RESPIRATORY DISEASES RECOGNITION THROUGH RESPIRATORY SOUNDS WITH THE HELP OF DEEP NEURAL NETWORK

CREATIVE AND INNOVATIVE PROJECT REPORT

Submitted by

AKSHARA M S (2019103004)
AKSHAYALAKSHMI V K (2019103505)
SRIHARI T(2019103586)

in partial fulfillment of the requirements for the award of

the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING,

GUINDY ANNA UNIVERSITY:

CHENNAI 600 025

JUNE 2022

ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to our project guide, **Mrs. Lalitha Devi K**, Teaching Fellow, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her constant source of inspiration. We thank her for the continuous support and guidance which was instrumental in taking the project to successful completion.

We are grateful to **Dr.S.Valli** , Professor and Head, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her support and for providing necessary facilities to carry out for our project.

We would also like to thank our friends and family for their encouragement and continued support. We would also like to thank the Almighty for giving us the moral strength to accomplish our task.

Akshara M S Akshayalakshmi V K Srihari T

Table of Contents

ABSTRACT:

Convention methods to detect respiratory methods are expensive and time consuming. Hence, we proposed an approach to recognize respiratory diseases from respiratory sounds. The data set we used contains respiratory sounds of patients who have COPD(Chronic obstructive pulmonary disease), URTI(upper respiratory tract infection), Bronchiectasis, Pneumonia, Bronchiolitis as well as healthy persons. The audios in the dataset contains heart sounds along with respiratory sounds. So after noise reduction, we used ICA to separate respiratory sounds from heart sounds. In order to make the data in the dataset more balanced, we used Info-GAN to augment the data. Finally we used a deep neural network model for classification and tested the results using 6 metrics.

1. INTRODUCTION

Respiratory sounds are important indicators of respiratory health and respiratory diseases. According to the World Health Organization (WHO), lung diseases are the third most common cause of death right after coronary heart disease and stroke. India is under the focus of UNICEF for being the country with highest number of unattended cases of Chronic Obstructive Pulmonary Disease (COPD). COPD silently kills lots of people especially women through indoor air pollution. The prominence and effect of respiratory diseases increased marginally with the advent of COVID-19. Since the ill effects of these common diseases are more, it is important to identify and recognize them.

Artificial Intelligence is one of the fastest growing fields in computer science.In recent years Artificial Intelligence has been applied in many health related fields. Deep learning is a field of artificial intelligence where neural networks are used to classify objects. With the advent of new techniques, neural networks is also being used in medical field.

Lung sound is produced when air flows during the process of respiration. Normal respiratory sounds are those when a patient has no respiratory issue. Abnormal respiratory sounds happen in people with issues on respiratory tract or lungs. These abnormal sounds are excessively forced ordinary breath sounds. These sounds include wheezes and crackles.

Wheezes are sharp, regular and constant extrinsic audios having a patch with a minimum of 400Hz. They are usually caused by the narrowing of airway, which then causes an airflow limitation. Crackles are generated as a result of air bubbles in the large bronchi. They are heard in patients with chronic bronchitis, bronchiectasis as well as COPD.

Earlier times doctors depended on their hearing to distinguish these sounds. Then later stethoscope is considered an effective technique for examining the patients. It gives so much details about the respiratory organs and the indications of the sickness that influence it.

Auscultation by stethoscope is whimsical because it relies on the capacity of the doctor and human hearing. Traditional methods of disease detection are prone to human errors. Stethoscope have stethoscope bias. That is stethoscopes have very different characteristics. So it can lead to batch effect in the final dataset. For example, recording all sick samples using one stethoscope and healthy samples using another. Different data samples can be recorded in different rooms with different conditions using different instruments. This can also lead to batch effect in the final dataset.

Using manual stethoscope alone, many diseases might be misdiagnosed or undetected due to inability of hearing its corresponding respiratory sounds. To overcome this errors ,usage of modern instruments like Microphone, stethoscope, Electronic Stethoscope, and Meditron stethoscope together along with machine learning can be used. The electronic stethoscope together with pattern recognition and artificial intelligence helps in proper auscultation and is an effective technique in clinical conclusion. Electronic stethoscope has the ability to store lung sounds as signals within a computer, allowing medical doctors to investigate these signals in time-frequency analysis with a
better interpretation

In the field of bio informatics, respiratory sound classification has become the limelight. It is essential to classify the respiratory sounds abnormality in an authentic way to overcome death rate.

In this paper we are going to identify and elaborate how deep learning could be used in the recognition of respiratory disease from the respiratory sounds. We have approached the problem with a neural network model architecture and chose the model that would give us best possible results

The dataset is preprocessed to handle loss in data augmentation. The data is fed to a ICA model to separate the lung sounds from the heart sound. The data is then fed to conditional GAN to augmenting the data. The augmented data is use for training the model.

We have used evaluation metrics like Accuracy score, Precision score, Recall score, f1-score,, Cohen’s kappa score, Matthew correlation coefficient as metrics to evaluate and compare the performance of different models against the same dataset.

1.1. OBJECTIVE

Respiratory sounds helps us in uniquely identifying the respiratory disease. This project aims to deliver a model that recognizes the respiratory diseases based on the respiratory sounds using a deep neural network. The respiratory diseases are identified as either healthy, COPD, URTI, Bronchiectasis, Pneumonia, Bronchiolitis, Asthma, LRTI.

1.2. PROBLEM STATEMENT

- Respiratory diseases are difficult to identify using common medical procedures.
- We need to build a neural model that recognizes the state of the respiratory tract as healthy or diseased.
- The model should also recognize the respiratory diseases based on the respiratory sounds.

1.3. CHALLENGES IN THE SYSTEM

Dataset: The dataset for audio samples contains unevenly distributed data, which makes training the model difficult.
Dimension of audio data: The dimension of audio data is huge, so processing on huge volumes of audio data is a tedious task. Dimensionality reduction has to be performed

1.4. SCOPE OF THE PROJECT

The proposed method requires audio collection from subjects under controlled environments using devices with high resolution making it feasible only under research environments. With efficient noise masking techniques, it can be deployed in a real-world scenario to deal with external noises provided we need ample amount of data to feed the network for effective classification.

2. LITERATURE SURVEY

Basu et al [4] proposed a deep neural network architecture for effective classification of respiratory disease from respiratory sounds. They used traditional method of data augmentation by time shifting, length shifting of random noise. Though the architecture seems simple and effective, the data augmentation technique doesn’t seem to produce reliable results. Kochetov et al [8] proposed a noise masking recurrent neural network architecture which uses a hybrid RNN-LSTM model to mask noise from the respiratory audio's content using a lung sound localization technique. Acharya et al[1] built a CNN-RNN hybrid model using mel-spectrograms and also proposed patient specific tuning in classification. They used log-quantization to reduce the memory footprint to use in a wearable. But the model suffers from lack of sufficient patient specific data. Jayalakshmy et al[7] proposed a CGAN based classification model using EMD Scalograms and just 3 IMF features. The architecture used ResNet transfer learning model for classification but suffers from the high computational power required for training the model.

Mondal et al[17] 's paper proposes a new method to distinguish between the normal and the abnormal subjects using the morphological complexities of the lung sound signals. The morphological embedded complexities used in these experiments have been calculated in terms of texture information, irregularity index, third order moment, and fourth order moment. Pinho et al[13] proposed an algorithm for automatic detection of crackle based on 4 main procedures: i. recognition of a potential crackle; ii. verification of its validity; iii. characterisation of crackles parameters; and iv. optimisation of the algorithm parameters.

In Mendes et al [12]'s study, a multi-feature approach is proposed for the detection of events, in the frame space, that contain one or more crackles. The performance of thirty-five features was tested. Rocha et al [16]'s paper presented a new method for the discrimination of explosive cough events, which is based on a combination of spectral content descriptors and pitch-related features.

3. PROPOSED APPROACH

We propose a model that predicts the respiratory disease based on respiratory sounds. The dataset has audio samples containing crackles, wheezes and respiratory cycles. The samples are taken from different locations of the chest.

The samples are recorded under real life conditions which signify the presence of unwanted data in them.

Initially each data is done noise removal, then fed to an ICA model which separates the multivariate signal into its independent components.

After which the audio is sliced into frames of size 0.5 secs. Each such frame is again sliced into frames of size 0.05. Thus, for each audio sample 13*400 features are extracted. To reduce the dimensionality further, Principal Component Analysis is done. This is repeated for each and every audio sample in the dataset.

The dataset being sparse, Generative Adversarial Training is done to Augment the data. The GAN model holds a generator and a discriminator. The Generator takes a random point from the latent space and a class label, and outputs a generated image. The discriminator takes a image as input and outputs the probability of the image being real or not and the probability of the image belonging to one of the classes.

The InfoGAN model is then combined with a deep neural network model with 5 layers. The model built as such recognizes the respiratory diseases from respiratory sounds.

