A PROJECT REPORT ON

Early Screening for Cardiovascular Disorders

REPORT SUBMITTED TOWARDS PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN
Electronics & Telecommunication

SUBMITTED BY

GROUP NO: G11

Name of the students	PRN
ADITYA ARUN	16070123007
GAURAV BHOSALE	16070123031
ANISH NARGUND	16070123061
MUSKAAN PARMAR	16070123064

Under the Guidance of

Dr. Mrinal Bachute

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE – 412115

(A CONSTITUENT OF SYMBIOSIS INTERNATIONAL (DEEMED UNIVERSITY))

2016-2020

CERTIFICATE

The project titled **Early Screening for Cardiovascular Disorders** submitted to the Symbiosis Institute of Technology, Pune for the final year project in Electronics & Telecommunication Engineering is based on our original work carried out under the guidance of **Dr. Mrinal Bachute**. The report has not been submitted elsewhere for the award of any degree.

The material borrowed from other source and incorporated in the report has been duly acknowledged and/or referenced.

We understand that we could be held responsible and accountable for plagiarism, if any, detected later on.

Sr. No.	Name of the students	PRN	Sign
1	ADITYA ARUN	16070123007	
2	GAURAV BHOSALE	16070123031	
3	ANISH NARGUND	16070123061	
4	MUSKAAN PARMAR	16070123064	

Date:

Dr. Mrinal Bachute
Assistant Professor, SIT, Pune

Dr. Neela R
Head of Department, SIT, Pune

ABSTRACT

This Project titled ‘Early Screening for Cardiovascular Disorders’ aims to provide a low-cost solution to mitigate the severe lack of medical and healthcare workers in the rural parts of the country. The WHO states that heart-related diseases are the number one cause of death globally, with three-quarters occurring in developing countries. In India, the doctor-patient ratio is less than the WHO-prescribed limit of 1:1000 with rural areas bearing the maximum brunt. The solution proposed by us aims to assist any worker or volunteer with no prior experience or skill in the medical industry to detect murmurs or arrhythmia in the heart sounds of the test subject, which could be symptoms for heart complications typically detected by experienced doctors. The Project aims to equip the frontline workers with a tool that can effectively and quickly conduct a primary screening on hundreds of possible patients and deliver the result in real-time. This is possible with a mobile device consisting of a microcontroller and a sensor with connectivity. The microcontroller is a Raspberry Pi unit with a sound card, specialized microphone, and chest piece of the stethoscope. The heart sound is stored into Raspberry Pi and then uploaded onto cloud storage for further processing and storage. We use Low Pass filters and Short Time Fourier Transform as preprocessing techniques to the neural network deployed on the Cloud, the data is sent onto Cloud, and Convolutional Neural Network provides the response to the user in a short interval of time. The network is trained on Pascal and PhysioNet datasets with data collected from digital stethoscopes. Since the training data used was measured with the digital stethoscope, we build our own to deliver the same data while mimicking the actual input. The classification accuracy of the model is 92%, which is achieved with the current model. However, as per the various literature, it can be further improved with an increase in the amount of training data. The system can be used for preliminary screening of CVD and can be integrated to support doctors so they can work on further diagnosis and testing.

ACKNOWLEDGEMENT

We want to thank our Project Mentor, Prof. Mrinal Bachute, for providing us with an opportunity to work under her guidance and for providing us with words of encouragement and letting us develop technical knowledge. Her valuable feedback during the project work helped us to sail through successfully. She has been supportive throughout our work and always motivated us to explore new ideas and think from different perspectives to overcome the problems that we faced.

We would also like to thank our Electronics and Telecommunication Department and Dr. Neela Rayavarapu for providing us the opportunity to work on this excellent project that proved to be a great learning experience for each one of us.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviated Word	**Expansion**
CVD	Cardiovascular Diseases
MFCC	Mel Frequency Cepstral Coefficients
CNN	Convolutional Neural Network
CDS	Clinical Decision Support
DFT	Discrete Fourier Transform
AR	Burg AR method
GCP	Google Cloud Platform
PCA	Principal Component Analysis
ANN	Artificial Neural Network
STFT	Short-Time Fourier Transform
BP	Backpropagation
HMM	Hidden Markov Model
DAE	Denoising Autoencoder
DHMM	Discrete Hidden Markov Model
LMS	Least Mean Square
ECG	Electrocardiogram
LSSVM	Least Square Support Vector Machine

CHAPTER I: INTRODUCTION

We decided on the topic "Early Screening for Cardiovascular Disorders" as our B.Tech. Project as the need for accuracy in detecting CVD is continuously increasing due to a rise in the number of cases of heart related disorders. Listening by means of stethoscope is an essential technique, being utilized by doctors for recognizing normal and abnormal heart systems. Listening to the voices, originating from the cardiovascular valves through stethoscope, doctors analyze whether there is any abnormality with respect to the heart. However, listening by means of stethoscope has various restrictions, for interpreting different heart sounds relies upon hearing capacity, experience, and particular expertise of the doctor. Such limitations may be reduced by developing biomedical based decision support system that would allow a less experienced person to distinguish whether a cardiac system is normal or not. This project, if implemented in real life can prove to be beneficial in case of rural areas especially India where the medical facilities and doctor availability is rare. It is a low-cost real time solution which can be used in preliminary screening of cardiovascular diseases. These devices can be used by NGOs in helping villagers with the detection. Thus, the early detection can be used to refer them to the doctors who can provide the necessary treatment.

This project is only to be used for preliminary detection purpose and should not considered as a replacement for doctors. This device would only help in screening. Once the preliminary testing is done, the patient may visit cardiologists to receive further treatment. So, the device informs the user about whether any CVD is present and not about the type of CVD present.

The aim of the project is to build a cost-effective stethoscope to detect the heart beats and check whether the patient is suffering from CVD or not. The stethoscope is built by modifying the analog stethoscope available with the local doctors. A mike is inserted in the tube near the stem area to amplify the audio signal received. It has a frequency range of 65Hz-18KHz. A microcontroller converts it to a spectral image by applying STFT, where parameters like window type, size and length can be varied. The obtained spectrogram will be then sent over network to GCP where Neural Network is deployed. The CNN will be used to classify the spectral images and detect any sort of abnormality present in the spectral images. We can improve the accuracy of the prognosis as we increase the data base.

CHAPTER II: LITERATURE SURVEY

In 2018, a study on CVD, which was a part of Million Death Study project set up by the RGI in collaboration with global health experts concluded that due to lack of healthcare facilities like doctor availability, lack of hospitals, unavailability of medicines, etc. lead to an increase in the mortality rate for rural males as compared to urban males[1]. Also, the same case was observed in the case of females.

B. Potnuru's study published in the Indian Journal of Public Health discussed the availability of doctors in India [2]. The result showed that for the year 2014, the doctor-population ratio was 4.8:10,000. It concluded that even by considering the growth prospects, it seems impractical to reach a doctor-population ratio of 1:1,000 in the next 15 years.

The above two studies show that there is a huge need in India to provide the medical facilities to the rural population at affordable rates and ensure that no remote area of India is deprived of quality healthcare. Thus, the challenge lies in providing low-cost working solutions.

L. L. Wyse's paper related to CNN discusses that as compared to visual images, choices are less for the case of audio and different representations have been used for different applications for the latter [3]. It rules out representations like MFCC, as it is a lossy representation. Since spectrograms are 2D plots having time and frequency on X and Y, respectively, the author makes a point that the CNN architectures for images can also be directly applied to sound. As compared to images using two layers for the neural network, audio requires a single layer only. Also, it suggests the use of log frequency representation as audio objects comprising energy across frequency, and a change in the pitch of the sound will result in a change in spatial extent. Thus, it summarizes that spectrograms have a significant role in applications that make use of neural networks as they retain more information. This paper shows that CNN is applicable to audio files as well and requires a single layer only. Also, the spectrograms would have a log frequency representation.

Another paper discusses the creation of mHealth system comprising of mobile-based, point-of-care CDS tool to assess and manage CVD risk [4]. This heart risk prediction tool was provided to health care workers and doctors for preliminary testing. The preliminary utility and efficiency of the CDS tool were obtained through pilot testing. The pilot testing was done in an Indian village on 292 participants above the age of 40. The result showed that 34% of the participants were detected as high CVD risk and were suggested to visit a doctor. The study concluded that with a third of its participants testing positive, there is a great need in India to work on CVD detection in rural areas.

The paper by Uguz in Journal of Medical Systems discusses the limitations of the use of a stethoscope by doctors in identifying abnormalities related to heart and thus, comes up with the solution of reducing these limitations by developing biomedical based decision support system [5]. The system consists of three stages- feature extraction, dimension reduction, and classification. In the feature extraction stage, features that represent heart sound signals are obtained by making use of DFT. It uses the AR to calculate the power spectrum density of each signal. PCA is used as a dimension reduction technique. The classification was carried out by applying the features as input to ANN. The heart sounds are obtained using an electronic stethoscope which makes use of noise reduction technology to reduce the surrounding noise.

Another work related to classification of heart sound published in EURASIP Journal on Advances in Signal Processing proposes the use of the 1D CNN model and divides heart sound as normal or abnormal irrespective of ECG readings [6]. Further, it compares the 1D CNN model with other models like BP neural network, HMM, and 2D CNNs in terms of effectiveness and accuracy. The DAE algorithm is used for extracting features from heart sounds, which can be provided to 1D CNN as input, and thus, 1D CNN can perform pooling operations. For the input to CNN, it does not use the conventional MFCC. A softmax classifier classifies the processed signals. It concluded that the accuracy of 1D CNNs was better than 2D CNNs, and in the case of 1D CNNs, accuracy can be improved by 1% with the help of 2D CNNs. Also, the method discussed did not require any pre-processing, which was an advantage and could classify heart sounds successfully.

R. Saracoglu's paper uses DHMM based system for classification of the heart sound. It studied two data sets, one that required pre-processing and another which did not require pre-processing [7]. For the first dataset, pre-processing was done using DFT, and PCA was used for feature reduction as PCA is a popular method for feature reduction in the biomedical field. The other data required steps from feature reduction. Then, DHMM is used for classifying reduced features. This method provided successful results, and its accuracy can be improved by applying different feature extraction methods. Also, this system proves to be helpful to less experienced doctors.

