]
class_weights = 1./torch.tensor(class_count, dtype=torch.float)
print(class_weights)
class_weights_all = class_weights[target_list]

weighted_sampler = WeightedRandomSampler(
    weights=class_weights_all,
    num_samples=len(class_weights_all),
    replacement=True
)
```

```
{0: 'abnormal', 1: 'normal'}
Distribution of classes:
 {'abnormal': 2128, 'normal': 2060}
tensor([0.0005, 0.0005])
```

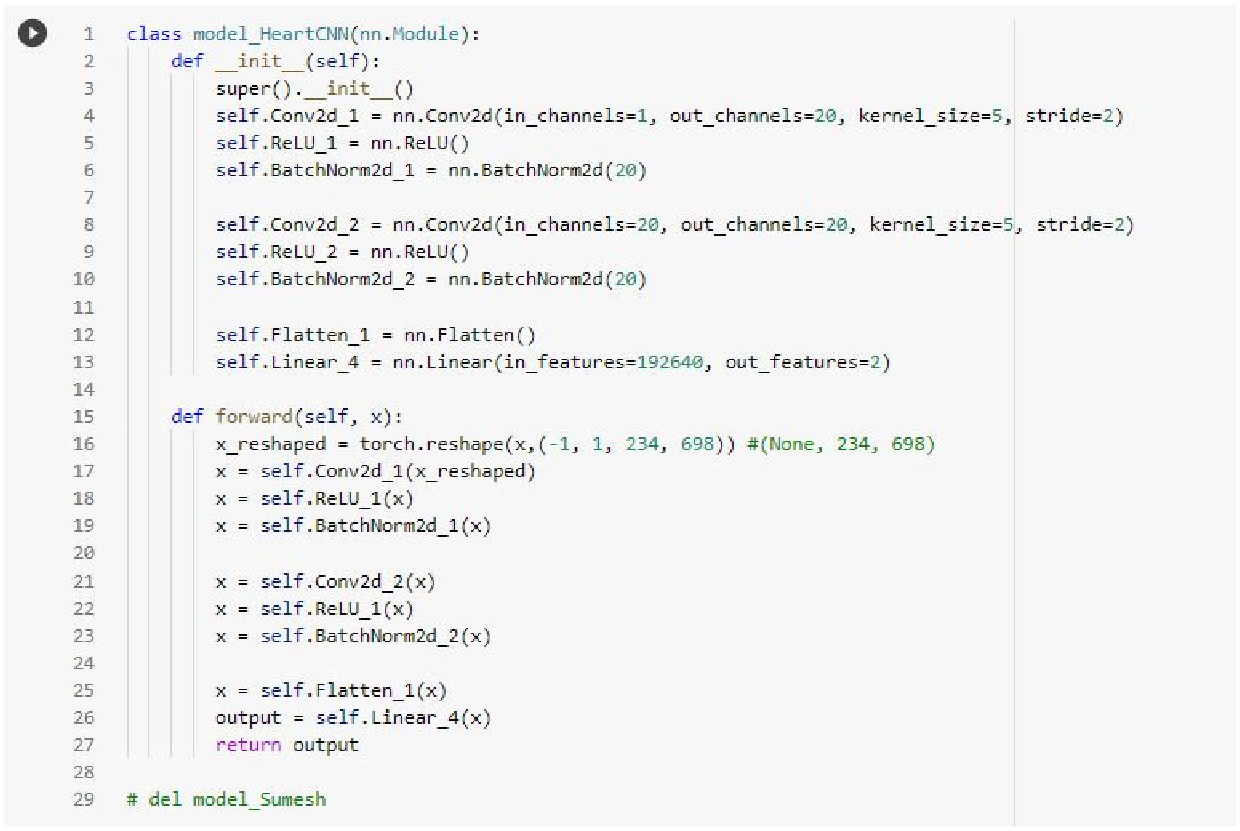
```
train_loader = DataLoader(dataset=train_set, shuffle=False, batch_size=12, sampler=weighted_sampler)
test_loader = DataLoader(dataset=test_set, shuffle=False, batch_size=12)
```

Figure B.6: Colab - Weighted Sampler



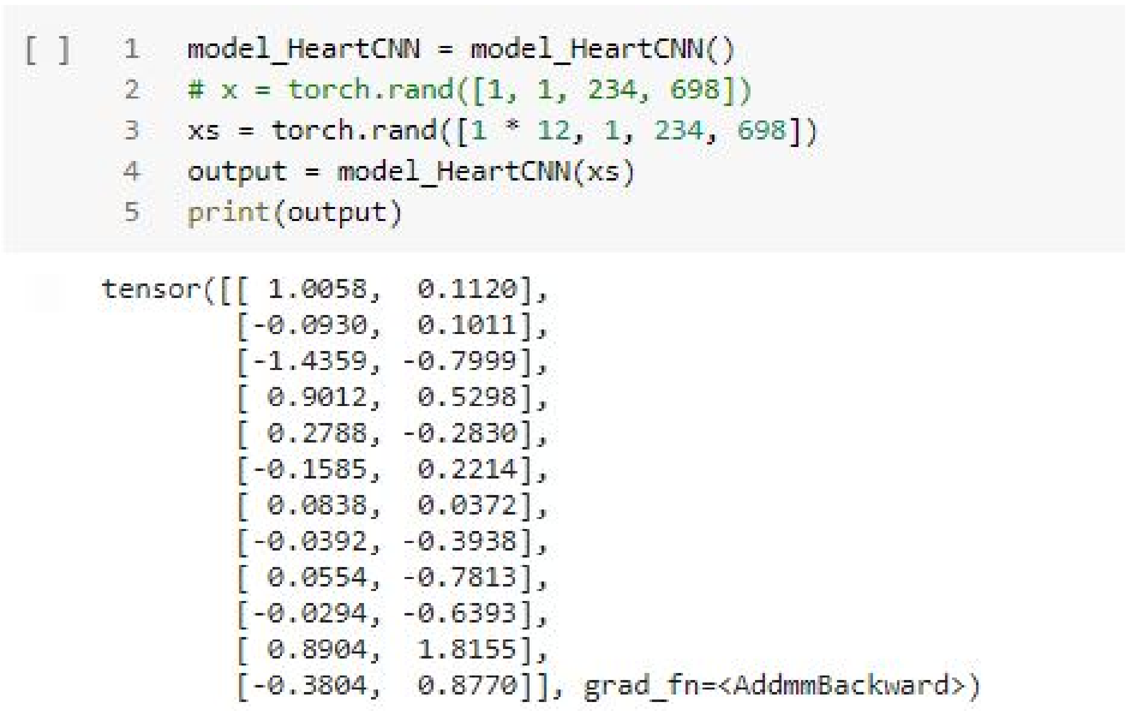
```
[ ]
def TrainModel(model,epochs):
  optimizer = optim.Adam(model.parameters(), lr=0.0015)
  # defining the loss function
  criterion = nn.CrossEntropyLoss()
  # checking if GPU is available
  if torch.cuda.is_available():
      model = model.cuda()
      criterion = criterion.cuda()