4. SYSTEM DESIGN

4.1. BLOCK DIAGRAM

The below diagram represents our system architecture. The ICBHI dataset is a huge corpora of audio samples collected from normal and diseased people. Preprocessing starts with first denoising the data and thenapplying Independent Component Analysis(Centering, Whitening, Normalizing) to separate individual sources in the multivariate signal

Secondly, the pre-processed audio samples are then frame-sliced. 13 MFCC features are extracted from each frame. At the end of this phase, each audio is identified to have 13*400 MFCC features and dimension of the audio output is then reduced using the Principal Component Analysis dimensionality reduction algorithm The generated feature sequence along with the one-hot encoded input label and latent vector is passed onto the generator of the InfoGan Model. Subsequently, discriminator model outputs the probability of the audio being real along with a predicted label.

These predictions are used as parameters to predict the loss of the generator, which enables to achieve better creation of samples by the generator. The loss is also fed to the auxillary model, which inturn improves the correctness of the created latent vectors.

Finally, the audio samples that are pre-processed and augmented are provided to the Deep Neural Network model. The results from the model are evaluated against the ground truth values using evaluation metrics like Accuracy, Recall, Precision, Cohen’s kappa score, F1 score and Matthews correlation coefficient.

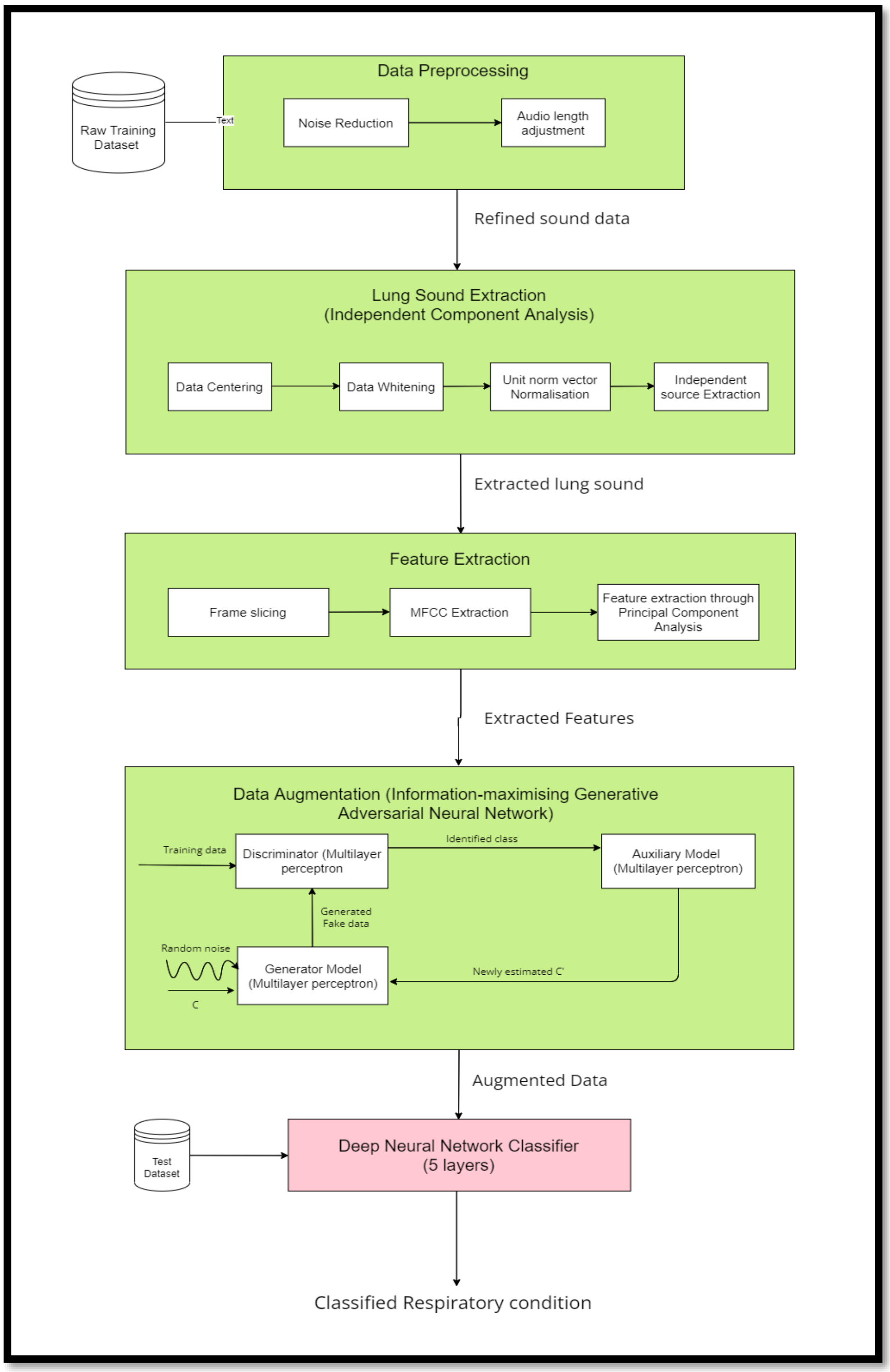

Data Preprocessing
Raw Training Dataset
Text
Noise Reduction
Audio length adjustment
Refined sound data
Lung Sound Extraction (Independent Component Analysis)
Data Centering
Data Whitening
Unit norm vector Normalisation
Independent source Extraction
Extracted lung sound
Feature Extraction
Frame slicing
MFCC Extraction
Feature extraction through Principal Component Analysis
Extracted Features
Data Augmentation (Information-maximising Generative Adversarial Neural Network)
Training data
Discriminator (Multilayer perceptron
Identified class
Auxiliary Model (Multilayer perceptron)
Generated Fake data
Random noise
C
Generator Model (Multilayer perceptron)
Newly estimated C'
Augmented Data
Test Dataset
Deep Neural Network Classifier (5 layers)
Classified Respiratory condition

Figure 4.1 Block Diagram

4.2. SYSTEM REQUIREMENTS

Python 3.8

Python Modules: Spicy, Librosa

RAM : Minimum of 4GB

5.DETAILED ARCHITECTURE

5.1 LIST OF MODULES:

The proposed model is split into 3 modules

1.Data pre-processing and Lung sound Extraction

2.Feature Extraction and Data Augmentation

3.Deep Neural Network Model

5.2 DATA PREPROCESSING AND LUNG SOUND EXTRACTION

MODULE INPUT:

The raw training dataset that is collected.

MODULE OUTPUT:

Extracted lung sounds free from heart sound intervention

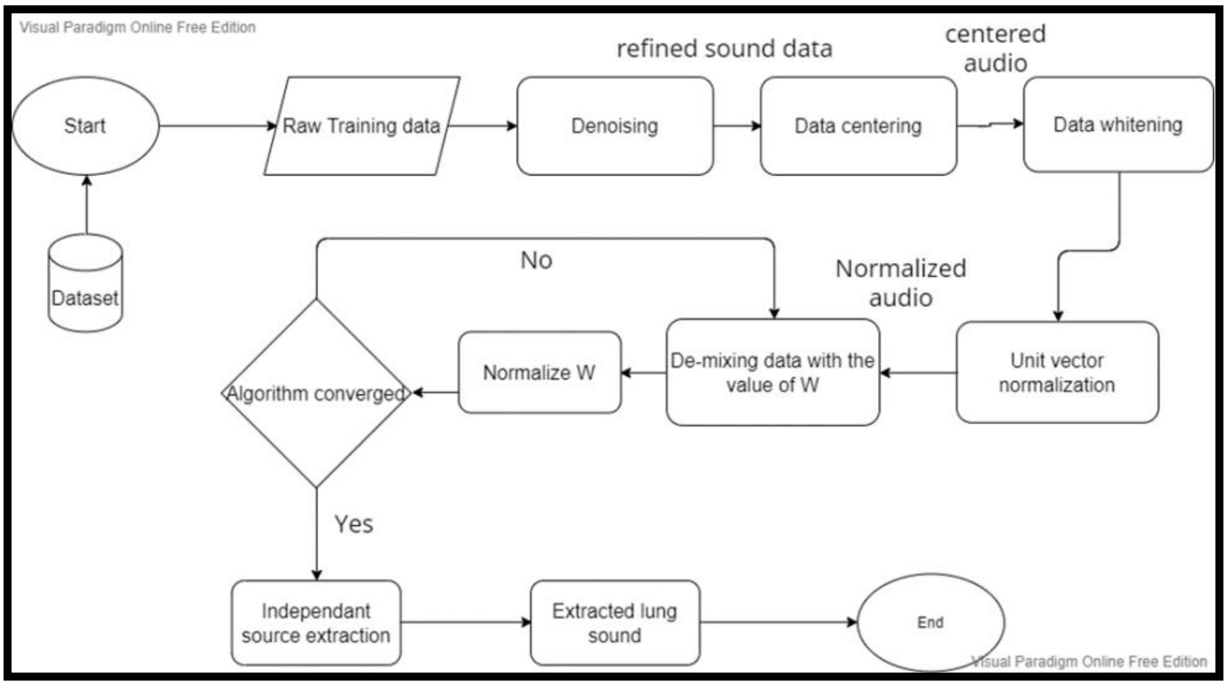


Figure 5.1 Data pre-processing and lung sound extraction

MODULE DESCRIPTION:

This module deals with the initial pre-processing of data. For feature extraction and other process in the model, the audio must be free from noise. So, the audio samples are first denoised. The main practical disadvantage of extracting lung sound is that, they are often overlapped by heart sounds. This may result in change in accuracy of the disease prediction. So, heart sound removal from lung sound is crucial. In this module, we use independent component analysis to remove heart sounds from lung sound. ICA separates the multivariate signals into individual underlying components. It involves multiple stages:

(i) Data centering:

Here the audio data is centered by subtracting the mean of the data. This helps to shift the scale.