The use of the LMS algorithm to improve the performance of the LSSVM classifier for classification of heart sounds as normal or abnormal was used in one of the works [8]. It uses RBF kernel function as a classifier for recognition of heart sounds and uses wavelet-based feature vectors as input. The heart sound spectrum is divided into sub-bands to extract features. It was observed that the accuracy was approximately 4.2% better than the standard LSSVM. Also, it suggests that this technique can be used in rural areas or mass camps where such facilities are unavailable.

The datasets of heart sounds were obtained from Open Michigan Educational Resources [9], PASCAL classifying heart sounds [10] and PhysioNet [11].

The above papers gave us the idea to create a low-cost tool that can detect CVD in preliminary screening by classifying heart sound as normal or abnormal. This project, if implemented in real life can be beneficial in case of rural areas, especially in a developing country like India where the medical facilities and doctor availability is rare. These devices can be used by NGOs in helping villagers with the detection and not much knowledge is required for operating the system. Thus, the early detection can be used to refer the high-risk cases to the doctors who can provide the necessary treatment.

CHAPTER III: EXPERIMENT WORK

The project is bifurcated into three crucial parts namely Data mining and preprocessing, Classification model and Deployment. Various experimentations were carried out in these three fields and with every step better results were achieved with optimized methods.

3.1 Taking input from microphone & Producing Real Time Spectrogram

A real-time spectrogram of the incoming audio signal is displayed and is recorded for 8 seconds. It also gives us an option to capture the screenshot of the spectrogram. The file is then saved in the local directory as **file.wav** and can be heard. It can also be passed through again to produce another spectrogram. The next audio signal recorded will be overwritten on the same file.wav file. The sub axes can be edited as per requirements of the user. Further details can also be added to the spectrogram. The below real time spectrogram or spectrum analyzer is obtained.

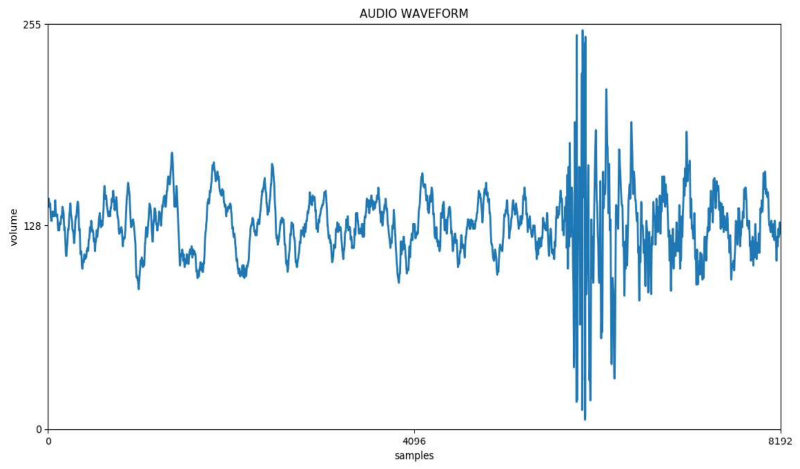
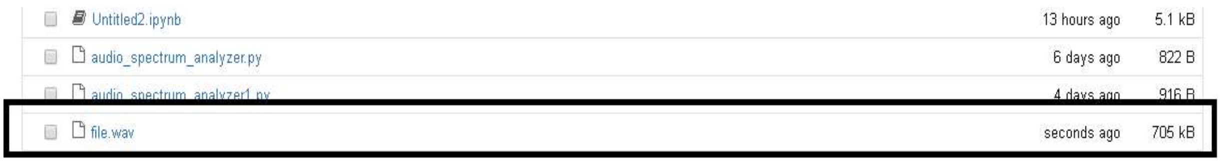


Fig (3.1): Real Time Spectrogram

3.2 Storing Data

The Audio that is recorded for the duration of 8 seconds, is then saved by defining the file using proper extensions. In our case we use **'.WAV'** format. This file can be re-written in a different format if required. This file is stored in the working directory for easy accessibility and to carry out further preprocessing steps. This audio has varied frequency and needs to be filtered out by passing through LPF and implement STFT, a signal processing algorithm, for data preparation.



Untitled2.ipynb	13 hours ago	5.1 kB
audio_spectrum_analyzer.py	6 days ago	822 B
audio_spectrum_analyzer1.py	4 days ago	916 B
file.wav	seconds ago	705 kB

Fig (3.2): Saved .WAV file in working directory.

3.3 STFT & LPF

STFT is used in the preprocessing part to generate spectral images from audio files using Python. A spectrogram is a 2-D plot which has time in seconds on the horizontal axis, frequency in Hertz on the vertical axis and amplitude at a particular time and frequency represented by the third dimension, i.e. colour in decibel. The blue colour on the plot represents approx. 60dB and red represents approximately 220dB. The python code consists of four functions as follows:

1. A function was created to implement a low pass filter, which is a filter that does not affect low frequencies and rejects high frequencies above the cut off frequency. For the project, applying a software low pass filter is essential so that we can eliminate the frequencies above the 500Hz range since the heart sounds lie in the range of 20Hz to 500Hz. Thus, frequencies above 500Hz are removed for further computation.
2. The next function performs the work of calculating STFT by selecting the overlap factor as 0.5. Before STFT computation, zero padding is done to the input to obtain more accurate amplitude estimates of resolvable signal components. After performing various trials with the window types for STFT calculation, we concluded that the Hanning window worked the best in our case.
3. The third function is to list the frequency bins that would help in generating the spectrogram.
4. The last function is for generating the spectrogram.

3.4 Comparing Output With and Without LPF

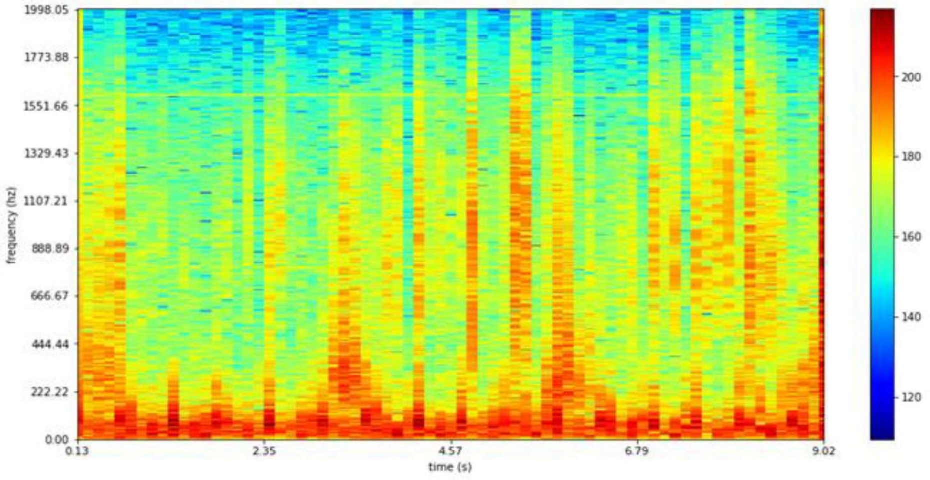


Fig (3.3): Spectrogram without Low Pass Filter

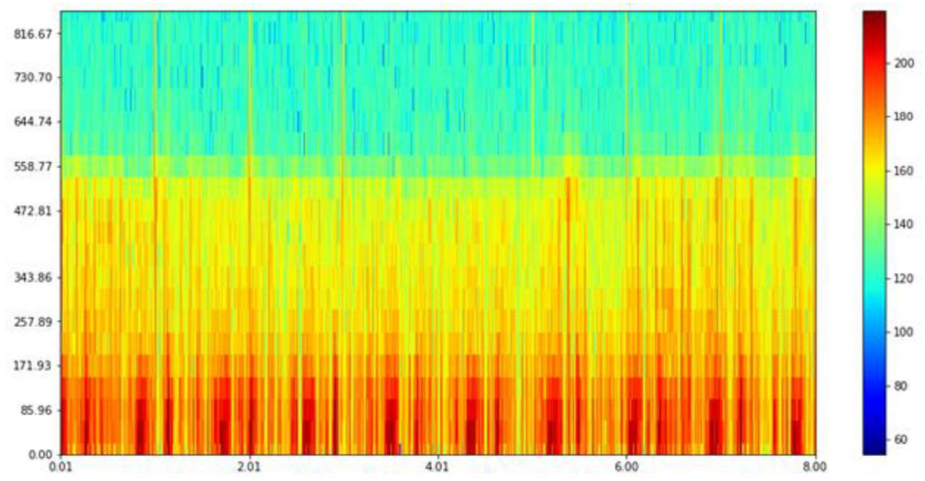


Fig (3.4): Spectrogram using Low Pass Filter

3.5 Introduction to the model

Classification of images is a supervised learning problem: identify a collection of target classes that are objected to be defined in pictures and use labeled example photos for training a model. However, raw pixel data alone do not provide a consistent representation enough to accept the myriad changes of an object as captured in an image. Early computer view models used raw pixel data as input for the model. The location of the object, background, ambient lighting, camera angle, and camera focus can all contribute to variations in raw pixel data, which are significant enough not to be corrected by taking weighted pixel RGB value averages.
Classic computer vision models also introduced new pixel data features such as color histograms, texture, and forms to model objects more flexibly. The drawback of the approach was that the feature extraction became a real burden as there were so many inputs to tweak. Features help train the model better and are input variables that help the model make predictions. Because features had to be so precisely defined, the construction of robust models was tough and often suffers a loss in making a precise prognosis.

3.6 Convolutional Neural Networks

An advancement in building picture categorization models accompanied the revelation that a CNN could be utilized to step by step extricate increasingly elevated features in a picture. Instead of preprocessing data to create features like textures and shapes, CNN takes raw pixel data from the image as input and "learns" how to extract those features, and eventually infers what they are.

CNN receives an input characteristic map: a three-dimensional matrix in which the dimensions of the pixels of the image are specified, which are the length and width, respectively. The third dimension is three corresponding to three color channels (RGB). CNN performs three operations.

3.6.1. Convolution: The convolution extricates tiles from the feature map and filters them, which can be of varying size and depth as the input feature map, to calculate new characteristics, generate an output matrix. The size of tiles extracted (usually 3x3 or 5x5 pixels) is determined by two parameters.

- Tile scale (usually 3x3 or 5x5 pixels) that has been excised.
- The quantity of filters applied, depth of the output matrix.