  for i in range(epochs):
      running_loss = 0
      for images, labels in train_loader:

          if torch.cuda.is_available():
            model.cuda()
            images = images.cuda()
            labels = labels.cuda()
          # Training pass
          optimizer.zero_grad()
          output = model(images)
          loss = criterion(output, labels)

          #This is where the model learns by backpropagating
          loss.backward()

          #And optimizes its weights here
          optimizer.step()

          running_loss += loss.item()
      else:
          print("Epoch {} - Training loss: {}".format(i+1, running_loss/len(train_loader)))
  return model
```

Figure B.7: Colab - Train Function



```
class model_HeartCNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.Conv2d_1 = nn.Conv2d(in_channels=1, out_channels=20, kernel_size=5, stride=2)
        self.ReLU_1 = nn.ReLU()
        self.BatchNorm2d_1 = nn.BatchNorm2d(20)

        self.Conv2d_2 = nn.Conv2d(in_channels=20, out_channels=20, kernel_size=5, stride=2)
        self.ReLU_2 = nn.ReLU()
        self.BatchNorm2d_2 = nn.BatchNorm2d(20)

        self.Flatten_1 = nn.Flatten()
        self.Linear_4 = nn.Linear(in_features=192640, out_features=2)

    def forward(self, x):
        x_reshaped = torch.reshape(x,(-1, 1, 234, 698)) #(None, 234, 698)
        x = self.Conv2d_1(x_reshaped)
        x = self.ReLU_1(x)
        x = self.BatchNorm2d_1(x)

        x = self.Conv2d_2(x)
        x = self.ReLU_1(x)
        x = self.BatchNorm2d_2(x)