(ii) Data whitening:

This involves the eigen value decomposition of the covariance matrix. Data whitening is used to remove correlation or dependencies between features in the data set. This helps to better train the model.

(iii) Unit norm vector normalization:

When features have different range, to change the value of those in the dataset to a common scale, without distorting difference in range, unit norm normalization is done. Here the matrices are divided by their respective magnitudes to get a normalized distribution of data.

(iv) De-mixing data with value w and normalize w:

A random variable for de-mixing matrix w is fixed. We update the value of w every time until convergence is reached. Convergence is said to be reached when the product of w and it's transpose is 1.

(v) Independent source extraction:

From the data that has been preprocessed, using independent source extraction we extract lungs sounds from the data. This extracted lung sound is then fed to the next module for feature extraction.

PSEUDO CODE

```
def  ICA(data : x)
    centered_x => center(x)
    whitened_x => whiten(centered_x)
    n_component => whitened_x.shape()
    w => np.zero(n_components)
    for i range(n_components)
        w_new => calculate(new_w)
        demixing(w, w_new)
        if(converged)
            break
        Result => np.dot(whitened_x, w)

for file in directory
    data_x, sampling_r => librosa.load(data_file)
    coeff => signal.firwin(sampling_r)
    filtered_x => lowfilter(coeff, data_x)
    row => np.zero(len(filtered_x))
    signal => ica(filtered_x)
    mfcc => librosa.mfcc(signal[0])
    save(mfcc)
    save(label)
```

5. 3 MODULE 2: FEATURE EXTRACTION AND DATA AUGMENTATION

5.3.1 SUBMODULE1: FEATURE EXTRACTION

MODULE INPUT:

A clear Lung sound (without noise and heart sound)

MODULE OUTPUT:

The MFCC features for all the audio samples.

MODULE DESCRIPTION:

For extracting mfcc features the audio must be clean without any noise. Thus, the sound file which is denoised and has heart sounds removed is fed to this module. Here, the library librosa is used for extracting mfcc features. It slices the audio frames and derives mfcc values for the sliced frames.These features are then integrated to produce the mfcc feature array of a given frame.

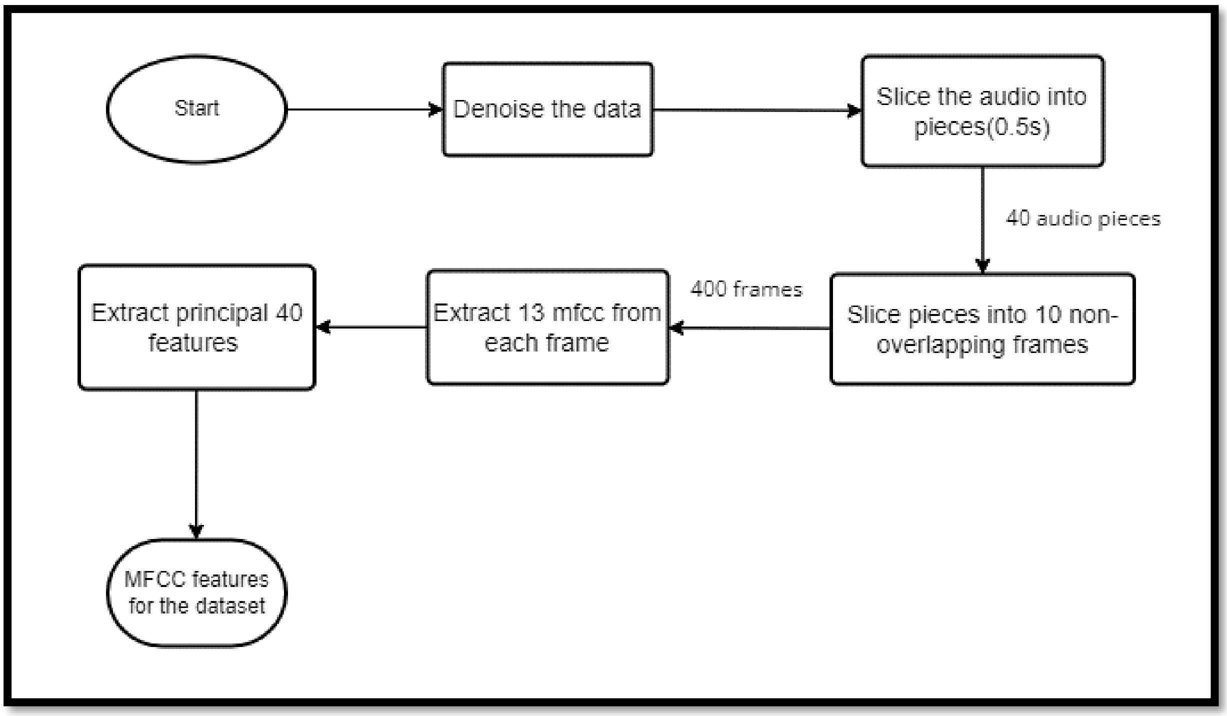

Start
Denoise the data
Slice the audio into pieces(0.5s)
40 audio pieces
Slice pieces into 10 non-overlapping frames
400 frames
Extract 13 mfcc from each frame
Extract principal 40 features
MFCC features for the dataset

Figure 5.2 feature extraction

5.3.2 SUB-MODULE2: DATA AUGMENTATION

This module covers the part of data augmentation in the dataset. Since the dataset isn't balanced(having an equal number of samples in all the classes), we have to add data to make it balanced. Traditional approaches like noise addition, time shifting was used for a long time. Here, we present an Information maximizing Generative Adversarial Neural Network's generator model to generate nearly real, fake audio samples from random noise.

MODEL INPUT:

The MFCC features of all the lung sounds in a dataset

MODEL OUTPUT:

Data to be augmented

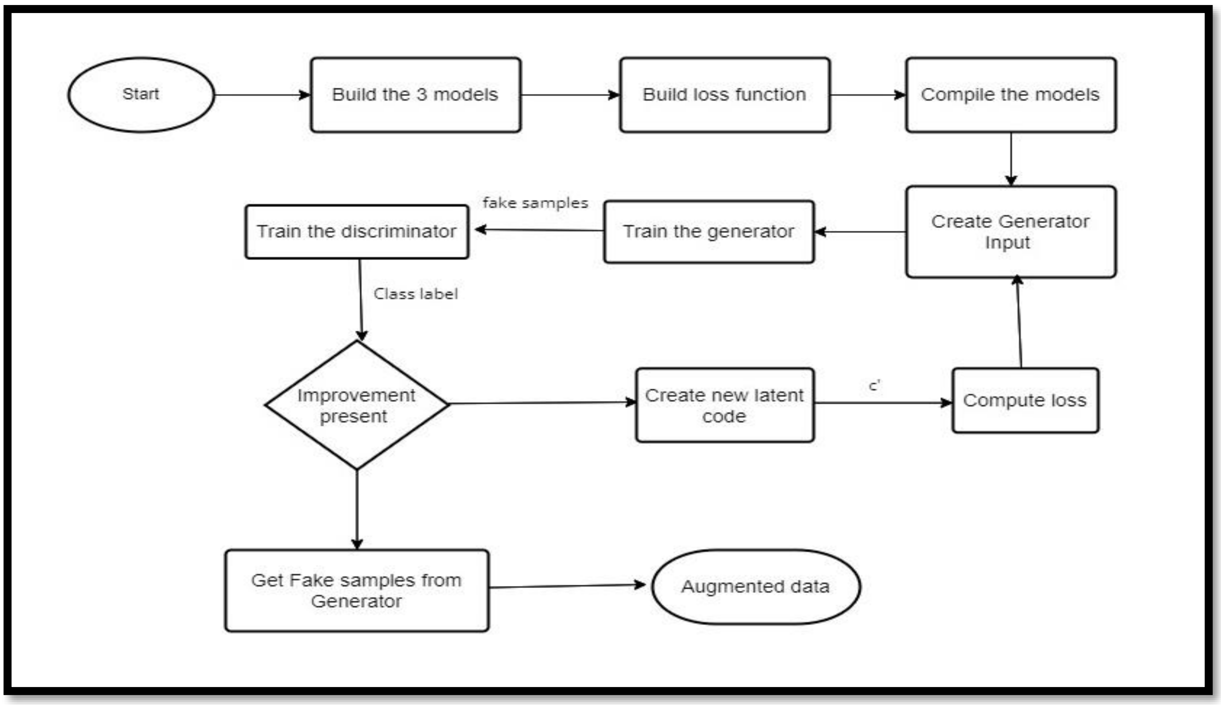


Figure 5.3 Data Augmentation

MODULE DESCRIPTION:

A GAN model acts like a minimax game between the generator and the discriminator where the generator always tries to maximise discriminator's loss and discriminator tries to reduce its noise. First, the generator and the discriminator models are built. Each model should be built in such a way that the number of layers in them is at least 4 and the number of neurons per layer can be between 64-512. Activation functions like relu, elu and leakyRelu are well suited for audio data. After the creation of models, a loss function is generated to compute the error rate of the discriminator in identifying the fake samples. Then, the models are compiled and the generator is trained on random noise. This generated input is fed to the discriminator model and it is trained. After training, the model evaluates its loss function. If the epoch number is less than the bound, then the process continues. Else the process halts.