During the convolution, the filters (matrices of the same tile size) move horizontally and vertically over the grid of the input layer, considering 1 pixel at a time and then drawing out features.

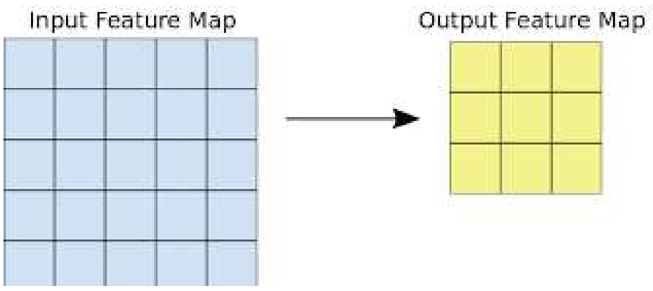


Fig (3.5): Input and Output Feature Matrix

A 3x3 depth one convolution was done on a 5x5 input characteristics map, also with depth 1. Since nine 3 x 3 positions are available for extracting tiles from a 5 x 5 feature map, a 3 x 3 output map is generated. If the output matrix is required to have the same features as the input matrix, zero paddings can be done to achieve this.

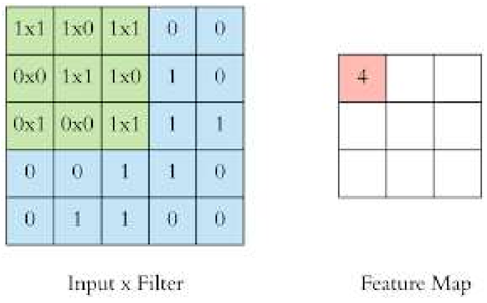


Fig (3.6): Mapping

The CNN continues to elementally multiply the filter matrix and the tile matrix for each filter-tile pair and performs a summation of all the elements to give a single value of resulting matrix. These values are output to the convoluted function map.

CNN will "learn" the optimal values for the filter matrices during training that improves the extraction of essential features of the input feature map. As the number of filters added to the data increases, so does the number of features that CNN can draw out. The tradeoff, however, is that filters make up the bulk of the tools used by CNN so that training time increases as quantity of filters are increased. The addition of filters to the network provides less incremental value compared to previous ones. Hence the aim is to construct a network architecture that uses a least quantity of filters necessary to extricate the characteristics needed for accurate image categorization.

3.6.2. ReLU: The rectified linear unit, ReLU, is used in a variety of neural networks as an activation function. After each convolution process, CNN applies ReLU transformation to the extracted feature to add non-linearity in the model. Mathematically, ReLU can be given as: $F(x) = max\ (0, x)$. The ReLU function returns x for all values of $x > 0$, and returns 0 for all values of $x \leq 0$. Advantages of using ReLU are:

- It is easy to calculate since there is no complex math. This helps the model to take less time to train or run.
- It converges more quickly. Linearity ensures that with the rises in the value of x, the slope does not level or "saturate." This has no problem with the fading of gradients, but this problem is face with other activations such as sigmoid or tanh.
- It is meagerly activated as ReLU is 0 for negative values; hence it is likely that a unit will be active. Sparsity makes the model more condense, which has less noise and overfitting but with a more accurate prognosis.

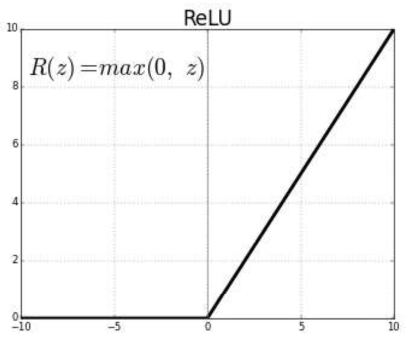


Fig (3.7): ReLU Function

3.6.3. Pooling

After ReLU operation, down sampling is done on the convolved features to bring down processing time. It diminishes the number of measurements of the feature map and holds the most significant details. The normal calculation utilized in this procedure is called max pooling. Max pooling is like convolution. The matrix is slid over, and tiles of characterized size are extricated. The maximum value produced is the new feature matrix for each tile, and other values are disposed of. Two criteria are used for Max pooling operations:

- filter size (usually 2x2 pixels)
- Stride is the distance between each extracted tile and the pixels. The stride decides the position of each extracted tile. A 2 by two filter, which has a stride of 2, indicates that the maximum pooling process extracts from the feature map all non-overlapping 2x2 tiles.

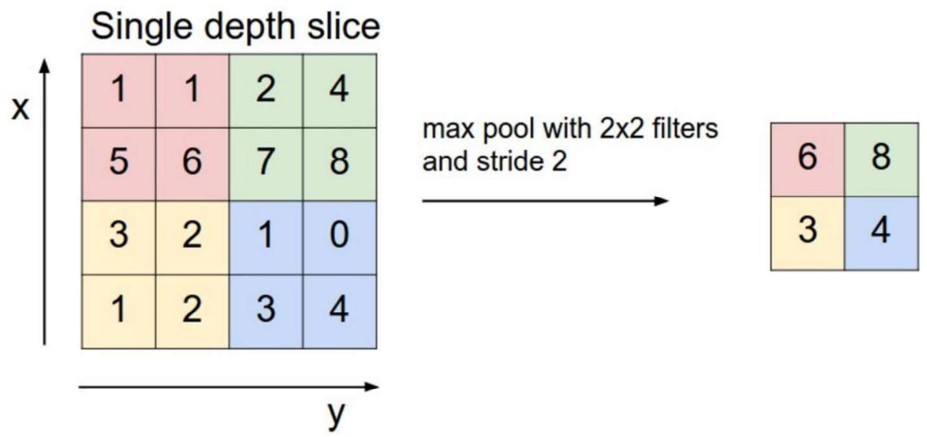


Fig (3.8): Max pooling

In Fig (3.8), Left: a 4x4 feature map underwent max pooling with 2x2 filter and stride of 2. Right: output feature map after max pooling. The resulting map is 2x2, with only the max values for every tile being retained.

3.7 Fully Connected Layers

There are one or more entirely connected layers at the end of a CNN. Fully connected layers are also known as dense layers, where every node in the preceding layer is connected to all the nodes of the next layer. Their job is to identify the characteristics extracted through the convolutions. The final completely connected layer usually consists of a softmax activation function, which gives a probability value of 0 to 1 for each classification label that is expected by the model to predict.

The Softmax work computes the likelihood conveyance of an occasion over 'n' of various occasions. As a rule, this function will compute the probabilities of each target class over all conceivable objective classes. Afterwards, the determined probabilities will be useful in deciding the objective class for the input.

Softmax is preferred because of the range of probabilities. The range is 0 to 1, and the addition of all probabilities is equivalent to one. On the off chance that the softmax work is utilized for the multi-arrangement model, it restores the probabilities of each class and the target class has a high likelihood. This is the fundamental property of the Softmax activation function.

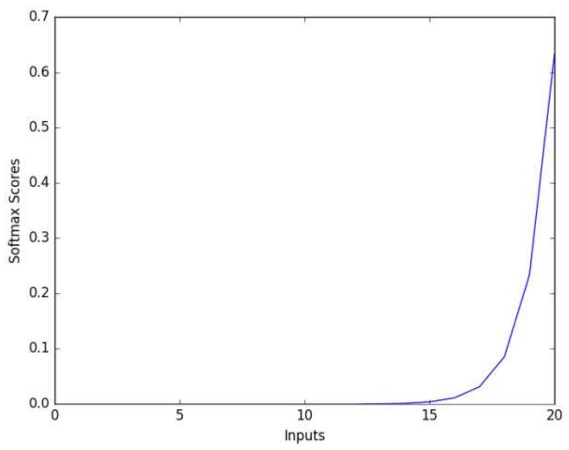


Fig (3.9): Softmax Function

The figure below illustrates an end to end construction of the CNN model. It contains two convolution layers for extraction of features and two dense layers for categorization.

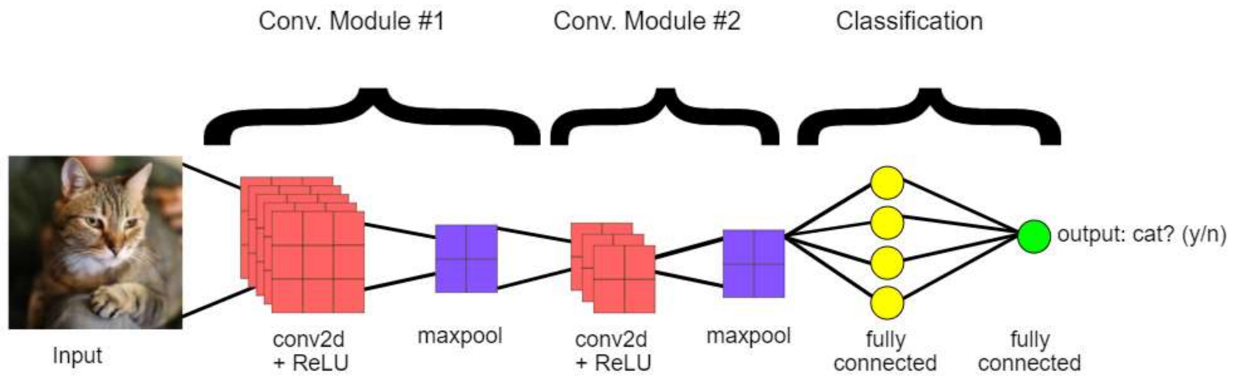


Fig (3.10): Example of a CNN Model

3.8 Sequential Model

Experimenting with different CNN models helps build a classification pipeline for heart sounds. The first model is a Sequential CNN model, and the second is the Inception model.

With sequential and the Functional API, building models layer-by-layer is easy for most problems. The functional API allows building models that are much more versatile since it easy to identify models where layers are linked to more than just previous and next layers. Linking layers to literally every other layer can be done with ease. It also helps to create complex networks. Along with this, it gives an additional advantage to define multiple inputs or output models as well as models that share layers.

An image classifier is assembled utilizing a sequential model to categorize the spectrograms. The sequential model is the simplest neural network model in Keras, which is a linear stack of layers. The audio files were converted to a spectrogram. Then to build the data pipeline for the model to use the data on the disc, the ImageDataGenerator method was used for data preprocessing. This was further used for feature scaling of the training and validation images where the format of the images was converted into floating-point tensors appropriately before feeding them to the network. Images were converted into proper RGB format and the tensor values were rescaled from 0 to 255 to values between 0 to 1 for easier computation.