        x = self.Flatten_1(x)
        output = self.Linear_4(x)
        return output

# del model_Sumesh
```

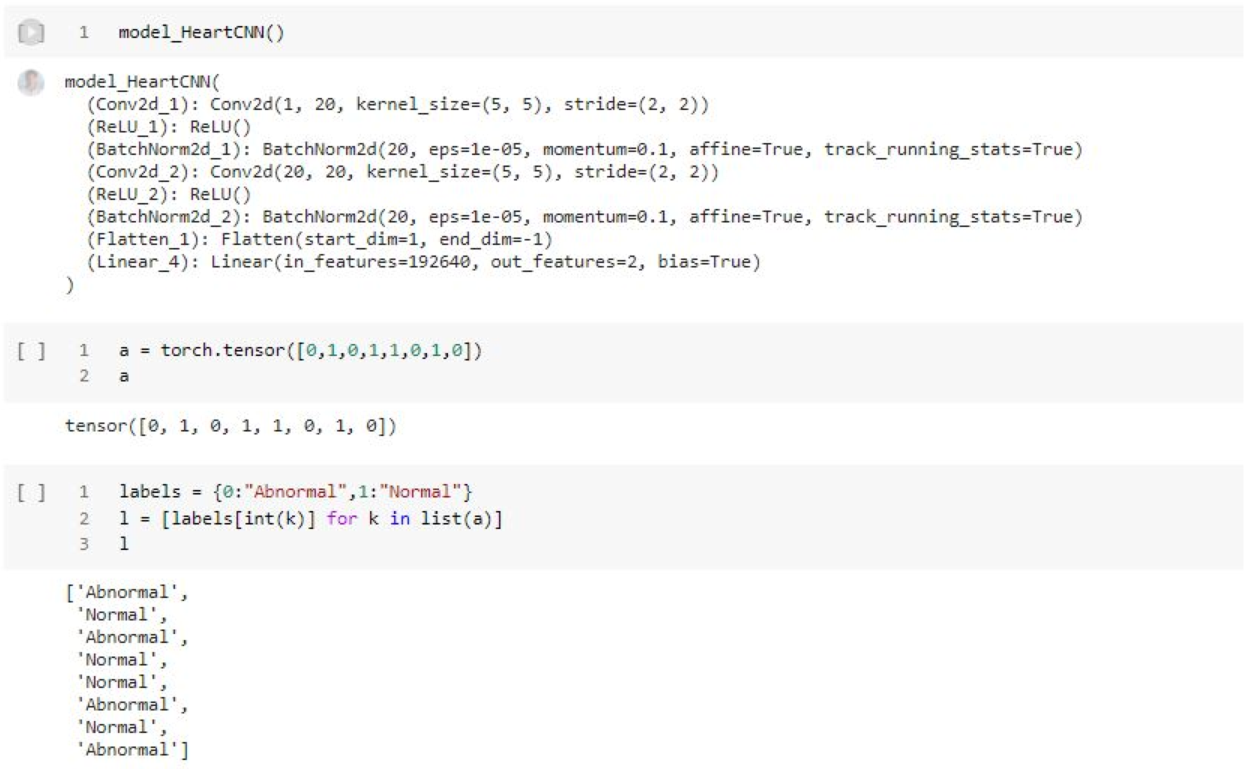
Figure B.8: Colab - Model Architecture



```
model_HeartCNN = model_HeartCNN()
# x = torch.rand([1, 1, 234, 698])
xs = torch.rand([1 * 12, 1, 234, 698])
output = model_HeartCNN(xs)
print(output)
```

```
tensor([[ 1.0058,  0.1120],
        [-0.0930,  0.1011],
        [-1.4359, -0.7999],
        [ 0.9012,  0.5298],
        [ 0.2788, -0.2830],
        [-0.1585,  0.2214],
        [ 0.0838,  0.0372],
        [-0.0392, -0.3938],
        [ 0.0554, -0.7813],
        [-0.0294, -0.6393],
        [ 0.8904,  1.8155],
        [-0.3804,  0.8770]], grad_fn=<AddmmBackward>)
```

Figure B.9: Colab - Sample Batch Prediction Result