5.4. MODULE 3: DEEP NEURAL NETWORK

MODEL INPUT:

The input for the classifier model is the augmented training data from previous module.

MODEL OUTPUT:

The output of the classifier phase is the trained model, which is then used to detect the lung disease when an audio sample is supplied.

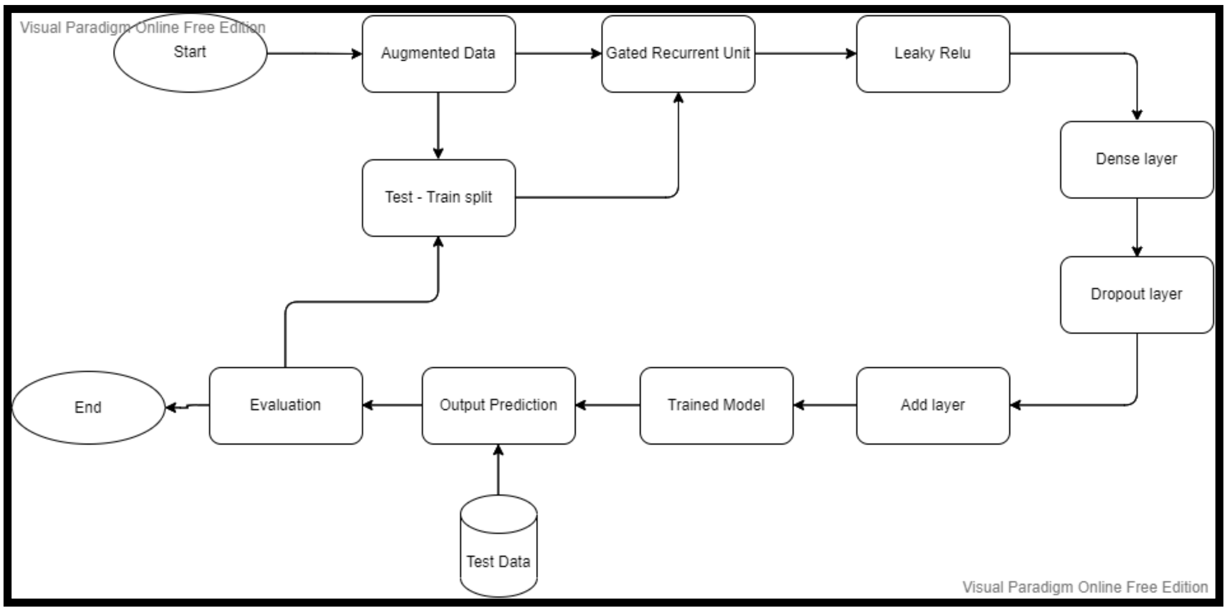


Figure 5.4 Deep Neural Network

MODULE DESCRIPTION:

This is the most important module of the model. The augmented data is split into train and test data. It gets augmented train data as input and produces the final classified output. We use a 5-layered deep neural network to classify the input as healthy, unhealthy and also returns the respiratory disease if unhealthy. The output of one layer is fed to the next layer. After classification it gets the test data as input and evaluates the model.

PSEUDO CODE

```
Def deepNeuralNetwork
        model => sequential ()
        model.add(GRU layer)
        model.add(Leaky Relu layer)
        model.add(Dense layer)
        model.add(dropout layer)
        model.add(add layer)
        model.fit(augmented data)

 def trainModel(X, y):
        Input_Sample = Input(X)
        Model_Enhancer = Model(inputs=Input_Sample, outputs=Output_)
        Model_Enhancer.compile(loss='categorical_crossentropy', metrics=['accuracy'], optim
izer=Adamax())
        ModelHistory = Model_Enhancer.fit(X_train,y_train)
        y_pred = Model_Enhancer.predict(X_test,y_test, batch_size = batch_size,
verbose = 1, callbacks = [MC])
        return y_pred, y_test, Model_Enhancer
```

6. IMPLEMENTATION

The classification model is built module-wise. First, the respiratory data is imported and the features are extracted following noise-reduction. Then, the lung sound is extracted using Independent Component Analysis.

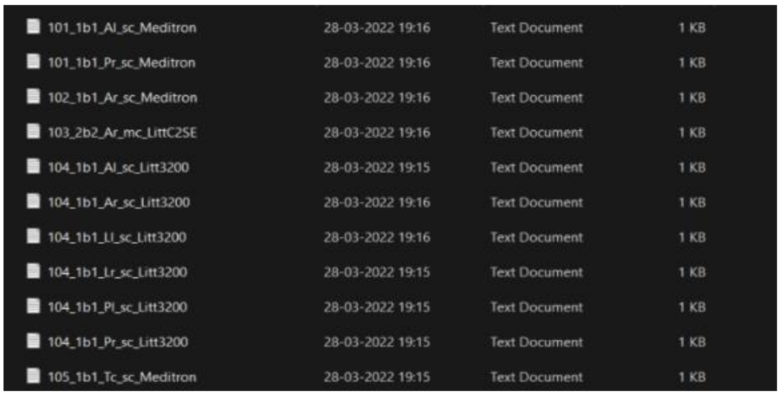
The resulting dataset is fed to a Generative Adversarial Network implementing Information-maximization to create audio samples resembling the original data. This network creates a single model that in turn creates class-wise data. This data is augmented with the original data and fed to the deep neural network model. This model classifies the audio sample into the respective respiratory disease identified.

6.1 DATASET

The Respiratory sound database was compiled for the scientific challenge organized at the International Conference on Biomedical Health Informatics – 2017, hence named ICBHI’17.

It contains audio samples collected independently by two research teams in two countries. It cinsists of a total of 5.5 hours of recordings containing 6898 respiratory cycles taken from 120 subjects.

DATASET



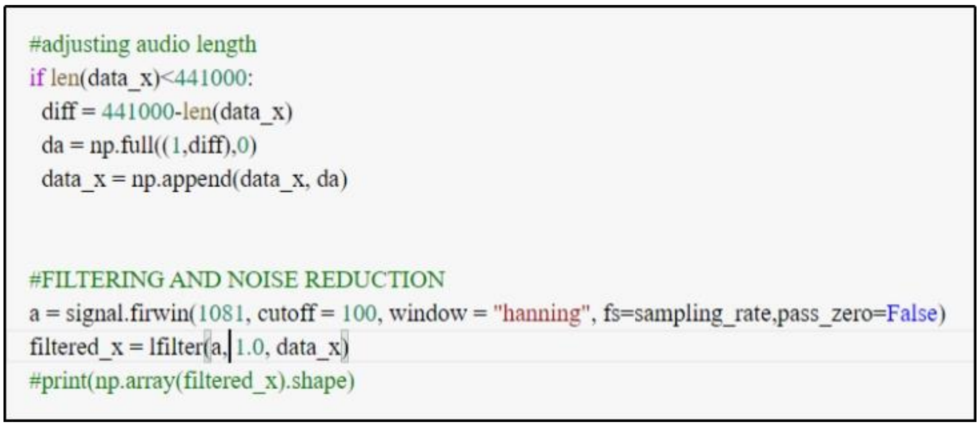
```
101_1b1_Al_sc_Meditron     28-03-2022 19:16     Text Document     1 KB
101_1b1_Pr_sc_Meditron     28-03-2022 19:16     Text Document     1 KB
102_1b1_Ar_sc_Meditron     28-03-2022 19:16     Text Document     1 KB
103_2b2_Ar_mc_LittC2SE     28-03-2022 19:16     Text Document     1 KB
104_1b1_Al_sc_Litt3200     28-03-2022 19:15     Text Document     1 KB
104_1b1_Ar_sc_Litt3200     28-03-2022 19:16     Text Document     1 KB
104_1b1_Ll_sc_Litt3200     28-03-2022 19:16     Text Document     1 KB
104_1b1_Lr_sc_Litt3200     28-03-2022 19:15     Text Document     1 KB
104_1b1_Pl_sc_Litt3200     28-03-2022 19:15     Text Document     1 KB
104_1b1_Pr_sc_Litt3200     28-03-2022 19:15     Text Document     1 KB
105_1b1_Tc_sc_Meditron     28-03-2022 19:15     Text Document     1 KB
```

Figure 6.1 The dataset

6.2 IMPLEMENTATION RESULTS

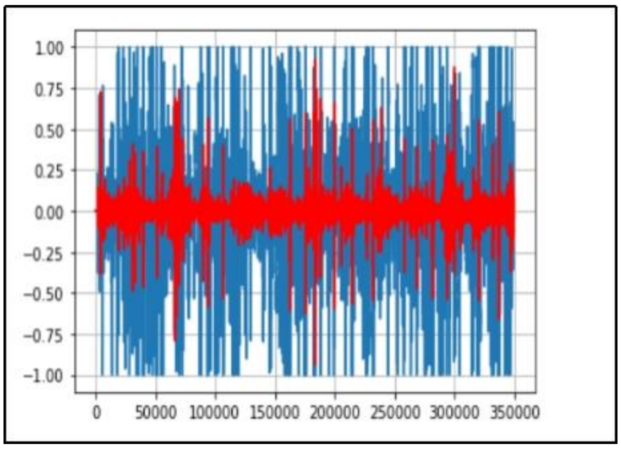
6.2.1 AUDIO LENGTH ADJUSTMENT AND NOISE REDUCTION

The length of the audio is adjusted to a standard of 20 seconds and noise reduction is performed using High Pass Finite Impulsive Response filter.