The model architecture consists of three convolutional blocks with each of them consisting max pool layer in them for down-sampling and discretization. The network consists of a fully connected layer or dense layer with 512 units that use RELU, which is an activation function that activates that unit. To compile the model, ADAM and Binary Cross Entropy were used as optimizers and loss function for the model.

The training parameters for the model consist of the batch size, total number of epochs, and steps for training and validation per epoch. The steps are calculated to be an absolute division, that is, dividing the total number of training data by a number of validation data. During every epoch, the data is sent in batches.

After generating the validation plots to evaluate the model's performance, the validation accuracy and validation loss for the model were 85.37% and 86.35%, respectively. From the graph, it's seen that the model over fits since the validation loss id far higher than that of training loss, which is a noticeable sign.

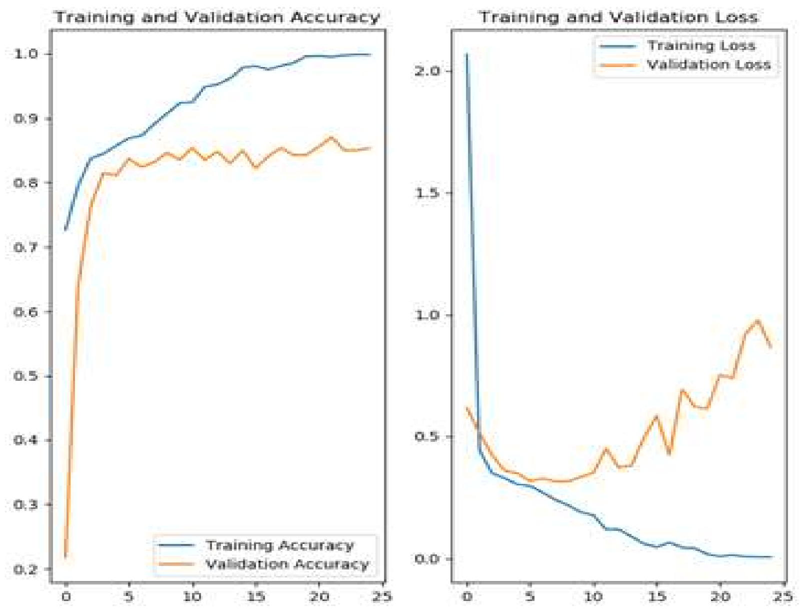


Fig (3.11): Evaluation I

3.9 Overfitting

Overfitting refers to a model that model training data too well. It happens when the model learns the details and noise of the training data that it affects the performance of the model negatively on the new data. A general rule of thumb is that the model starts with overfitting, and then takes measures against it. The steps are taken to overcome it are:

- Progressive resizing
- Regularization
- Addition of data
- Change in Architecture for better Generalization

After preprocessing input image size was 625X1260. The previous model was trained on these parameters. Progressive resizing, as the name suggests, is to gradually resize the image taking the ratio of the "height" to "width" into consideration. Scaling down the images resulted in a decrease in the validation loss. Since the small size of the images help the model to generalize well compared to larger image size, the training time is also significantly reduced. This, in turn, helped to be a useful approach to image modeling for overfitting.

Other parameters that were changed in order to battle overfitting are the "batch size" and "epochs.” In the event that the model is iterated for an excessive number of epochs, it might make the model over-fit the training data. It implies that the model doesn't get familiar with the information; it remembers the data. Hence, the number of epochs was reduced from 50 to 25.

The batch size can be defined as the is the number of training examples sent to the model in a one forward or backward pass. If the batch size is increased, more memory is used while training. Increasing the batch size results in better performance of the model and helps converge faster. Due to this large size, more accurate gradient is attained that can simultaneously optimize loss over a large set of images.

Adding more data helps the model map better. Also, increasing the data may reduce the chance of memorizing as it often happens in small data sets. With the ascent in the number of training examples, the model can diminish the generalization error, making the model robust. Data augmentation was not used since the data is based on time slices, hence flipping or tilting the existing data won't help reduce the overfitting.

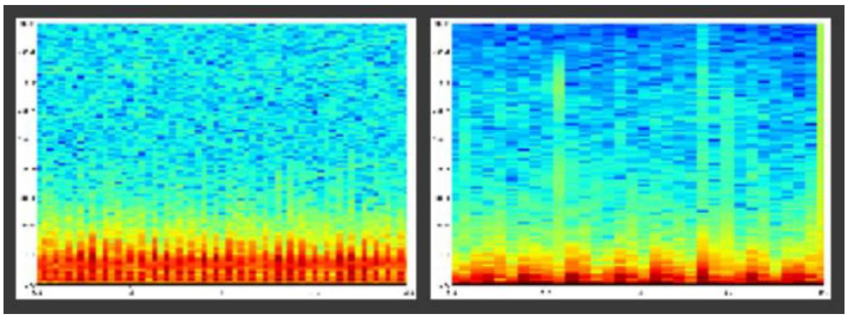
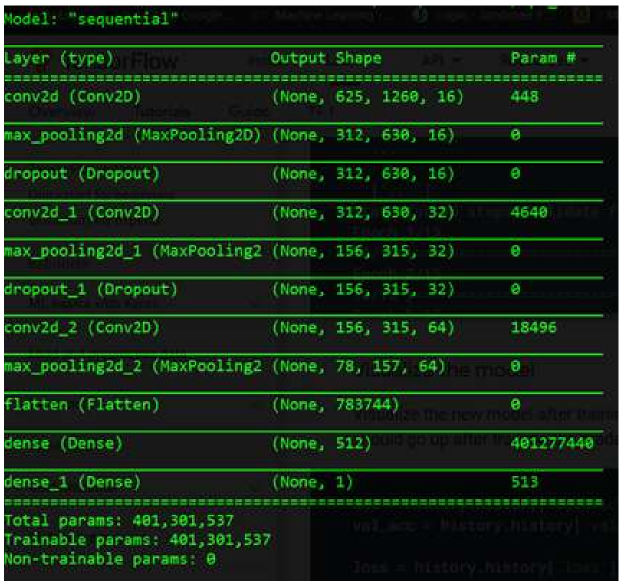