```
model_HeartCNN()
```

```
model_HeartCNN(
  (Conv2d_1): Conv2d(1, 20, kernel_size=(5, 5), stride=(2, 2))
  (ReLU_1): ReLU()
  (BatchNorm2d_1): BatchNorm2d(20, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (Conv2d_2): Conv2d(20, 20, kernel_size=(5, 5), stride=(2, 2))
  (ReLU_2): ReLU()
  (BatchNorm2d_2): BatchNorm2d(20, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (Flatten_1): Flatten(start_dim=1, end_dim=-1)
  (Linear_4): Linear(in_features=192640, out_features=2, bias=True)
)
```

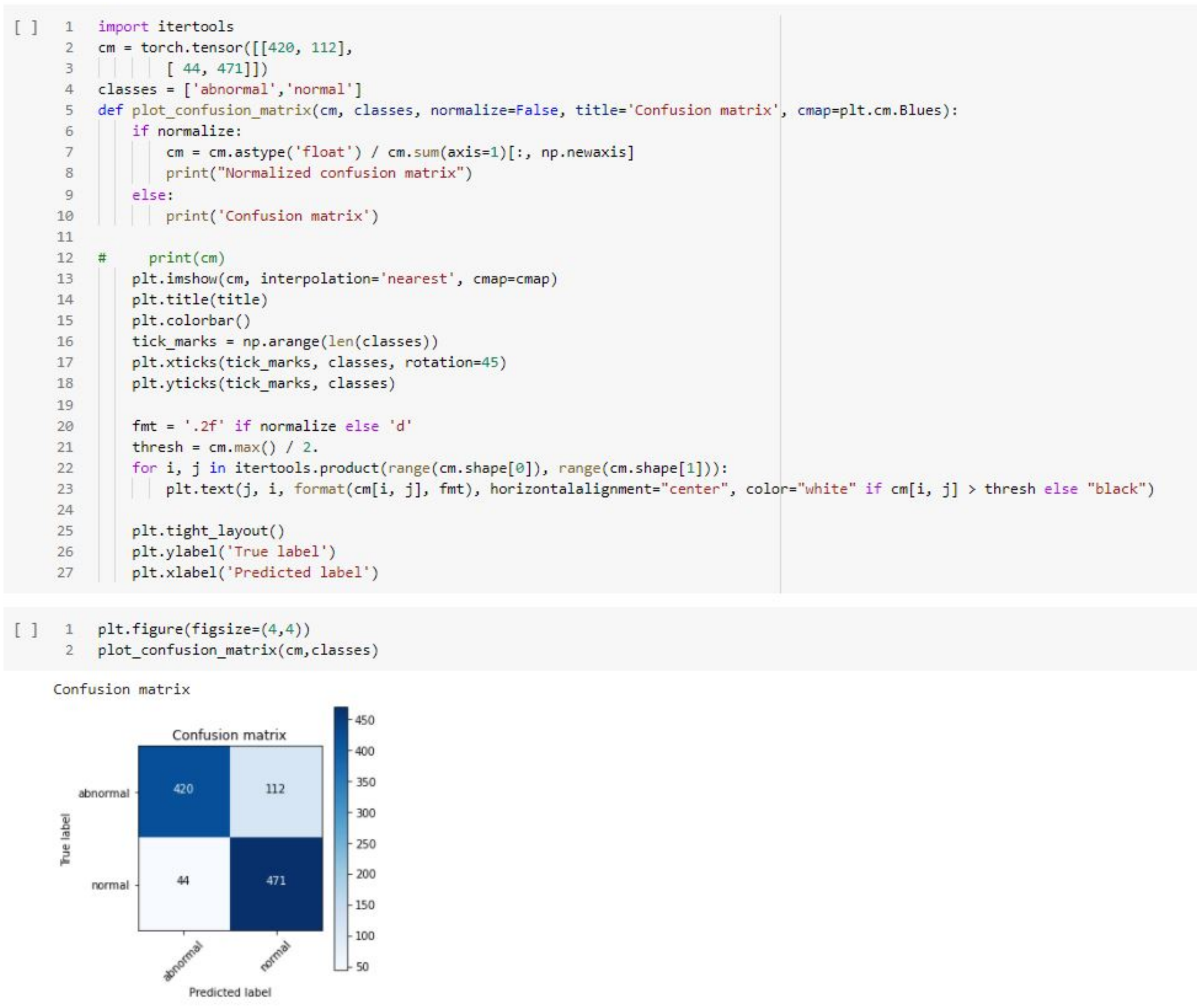
```
[ ]
a = torch.tensor([0,1,0,1,1,0,1,0])
a
```

```
tensor([0, 1, 0, 1, 1, 0, 1, 0])
```

```
[ ]
labels = {0:"Abnormal",1:"Normal"}
l = [labels[int(k)] for k in list(a)]
l
```

```
['Abnormal',
 'Normal',
 'Abnormal',
 'Normal',
 'Normal',
 'Abnormal',
 'Normal',
 'Abnormal']
```

Figure B.10: Colab - Prediction Mapped to Labels



```
[ ]  import itertools
     cm = torch.tensor([[420, 112],
                [ 44, 471]])
     classes = ['abnormal','normal']
     def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix', cmap=plt.cm.Blues):
         if normalize:
             cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
             print("Normalized confusion matrix")
         else:
             print('Confusion matrix')

     #     print(cm)
         plt.imshow(cm, interpolation='nearest', cmap=cmap)
         plt.title(title)
         plt.colorbar()
         tick_marks = np.arange(len(classes))
         plt.xticks(tick_marks, classes, rotation=45)
         plt.yticks(tick_marks, classes)

         fmt = '.2f' if normalize else 'd'
         thresh = cm.max() / 2.
         for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
             plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center", color="white" if cm[i, j] > thresh else "black")

         plt.tight_layout()
         plt.ylabel('True label')
         plt.xlabel('Predicted label')
```

```
[ ]  plt.figure(figsize=(4,4))
     plot_confusion_matrix(cm,classes)
```

```
Confusion matrix
```

Figure B.11: Colab - Confusion Matrix Code