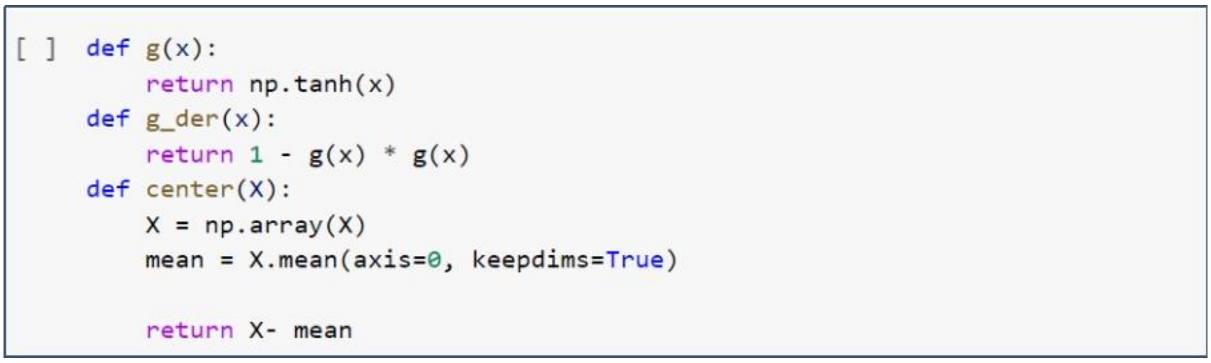
```
#adjusting audio length
if len(data_x)<441000:
  diff = 441000-len(data_x)
  da = np.full((1,diff),0)
  data_x = np.append(data_x, da)


#FILTERING AND NOISE REDUCTION
a = signal.firwin(1081, cutoff = 100, window = "hanning", fs=sampling_rate,pass_zero=False)
filtered_x = lfilter(a, 1.0, data_x)
#print(np.array(filtered_x).shape)
```

Sample output for Noise reduction

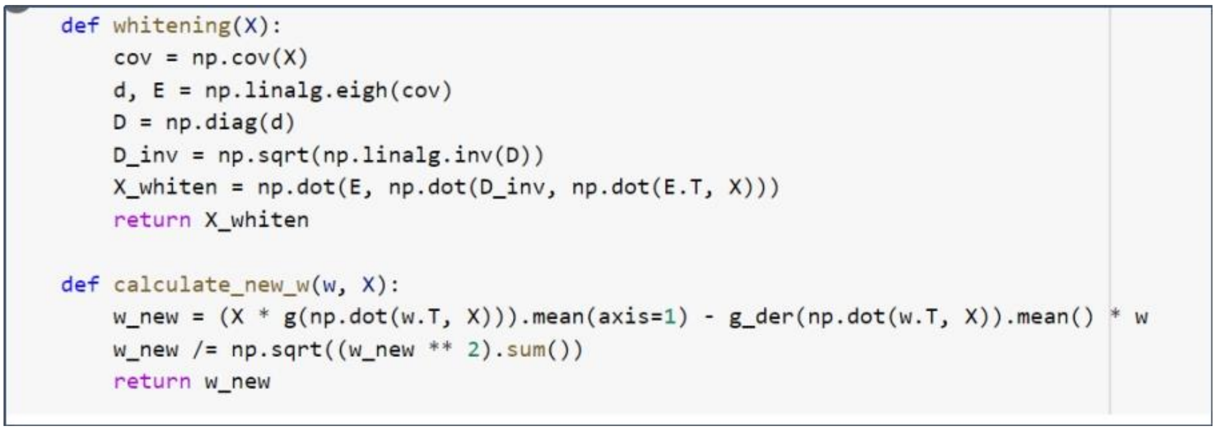


6.2.2 INDEPENDENT COMPONANT ANALYSIS

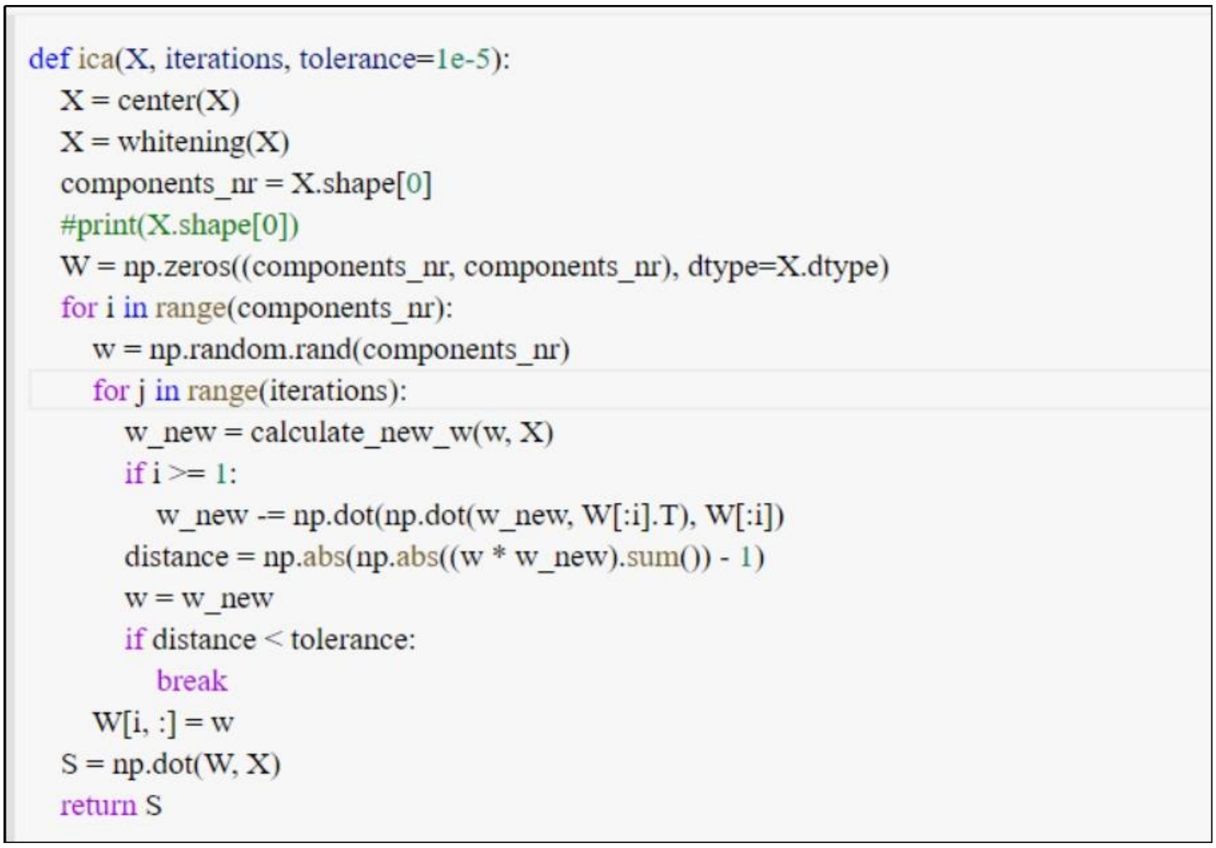
Independent Component Analysis technique is applied to the data to extract lung sound. The data is first centered, whitened and normalized following which the value for convergence is calculated to decide if a new iteration needs to begin.