Fig (3.12): Input to the model

Another technique to reduce over-fitting is the introduction of network dropouts. It is a kind of regularization which forces weights in the network to acknowledge little qualities, which regularizes the distribution of weights and lessens overfitting on fewer data. Dropout is one of the regularization procedures utilized in this model. At the point when you apply dropout to a layer, various yield units are arbitrarily dropped from layer during the training. It takes some number which is a fraction as input, for example 0.6, 0.7, etc. applies that 60% or 70% of the output units will be discarded randomly from the applied layer.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 625, 1260, 16)     448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 312, 630, 16)      0
_________________________________________________________________
dropout (Dropout)            (None, 312, 630, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 312, 630, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 156, 315, 32)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 156, 315, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 156, 315, 64)      18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 78, 157, 64)       0
_________________________________________________________________
flatten (Flatten)            (None, 783744)            0
_________________________________________________________________
dense (Dense)                (None, 512)               401277440
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 401,301,537
Trainable params: 401,301,537
Non-trainable params: 0
```

Fig (3.13): Model Architecture I

On visualizing the performance of the new model, there is a significant reduction in the validation loss, from 86% to 45% approximately. But at the same time, there should be an increase in the validation accuracy but a loss in the accuracy for prediction is noticed. The validation accuracy, which was about 85% earlier, but with the new model, it has decreased to

82%. In Fig (3.14), it is clearly visible that the model has a significant dip in the prognosis accuracy but, on the contrary, has improved on the losses.

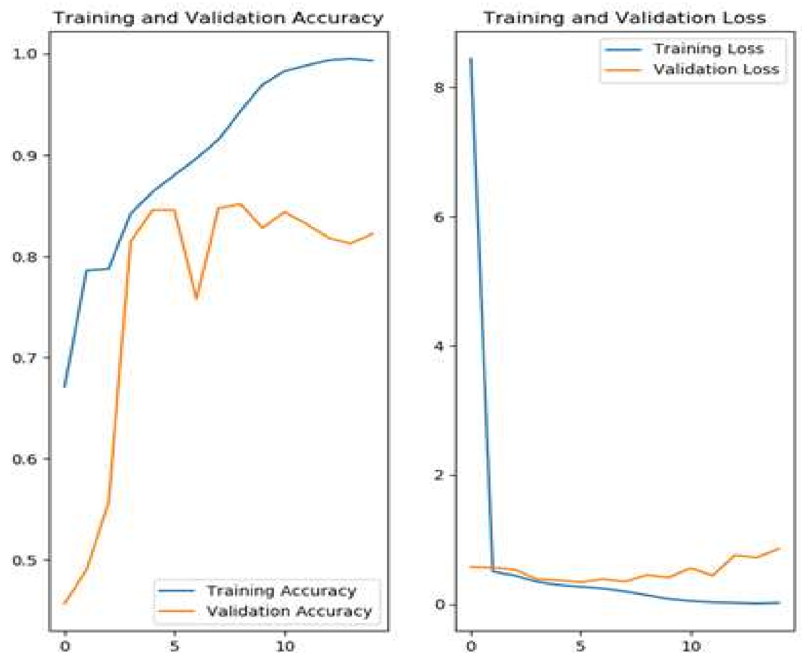


Fig (3.14): Evaluation II

Further improvement that was need to do was to increase the validation accuracy while keeping the model from overfitting as well as underfitting by keeping validation loss to a minimum and not let the training accuracy be less than the validation accuracy. In addition to this, the test accuracy was also to be checked by using new data while restoring the best weights and preserving the state of the optimizer. During training, the model may have reached a level where it can perform at its best at a particular epoch. But when deployed, the model will use the last epochs' details to classify the data. In order to use the best state of the model, "Early Stopping" mechanism was implemented.

3.10 Early Stopping

It is a regularization system used to dodge over-fitting when preparing the model with an iterative strategy, for example, gradient descent. The two methodologies prepare the data to be more apt for each iteration. To a certain extent, this improves the performance of the model on data outside the training set. Be that as it may, past that point, improving the model's fitness to the training data comes with an increased error in Generalization. Early stooping principles give direction on what number of repetitions can be made before the model starts to over-fit. Its standards offer directions about what number of attempts can be made before the model begins to once again over-fit. Its principles have been utilized in a few diverse AI procedures, with fluctuating degrees of hypothetical history. The early stop is a method that allows assigning an arbitrary number of training periods and stopping training until the output of the model stops improving the data set of validation. This includes a split validation to be given and the Early Stop callback to determine the output variable on which the output will be checked on the split validation. The trigger to stop training is when the chosen performance measure stops improving: Validation loss. At this point, the model will return the number of epoch and the step at which the early stopping algorithm was triggered.

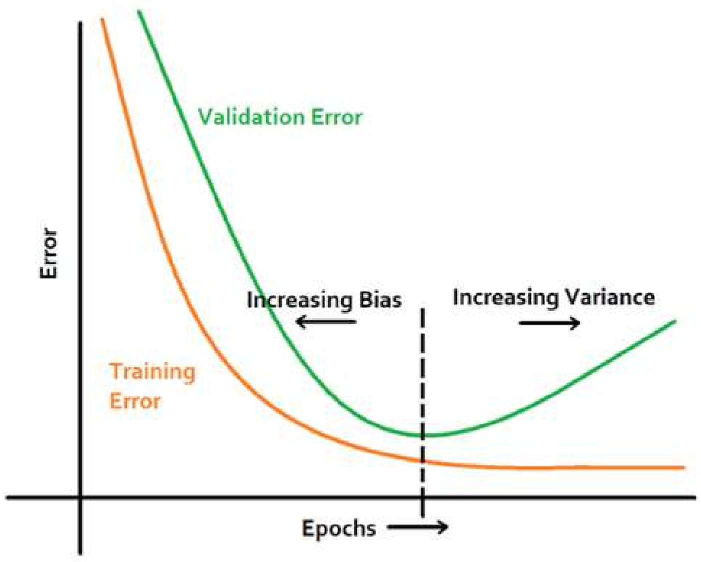


Fig (3.15): Monitoring Validation loss for Early Stopping

The parameter used to keep watch over the training epoch for an arbitrary amount of repetitions, after which the algorithm comes into the picture is patience. But due to this, a drawback is faced. The model retrieved after the specified amount of patience may have more loss. Also, with this it is difficult to get the best model through the entire learning process. Hence by using other callbacks, the best weights are restored, which has the least loss. After training, the entire model has been saved. Due to this, the state of the optimizer is also preserved. This is helpful while launching the model for predictions as the accuracy is maintained while testing of the model.

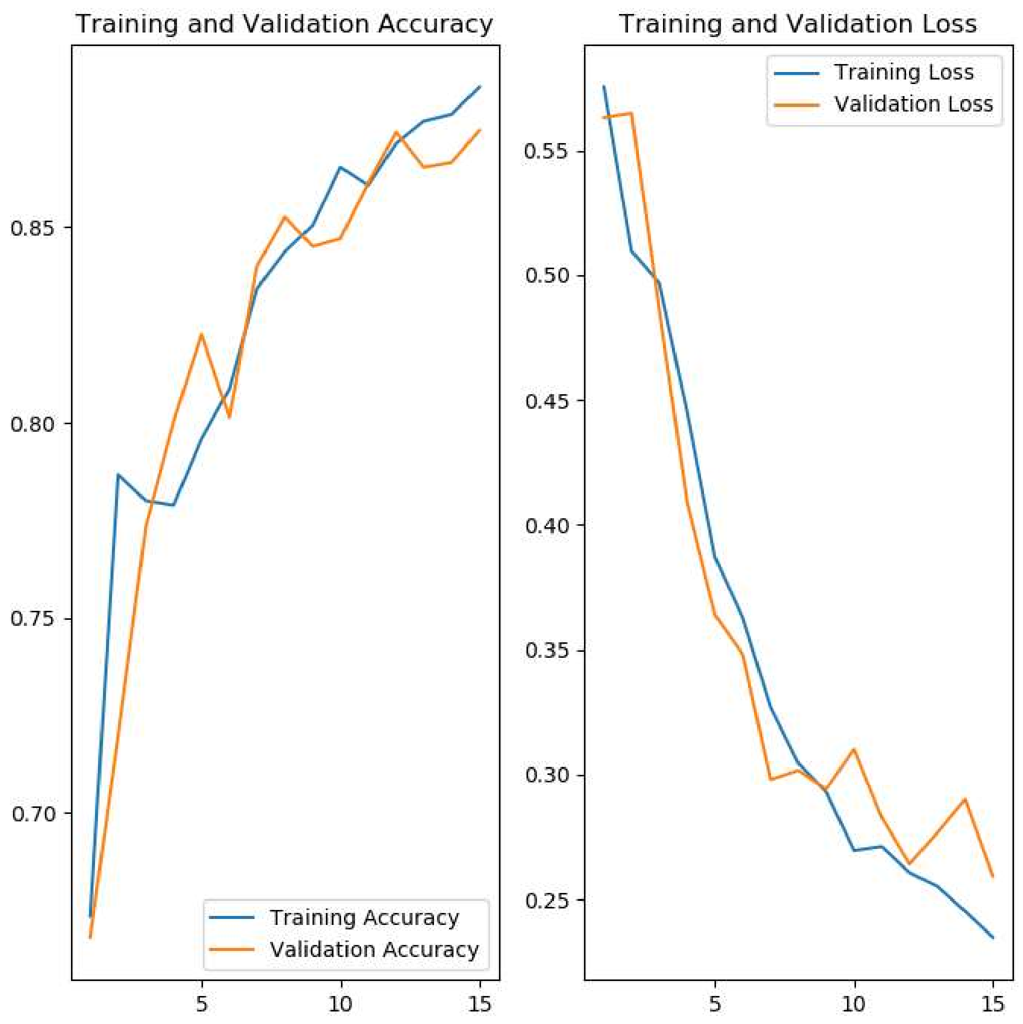
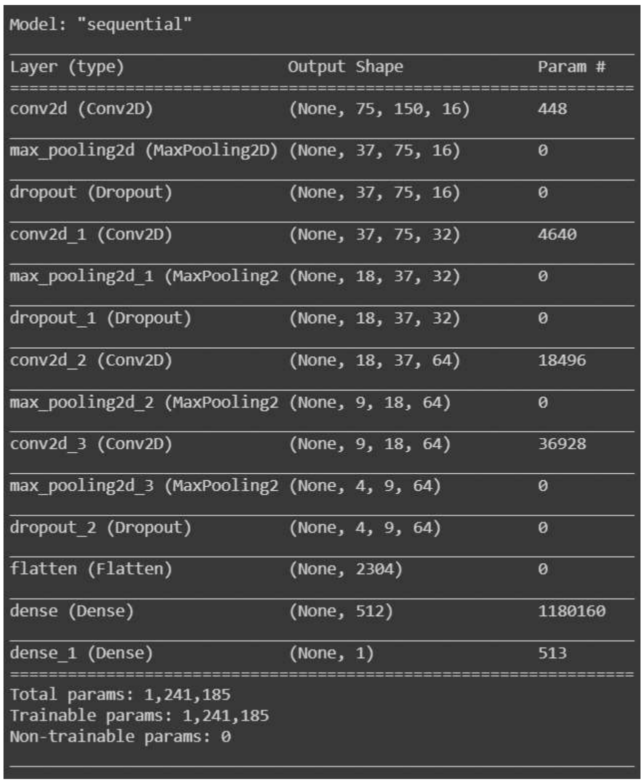


Fig (3.16): Evaluation III

The above evaluation of the model shows that the model doesn't suffer from overfitting. From the graph, you can see that the validation loss is close to the training loss. Also, the training accuracy is more than that of the validation accuracy, which shows that the model is not underfitting either.

The model attains the following parameters after implementing the Early Stopping Algorithm: training accuracy -90 %, validation accuracy -87%, Training loss 23-%, validation loss-26%. The final model architecture is given in the image below.



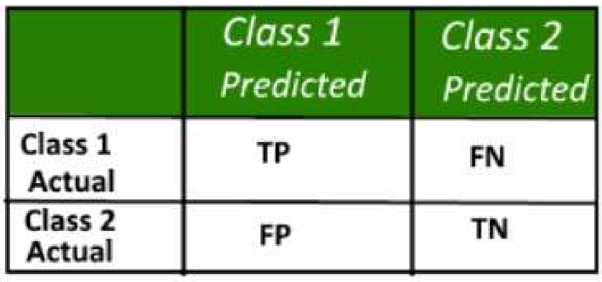
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 75, 150, 16)       448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 37, 75, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 37, 75, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 37, 75, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 18, 37, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 18, 37, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 18, 37, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 9, 18, 64)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 9, 18, 64)         36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 9, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 9, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1180160
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,241,185
Trainable params: 1,241,185
Non-trainable params: 0
_________________________________________________________________
```

Fig (3.17): Model Architecture II

3.11 Confusion Matrix

In the field of AI, and explicitly the issue of statistical classification, there is an uncertainty matrix, otherwise called an error matrix. It is a table that is frequently used to characterize the yield of an arrangement model (or classifier) on an assortment of test information for which genuine qualities are known. It helps imagine the yield of a calculation. This permits basic acknowledgement of uncertainty between classes, e.g., one class is generally inaccurately marked as the other. Most yield measurements are determined from this matrix.

It is a portrayal of the impacts of the forecast on the issue of arrangement. The quantity of exact and off base forecasts is summed up by tally esteems and separated by class. This is the way into the Error matrix. The matrix uncovers how befuddled the model is the point at which it makes predictions. This gives us understanding not just into the mix-ups made by the classifier, yet additionally, more fundamentally, into the kinds of mistakes that are being made.



	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Fig (3.18): Standard Confusion Matrix

3.11.1 Definition of the Terms

- Class 1: Positive
- Class 2: Negative
- Positive (P): Observation is positive
- Negative (N): Observation is not positive
- True Positive (TP): Observation is positive
- False Negative (FN): Observation is positive
- True Negative (TN): Observation is negative
- False Positive (FP): Observation is negative

3.11.2 Classification Rate/Accuracy

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

3.11.3 Precision

$$\textit{Precision} = \frac{TP}{TP + FP}$$

3.12 Setting Up Raspberry Pi and Raspbian OS

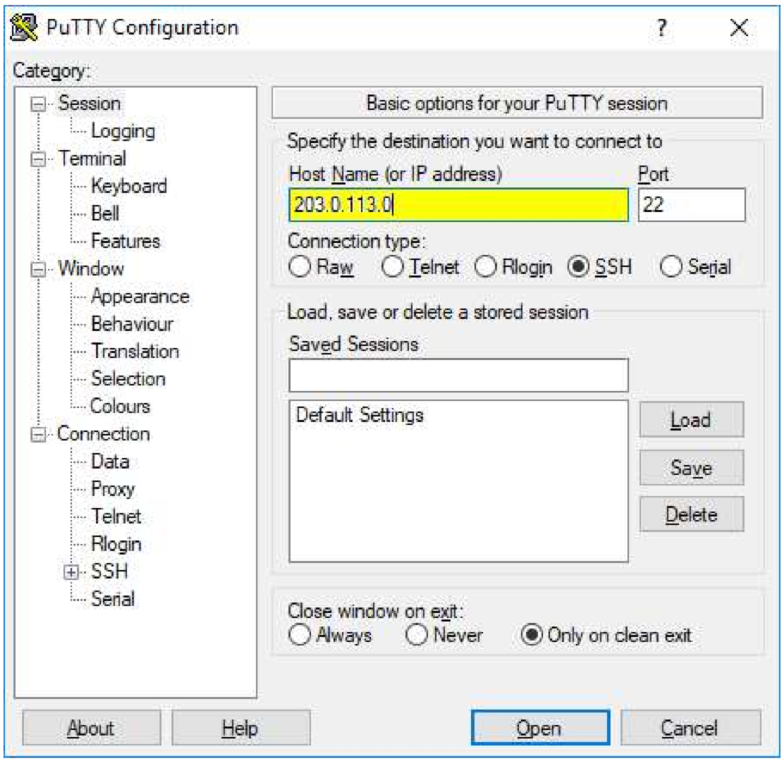
Raspbian is the preferred operating system for the Raspberry Pi – Zero.
Raspbian is the most desirable operating system for any general use on the Pi. It is a free OS that is based on Debian-Linux OS, optimized for the Pi hardware. Raspbian has more than over thirty-six thousand packages: which are software available for this format for simple installation on your Raspberry Pi. The project used Balena Etcher to burn the Image on the SD card. The Image Burn SD card has the Raspbian OS, which has to be installed on the first Boot Up.

3.12.1 Booting Raspberry Pi Headlessly

To use a monitor or keyboard to run your Pi (known as headless), but it will still require to do some wireless settings, there is a facility to allow wireless networking and SSH when creating an image.
Once an OS image is created on an SD card, by burning it onto an SD card reader on a Linux/Windows machine, the boot folder can be located and edited. Adding specific files to this folder will initiate several setup features on the first boot of the Pi itself.
Setting up wireless networking
It will require to have a wpa_supplicant.conf file for your appropriate wireless network, which we want to connect when we boot up during the first start-up. This .conf file in the boot folder located in the config settings of our boot-up folder, and when the Raspberry Pi first boots. It will copy this file into the exact section in the Debian root file system and use these settings to start up our Raspberry Pi wireless networking to connect the Wi-Fi.

3.12.2 Enabling SSH

Secure Shell is often used protocol to secure networks for working securely across unsecured channels. For purposes like remote command-line,any network can be thus secured with SSH.
SSH can be allowed to execute by placing a file called ssh into the boot folder. This flags the Pi to allow the SSH into the system on the next boot.
An SSH client like putty to ssh into our raspberry pi and install VNC Server.
Putty configuration -
username: lemonpi
password: applepi


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 203.0.113.0

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Fig (3.19): PuTTY Terminal Login

3.12.3 VNC Server.

To use the Rasberry Pi with a graphical user interface instead of the command-line interface for ease of understanding and demonstration. VNC is a graphical desktop server that allows us a sharing arrangement that permits us to remotely access the desktop GUI of one PC, which is running the Server, from another PC or device which is running VNC Viewer. It sends the keyboard strokes and click events to VNC Server launched and receives updates in real-time to the display on our viewer PC in return.

We will see the Raspberry Pi on the VNC Viewer on our connecting PC or mobile device. This allows us to control the server and Rasberry Pi remotely from anywhere. We enable VNC using its specific commands.

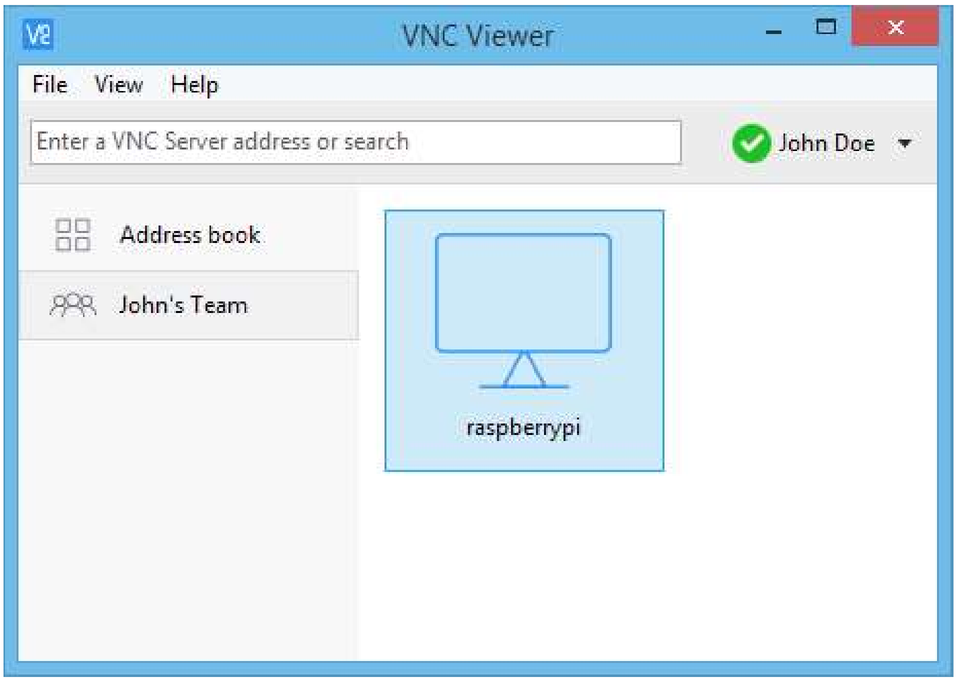


Fig (3.20): VNC Viewer

The VNC server is active, and our setup of Raspberry Pi is complete.

3.13 Cloud Computing

Cloud computing is when needed on the availability of system resources, like computing power and data storage, without immediate ongoing control by the user.

Cloud computing is the providing of various services like databases, servers, storage networking, analytics, software, and AI , aka 'Cloud' to offer more agile work, adaptable assets, and scalability. You spend solely on services you use, which helps us lower our running costs, use our resources more efficiently and scale as per our needs.

3.13.1 Benefits

The above model has been developed locally our projects aim to process all the images on the Cloud for the following benefits -

1) The local microcontroller does not enough CPU power to run a neural network prediction.
2) The computation time is far lower compared to any local-machines even if we include delay in transmission of data to and from the server.
3) Cloud architecture is scalable with demand.

4)The architecture and model are accessible from everywhere.
5) It provides us with the off-site backup of all the data we collect in actual testing to improve the model further.
6) It provides robust security , provided by the people working the domain with frequent updates

3.14 Backing up input .WAV file (Audio File to Google Drive)

The main reason to backup the obtained data is to use this new data collected from actual testing to improve our model further and keep improving it. Neural Networks have proven to be much better at predicting the more considerable the amount of dataset they have to train on for the model. This accomplishes the task of having backup data on the Cloud instead of the device which has low memory and will require onsite interaction with the device.
The clone script that we are going to use for this is from Rclone.
Rclone *("rsync for cloud storage")* is a command-line program to sync files and directories to and from different cloud storage providers.

3.14.1 Features

- Timestamps saved on files
- Unfinished syncs kept on a complete file basis
- Copy mode to mimic changed files
- Sync It is a (one way) method to make a directory identical
- Can be synced to and from various networks
- Able to serve local or remote files over HTTP

Configuring a Remote Storage Service requires to set up one of the rclone remote storage API on a raspberry pi computer, the following steps need to be followed. If a service is already installed, we will see a configuration menu that allows us to change the services for the remote Storage.

3.14.2 Steps for Google Drive

Open putty login SSH session to Raspberry Pi and execute the commands which are given below, then heed rclone interactive config pro.

3.14.2.1 rclone config

- We can select new, At the name,> prompt, Enter our preferred service name, example- google media
- At the Storage> prompt   enter for google drive
- At the auto-config menu? We should select n
- rclone will display a long URL at the link: prompt.
- On the RPI SSH session, click and hold the left mouse button and highlight.
- On the system web, service security web page, Select user account if appropriate. On service security web page confirm access.
- On the authentication page, a Long confirmation code will be used represented—Right-click to copy verification code.
- On RPI SSH session at prompt Enter verification code> right-click, paste verification code then Enter to accept
- At team drive? quick Select n if the drive is not part of a team
- Review and allow settings when prompted.

3.14.2.2 Using rclone

```
rclone copy -v /home/pi/rpi-sync gdmedia:/rpi-sync
```

This command copies the files from rpi-sync folder in RaspberryPi to the Gdrive folder rpi-sync folder. The above line of code with the script written by the team allows us to update any files created in rpi-sync folder to push it on to the Cloud. We wrote a script to automate the above task in raspberry pi so that it checks a .WAV file in the specified directory and uploads it to the Cloud.

3.15 AI Model on Cloud

The Next step is to deploy our AI model on Cloud -

Google Cloud Platform (GCP) is a set of tools offered by Google; it is a suite of various cloud computing services that runs on the same infrastructure, which googles internally runs for its

products like Gmail, YouTube, and Search. Adjacent with an assortment of administration mechanisms, it implements a range of cloud services.
The project will be using an AI platform for deploying our AI model. The project is using the AI platform to train our deep learning (DL) models, to deploy our trained model for prediction in the GCloud, and to use our model to make decisions about newly acquired data.