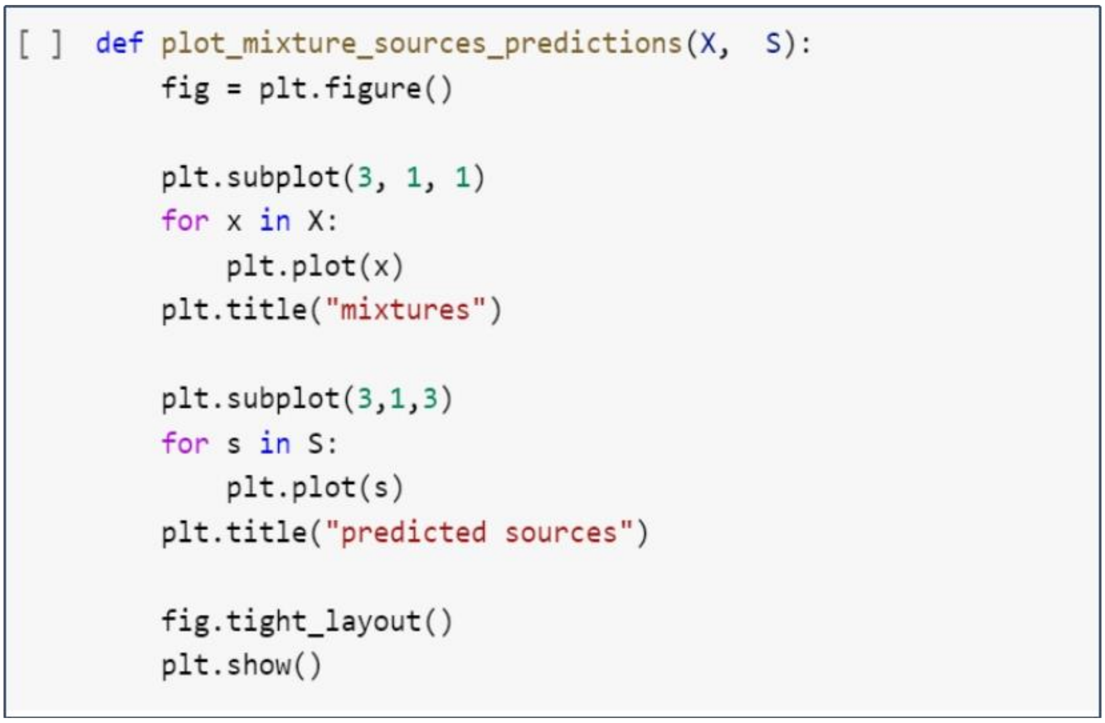
```
[ ]  def g(x):
         return np.tanh(x)
     def g_der(x):
         return 1 - g(x) * g(x)
     def center(X):
         X = np.array(X)
         mean = X.mean(axis=0, keepdims=True)

         return X- mean
```



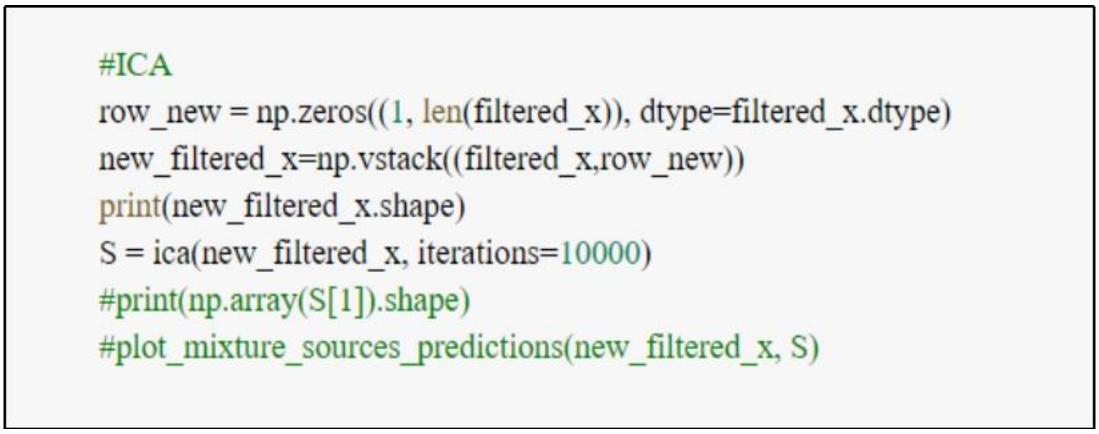
```
def whitening(X):
    cov = np.cov(X)
    d, E = np.linalg.eigh(cov)
    D = np.diag(d)
    D_inv = np.sqrt(np.linalg.inv(D))
    X_whiten = np.dot(E, np.dot(D_inv, np.dot(E.T, X)))
    return X_whiten

def calculate_new_w(w, X):
    w_new = (X * g(np.dot(w.T, X))).mean(axis=1) - g_der(np.dot(w.T, X)).mean() * w
    w_new /= np.sqrt((w_new ** 2).sum())
    return w_new
```



```
def ica(X, iterations, tolerance=1e-5):
    X = center(X)
    X = whitening(X)
    components_nr = X.shape[0]
    #print(X.shape[0])
    W = np.zeros((components_nr, components_nr), dtype=X.dtype)
    for i in range(components_nr):
        w = np.random.rand(components_nr)
        for j in range(iterations):
            w_new = calculate_new_w(w, X)
            if i >= 1:
                w_new -= np.dot(np.dot(w_new, W[:i].T), W[:i])
            distance = np.abs(np.abs((w * w_new).sum()) - 1)
            w = w_new
            if distance < tolerance:
                break
        W[i, :] = w
    S = np.dot(W, X)
    return S
```

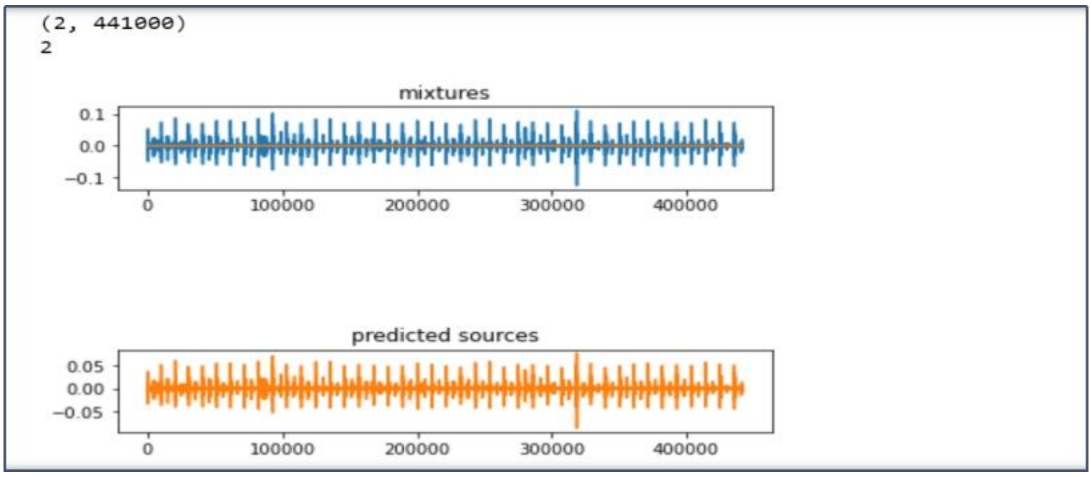


```
[ ]  def plot_mixture_sources_predictions(X,  S):
         fig = plt.figure()

         plt.subplot(3, 1, 1)
         for x in X:
             plt.plot(x)
         plt.title("mixtures")

         plt.subplot(3,1,3)
         for s in S:
             plt.plot(s)
         plt.title("predicted sources")

         fig.tight_layout()
         plt.show()
```

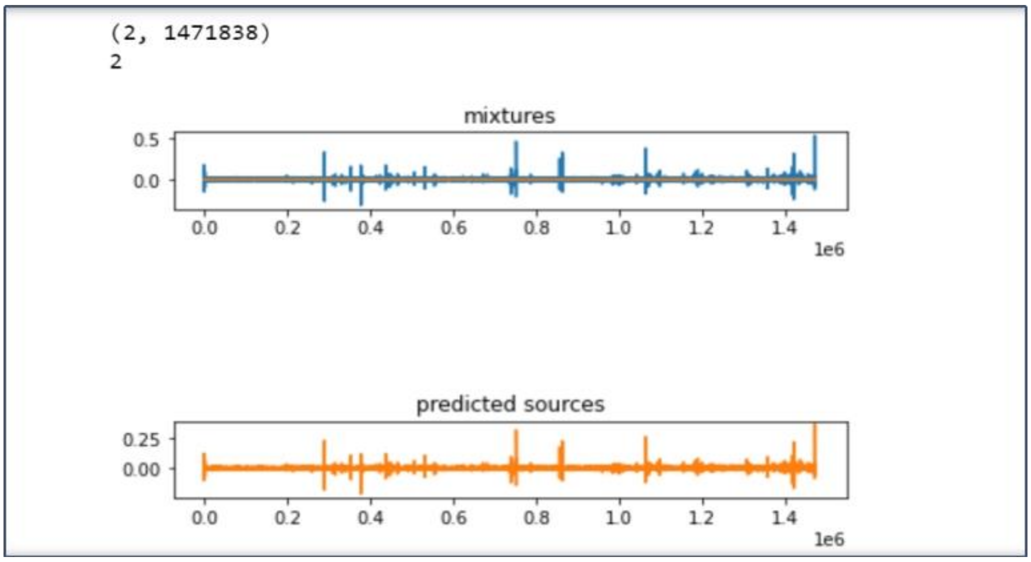


```
#ICA
row_new = np.zeros((1, len(filtered_x)), dtype=filtered_x.dtype)
new_filtered_x=np.vstack((filtered_x,row_new))
print(new_filtered_x.shape)
S = ica(new_filtered_x, iterations=10000)
#print(np.array(S[1]).shape)
#plot_mixture_sources_predictions(new_filtered_x, S)
```