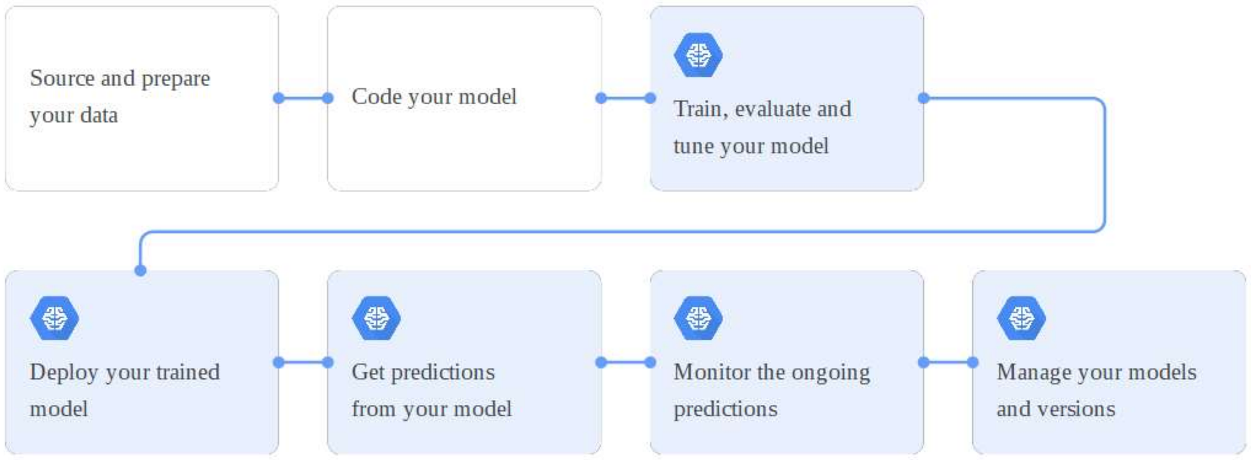

Source and prepare your data
Code your model
Train, evaluate and tune your model
Deploy your trained model
Get predictions from your model
Monitor the ongoing predictions
Manage your models and versions

Fig (3.21): AI Platform Workflow

The project is interested in the service of the google AI platform, it allows us to make prediction with our own hosted model irrespective of the platform on which it was trained.

3.16 Deploying the Neural Network AI platform

The Neural Network, the model, built before has to be modified so that the model can be deployed on the AI platform. The Neural Network model above accepts Tensors as inputs, but we need to write the serving function for it.To serve predictions from Google's AI Platform, it must export our trained ML model as an artifact in the bucket.
The model must be exported as the estimator model from TensorFlow. This model requires a serving function; This requires a serving input function for the model to be able to make online predictions. The serving_input_fn in the pipeline.py is required for it to provide to the predict() method. It is basically telling the model what data it will be getting from the user. The project requires the serving function so that the model can take input as base64 binary data.
To deploy our trained model on AI Platform, It is needed to:

- Upload our saved TF model to our 'NN' Cloud Storage bucket.
- The AI Platform Prediction model resource named 'Heartrisk2 named was created.
- Creating the AI Platform Prediction version, defining the Cloud Storage path to your saved model 'Heartriskv2' in the proper bucket.

3.16.1 Store our model in Cloud Storage

It is recommended that a dedicated bucket is used for storage of the model you're applying for the prediction. Accessing appropriate permissions that have to be taken as an API key. We require these permissions otherwise; the request to build a model version fails. The Neural Network is stored in the 'NN' named bucket.

3.16.2 Deploy model and deploying its version

An Prediction model is a vessel for the different versions of our AI model, which we created. To deploy this model, we create a model resource and link it to our model version. The project needs to create a model and versions otherwise it would not deploy this is useful to change versions as and when required by the user.

3.16.3 GCloud

The project requires the google cloud SDK which can be downloaded from official google site.In the following code, it is must replace the model name with our desired name for our model. It is a must to replace region with where we want our nodes to run. Example – north-eastasia-2. It is needed to create a model version and framework.
This will host our Neural network model to the specified bucket and model version as specified in the code, which is passed on to make online predictions.

3.17 Getting Online Predictions

The AI Platform Prediction, online prediction is a service optimized to run our data through our hosted models with as much low latency as possible. The code sends small batches of data to predict the service, and it returns our predictions in the response.

3.18 Formatting our input data for online prediction

3.18.1 Formatting our data as input instances as JSON strings The usually used format for the model prediction is a list of data instances. They are both simple lists of values or parts of a JSON object based on our inputs in our training model. TensorFlow models can accept more complex data but we will be using JSON.

3.18.2 Binary data in prediction input

JSON does not support UTF-8 encoded strings. Thus, in our case images, be must be sent in JSON format. If we have binary data in your inputs, such as images, we are going to use the base64 representation to encode our images .The string has to be encoded inside a JSON format. The following code is used to encode the Image into a base64 instance:
In Python 3.5, base64 encoding gives a byte sequence instead of the same in python2. To convert this to a string to make it JSON serializable format:

{‘inputimage_bytes': {'b64': base64.b64encode(jpeg_data).decode()}}

In our TensorFlow model code, it is required to name the layers for our binary input tensors so they terminate with '_bytes.' This allows us to input data in the b64 format.

3.18.3 Base64

Base64 is most regularly used to encode binary data . the data can be a binary file , a image or audio file. The can they be then used in website and webpages. Also, it is often used for data which is unsupported by any other formats.Thus, this function converts the input of the Neural Network instead of accepting Image tensors to a Base64 input, which can be sent over the network.The base64 data is then formatted into JSON object like shown below- {"instances": [{"values_image": [1, 2, 3, 4, 5, 6, 7(base_64)], "key": 1}]}

These instances are the Image where value is input corresponding to the name of the first layer of the neural network layer, and values associated with it are the data. Keys are often not required for single prediction. We write a function to provide the JSON file as input and returned prediction from the deployed model is then parsed in our script. The output is the score of the activation function based on which we tell the user if he hearts risk or not based on the sigmoid if the output is greater 0, then the heart is categorized as normal and below 0 as abnormal.

3.19 Final Implementation on Raspberry Pi

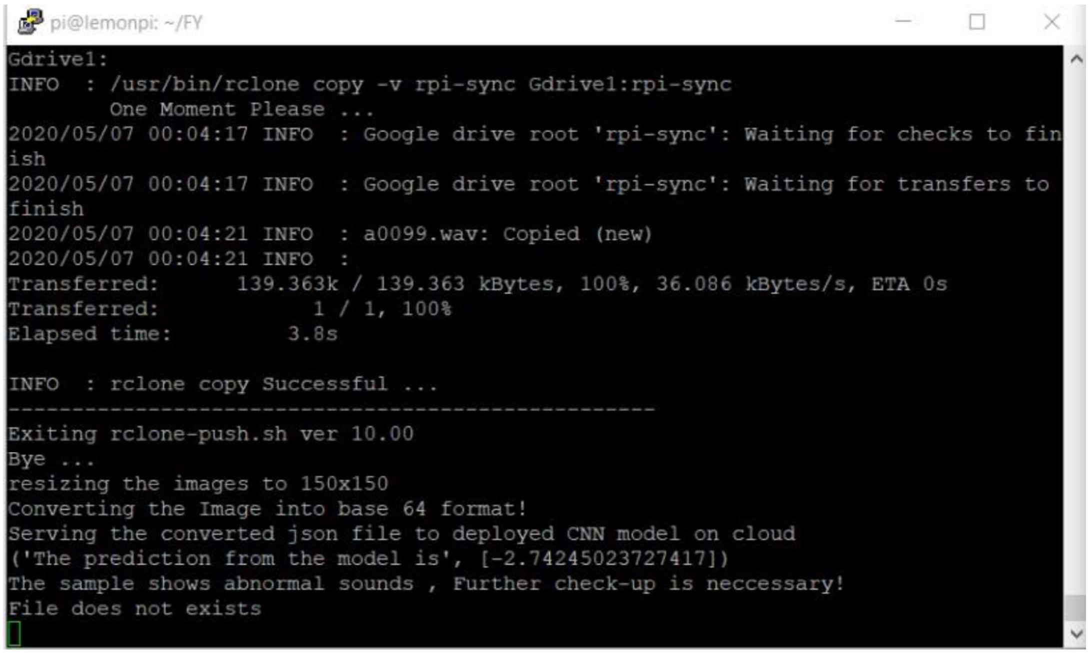
The whole pipeline it to be implemented on Raspberry Pi.

The files are located FY folder on Raspi Home.

Trial.py consists of the following process STFT, LPF, generating spectrograph. Pipeline.py consists of backing up data on the Cloud, converting images to byte64, sending data, and receiving output from the deployed model and outputting the prediction.

Checker.sh files are bash files which when runs does the following –

- Checks the raspi-sync folder if any .wav file is present.
- The .wav is uploaded on the Cloud.
- The trial.py script is launched.
- LPF, STFT is applied, and the output is saved in another folder as a spectrograph.
- Pipeline.py script is launched.
- The Image is converted to base 64 formats.
- The file is sent as JSON format to the deployed model, which returned the output.
- The output is parsed and then displayed on the terminal screen.



```
Gdrive1:
INFO  : /usr/bin/rclone copy -v rpi-sync Gdrive1:rpi-sync
        One Moment Please ...
2020/05/07 00:04:17 INFO  : Google drive root 'rpi-sync': Waiting for checks to fin
ish
2020/05/07 00:04:17 INFO  : Google drive root 'rpi-sync': Waiting for transfers to
finish
2020/05/07 00:04:21 INFO  : a0099.wav: Copied (new)
2020/05/07 00:04:21 INFO  :
Transferred:      139.363k / 139.363 kBytes, 100%, 36.086 kBytes/s, ETA 0s
Transferred:            1 / 1, 100%
Elapsed time:         3.8s

INFO  : rclone copy Successful ...
---------------------------------------------------
Exiting rclone-push.sh ver 10.00
Bye ...
resizing the images to 150x150
Converting the Image into base 64 format!
Serving the converted json file to deployed CNN model on cloud
('The prediction from the model is', [-2.74245023727417])
The sample shows abnormal sounds , Further check-up is neccessary!
File does not exists
```

Fig (3.22): Device predicts Normal Auscultations

The project has implemented all functions described above and are able to get prediction from the cloud in real time given a .wav file. The script needs to be run only once and device will keep functioning until you shut down the device.

CHAPTER IV: RESULT & DISCUSSION

The final Classification rate that we achieved is 92%. The training accuracy, validation accuracy, Training loss and validation loss of the model are 90, 87%, 23% and 26% respectively. This is that proof that the model is neither underfitting nor overfitting. When deployed, the model maintains the state of the optimizer and preserves the best weights. This is beneficial as the Classification rate and the precision is maintained. The precision attained is 0.86. The Confusion matrix computed for the model that we have deployed is given below. We have taken a test data of 539 spectrograms, out of which 421 are abnormal, and 118 are normal.

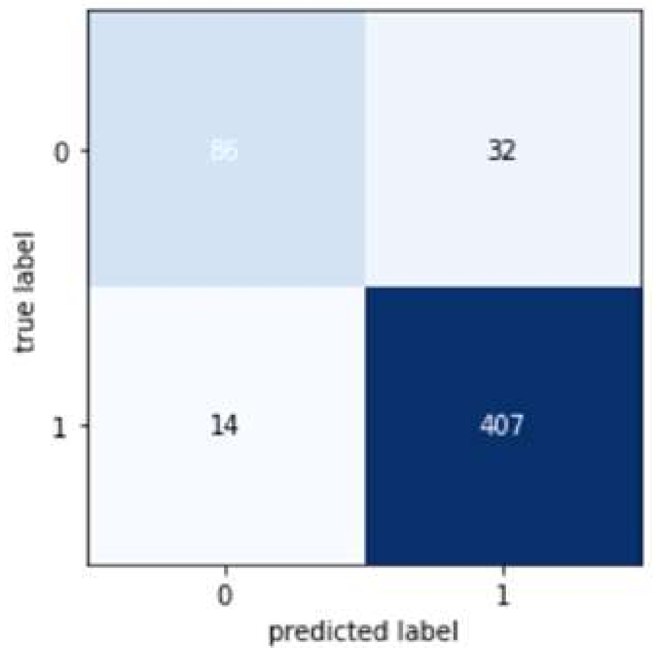
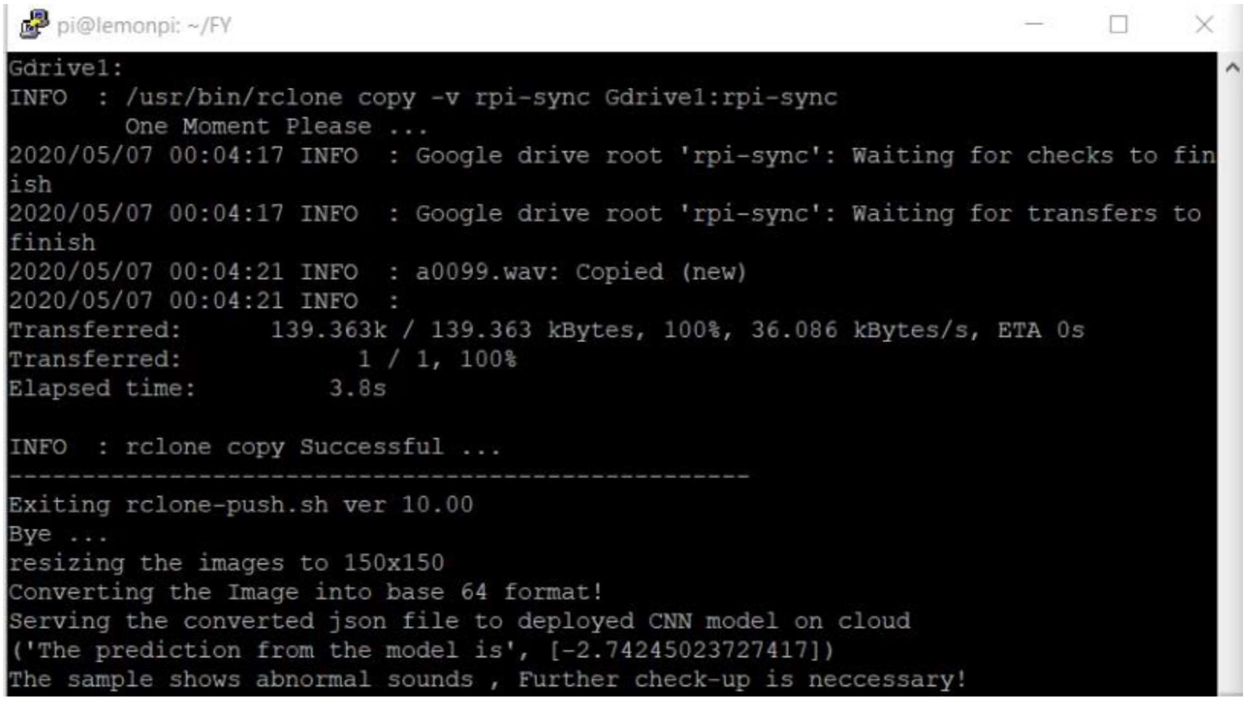


Fig (4.1): Confusion Matrix for the Model

From the given Confusion matrix:

- True Positive (TP) : model predicts sound to be abnormal when it indeed is abnormal - 86
- False Negative (FN): model predicts sound to be normal when it indeed is abnormal -32
- True Negative (TN): model predicts sound to be normal when it indeed is normal -407
- False Positive (FP) : model predicts sound to be abnormal when it indeed is normal - 14

The number of False Negatives (FN) is minimum. False Positives are the examples that have been classified as negative but are indeed positive (in our case abnormal heart sound). Hence the FP in our model are kept at a minimum of 5%.



pi@lemonpi: ~/FY

```
Gdrive1:
INFO  : /usr/bin/rclone copy -v rpi-sync Gdrive1:rpi-sync
        One Moment Please ...
2020/05/07 00:04:17 INFO  : Google drive root 'rpi-sync': Waiting for checks to fin
ish
2020/05/07 00:04:17 INFO  : Google drive root 'rpi-sync': Waiting for transfers to
finish
2020/05/07 00:04:21 INFO  : a0099.wav: Copied (new)
2020/05/07 00:04:21 INFO  :
Transferred:      139.363k / 139.363 kBytes, 100%, 36.086 kBytes/s, ETA 0s
Transferred:            1 / 1, 100%
Elapsed time:         3.8s

INFO  : rclone copy Successful ...
---------------------------------------------------
Exiting rclone-push.sh ver 10.00
Bye ...
resizing the images to 150x150
Converting the Image into base 64 format!
Serving the converted json file to deployed CNN model on cloud
('The prediction from the model is', [-2.74245023727417])
The sample shows abnormal sounds , Further check-up is neccessary!
```

Fig (4.2): Device predicts Abnormal Asculations

From the above screenshot we can see, that device is able function effectively and under a minute give us prediction according to the asculations recorded , the device applies LPF and STFT on audio ,backups the audio onto the cloud , resizes images , converts to b64 format, sends the json file to the deployed CNN model on cloud and returns with a prediction of -2.7. Considering that the final layer has sigmoid function, the device is able to determine that the patient needs to undergo further checks.

CHAPTER V: CONCLUSION AND FUTURE WORK

5.1 Conclusion

We have successfully developed and proven that our hypothesis can be implemented in real-time to Early Screen for Cardiovascular Disorder. There is a need for accuracy in detecting cardiovascular disorders is continuously increasing due to an increase in the number of cases of heart-related diseases. Listening via stethoscope is the primary method, being used by physicians for distinguishing typically any abnormal cardiac systems. However, listening via stethoscope has got several limitations, for interpreting different heart sounds on hearing ability, experience, and respective skill of the physician. Such restrictions may be reduced by developing biomedical based decision support systems. We provide an automated real-time solution wherein the user has to follow simple guidelines to test a subject, and receive his score. This score can then be used to evaluate the patient further and send the patent for further tests to the doctor. Various NGOs can use the device, and people with little or no training can use the device in rural areas and provide primary screening with quick results. This device is very cheap to construct, and cloud subscriptions are incredibly affordable for such tasks as hosting and serving predictions. The model and architecture that we have used can be used screening, testing for various other diseases depending on the input sensor.

5.2 Future Work

As the system acquires more data, we will be working on improving the accuracy of the Neural Network and would also look into ways to make the hardware more cost-effective. We would also test the device on-field and gather information for improving accuracy with data like the history of the patient and their family to give a more accurate suggestion. Complete the hardware with the casing, including LEDs, for indicating whether the heart system is normal or abnormal. Thus, a person with no knowledge of the working can use the device.

We would also work on adding digital stethoscope functions to the device like real-time spectrogram, heartbeats, etc. Also, we can improve neural network by using memory Block such as LSTM's or GRU. Since the spectrogram is in time domain - Convolutional LSTM.

REFERENCES

[1] Heart Diseases in India: What Statistics Show
Available: https://www.livemint.com/Politics/fKmvnJ320JOkR7hX0lbdKN/Rural-India-surpasses-urban-in-heart-diseaserelated-deaths.html
[2] B. Potnuru, "Aggregate availability of doctors in India: 2014-2030," in Indian Journal of Public Health, 2017, pp 182-187
[3] L. L. Wyse, "Audio Spectrogram Representations for Processing with Convolutional Neural Networks", in CoRR abs/1706.09559, 2017
[4] A. Raghu, D. Praveen, D. P. Peiris, L. Tarassenko and G. Clifford, "Engineering a mobile health tool for resource-poor settings to assess and manage cardiovascular disease risk: SMART health study," in BMC Medical Informatics and Decision Making, 2015
[5] H. Uguz, "A Biomedical System Based on Artificial Neural Network and Principal Component Analysis for Diagnosis of the Heart Valve Diseases," in Journal of Medical Systems, 2010, pp 61-72
[6] Fen Li, -Feature extraction and classification of heart sound using 1D convolutional neural networks
[7] R. Saracoglu, "Hidden Markov model-based classification of heart valve disease with PCA for dimension reduction," in Engineering Applications of Artificial Intelligence, 2012, pp 1523-1528
[8] S. Ari, K. Hembram and G. Saha, "Detection of cardiac abnormality from PCG signal using LMS based least square SVM classifier," in Expert Systems with Applications, 2010, pp 8019-8026

Datasets:
[9] R. Judge and R. Mangrulkar, "Heart Sound and Murmur Library", 2015, Retrieved from Open.Michigan Educational Resources
[10] P. Bentley, G. Nordehn, M. Coimbra and S. Mannor, "The PASCAL classifying heart sounds challenge 2011"
[11] Classification of Normal/Abnormal Heart Sound Recordings: the PhysioNet/Computing in Cardiology Challenge 2016, Gari D. Clifford, Chengyu Liu, Benjamin Moody, David Springer, Ikaro Silva, Qiao Li, Roger G. Mark