Comparison of audio samples before and after ICA



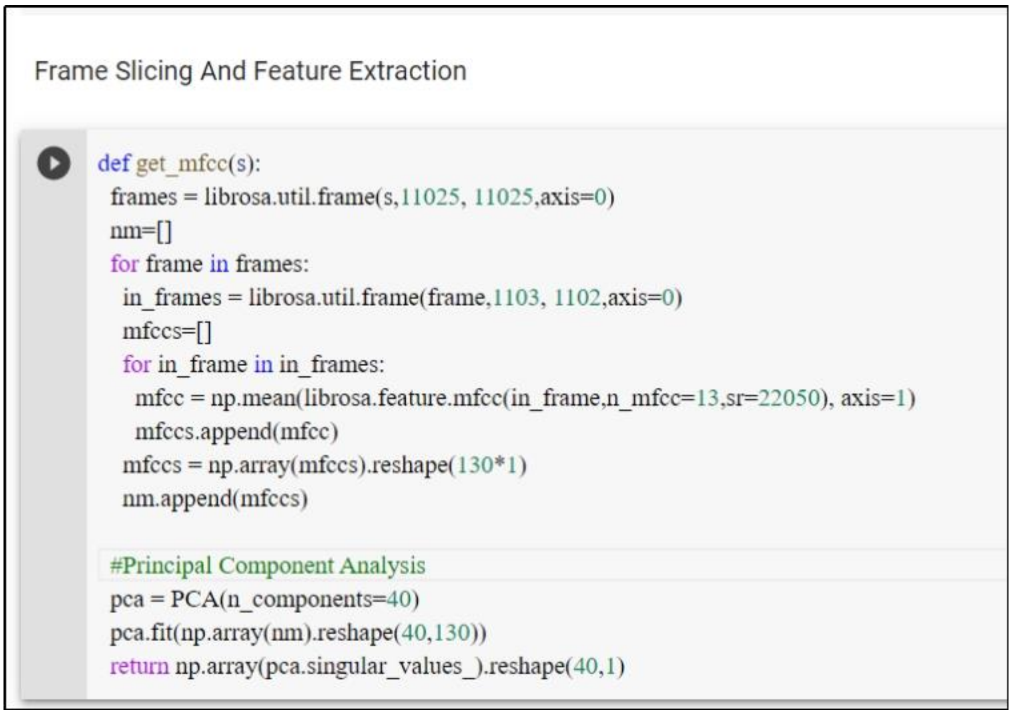
```
(2, 441000)
2
```



```
(2, 1471838)
2
```

6.2.3 FRAME SLICING AND FEATURE EXTRACTION

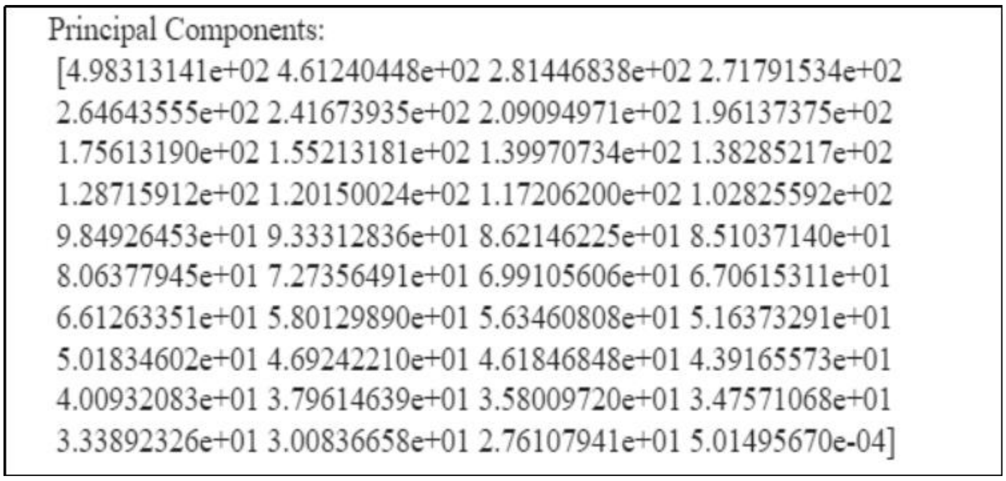
In order to target the respiratory cycles of each audio, they are sliced into pieces of length 0.5 seconds each. These pieces are then sliced into frames of length 0.05 seconds each and frame step as 0.05 seconds. Thus, a total of 400 frames are extracted from the audio. 13 Mel frequency cepstral coefficients are extracted from each frame, providing a total of 5200 mfcc features per sample. This is downsized by applying Principal Component Analysis to the data.



Frame Slicing And Feature Extraction

```
def get_mfcc(s):
  frames = librosa.util.frame(s,11025, 11025,axis=0)
  nm=[]
  for frame in frames:
    in_frames = librosa.util.frame(frame,1103, 1102,axis=0)
    mfccs=[]
    for in_frame in in_frames:
      mfcc = np.mean(librosa.feature.mfcc(in_frame,n_mfcc=13,sr=22050), axis=1)
      mfccs.append(mfcc)
    mfccs = np.array(mfccs).reshape(130*1)
    nm.append(mfccs)

  #Principal Component Analysis
  pca = PCA(n_components=40)
  pca.fit(np.array(nm).reshape(40,130))
  return np.array(pca.singular_values_).reshape(40,1)
```

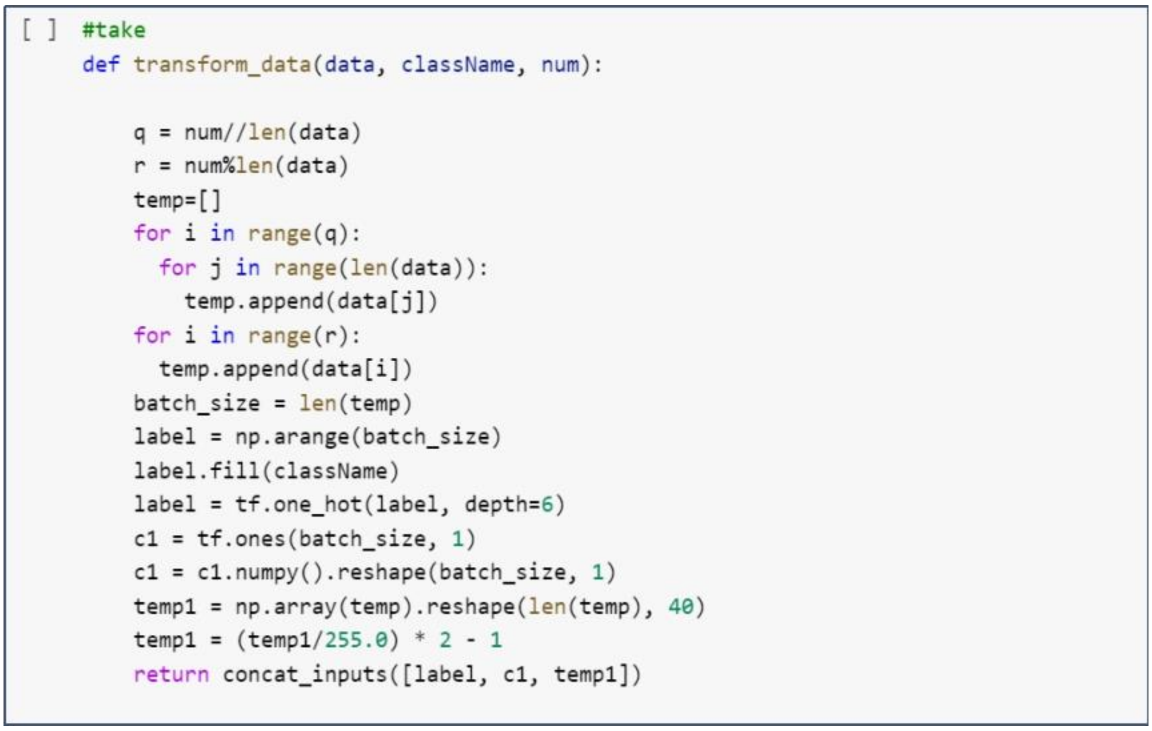
Sample Output



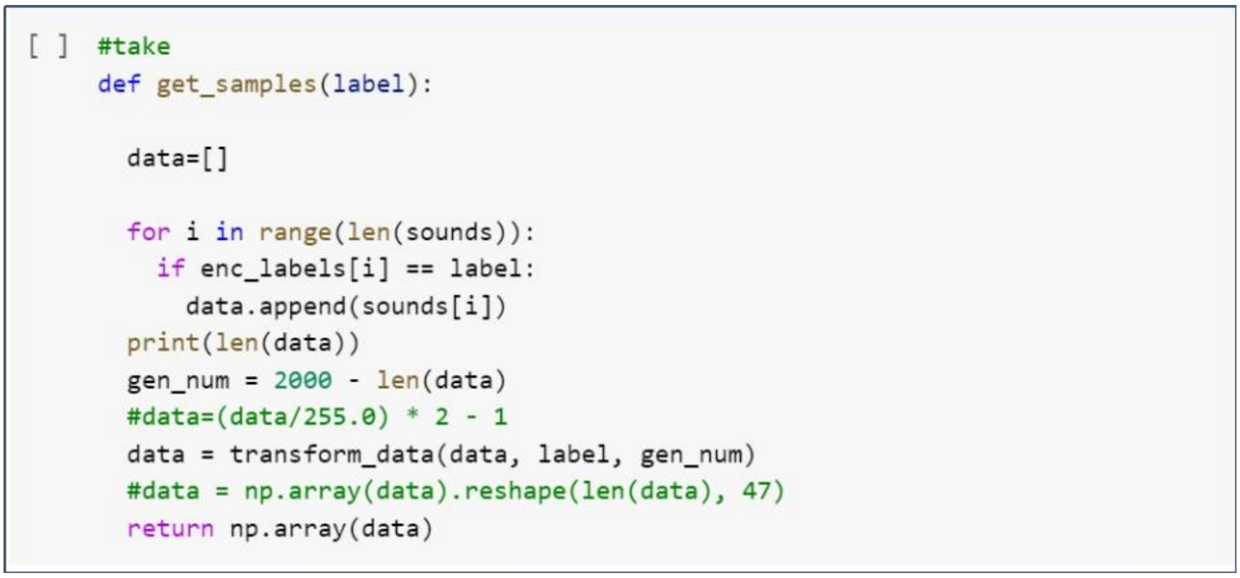
```
Principal Components:
[4.98313141e+02 4.61240448e+02 2.81446838e+02 2.71791534e+02
 2.64643555e+02 2.41673935e+02 2.09094971e+02 1.96137375e+02
 1.75613190e+02 1.55213181e+02 1.39970734e+02 1.38285217e+02
 1.28715912e+02 1.20150024e+02 1.17206200e+02 1.02825592e+02
 9.84926453e+01 9.33312836e+01 8.62146225e+01 8.51037140e+01
 8.06377945e+01 7.27356491e+01 6.99105606e+01 6.70615311e+01
 6.61263351e+01 5.80129890e+01 5.63460808e+01 5.16373291e+01
 5.01834602e+01 4.69242210e+01 4.61846848e+01 4.39165573e+01
 4.00932083e+01 3.79614639e+01 3.58009720e+01 3.47571068e+01
 3.33892326e+01 3.00836658e+01 2.76107941e+01 5.01495670e-04]
```

6.2.4 DATA AUGMENTATION USING INFOGAN

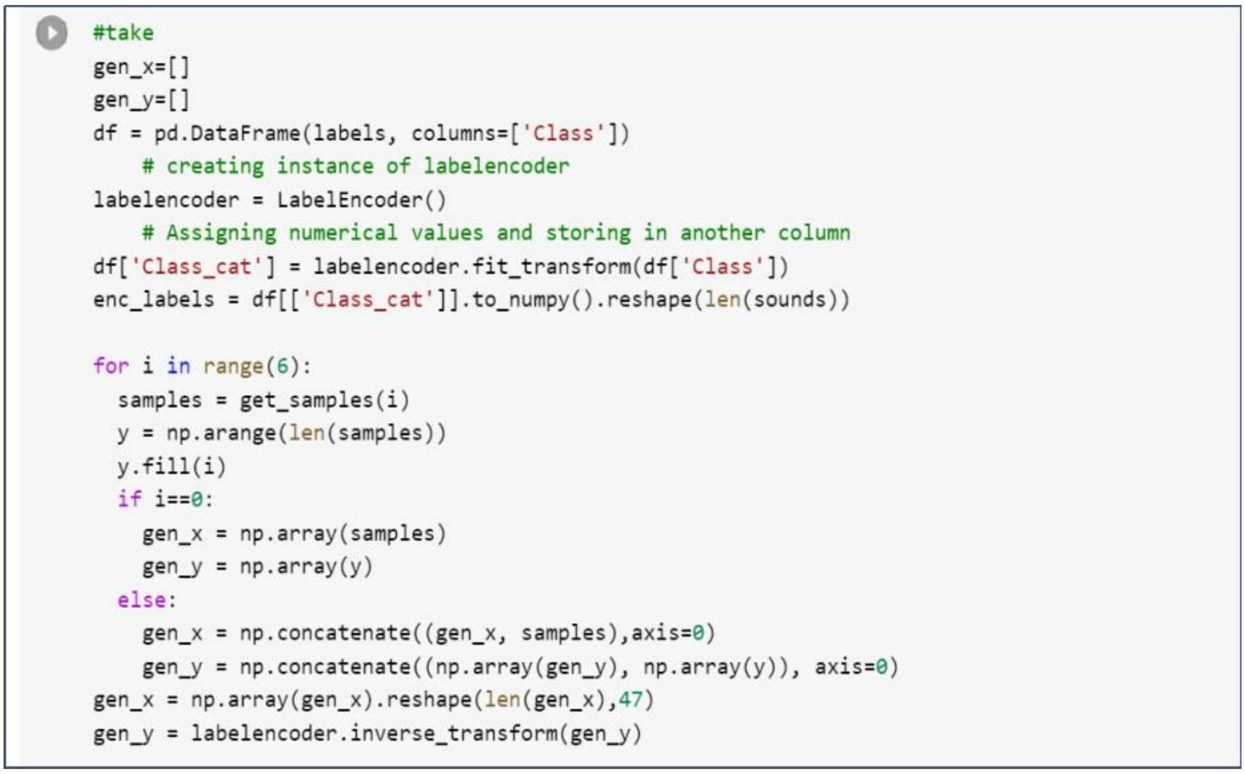
Here, the number of samples required for each class to get a balanced distribution is calculated. Then, samples from that class are transformed so as to be fed into an InfoGAN. These samples are then fed for the network and the augmenting samples are created by the generator network.



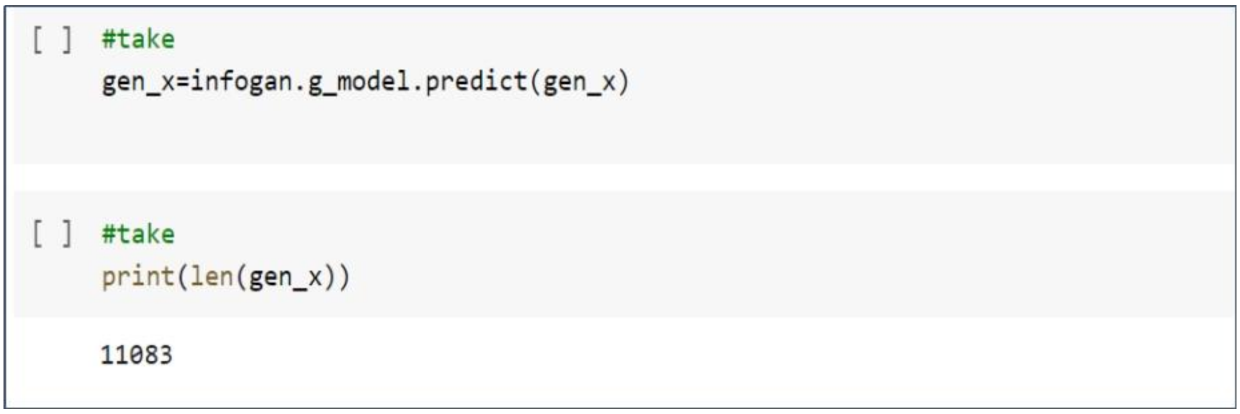
```
[ ] #take
    def transform_data(data, className, num):

        q = num//len(data)
        r = num%len(data)
        temp=[]
        for i in range(q):
          for j in range(len(data)):
            temp.append(data[j])
        for i in range(r):
          temp.append(data[i])
        batch_size = len(temp)
        label = np.arange(batch_size)
        label.fill(className)
        label = tf.one_hot(label, depth=6)
        c1 = tf.ones(batch_size, 1)
        c1 = c1.numpy().reshape(batch_size, 1)
        temp1 = np.array(temp).reshape(len(temp), 40)
        temp1 = (temp1/255.0) * 2 - 1
        return concat_inputs([label, c1, temp1])
```



```
[ ] #take
    def get_samples(label):

      data=[]

      for i in range(len(sounds)):
        if enc_labels[i] == label:
          data.append(sounds[i])
      print(len(data))
      gen_num = 2000 - len(data)
      #data=(data/255.0) * 2 - 1
      data = transform_data(data, label, gen_num)
      #data = np.array(data).reshape(len(data), 47)
      return np.array(data)
```

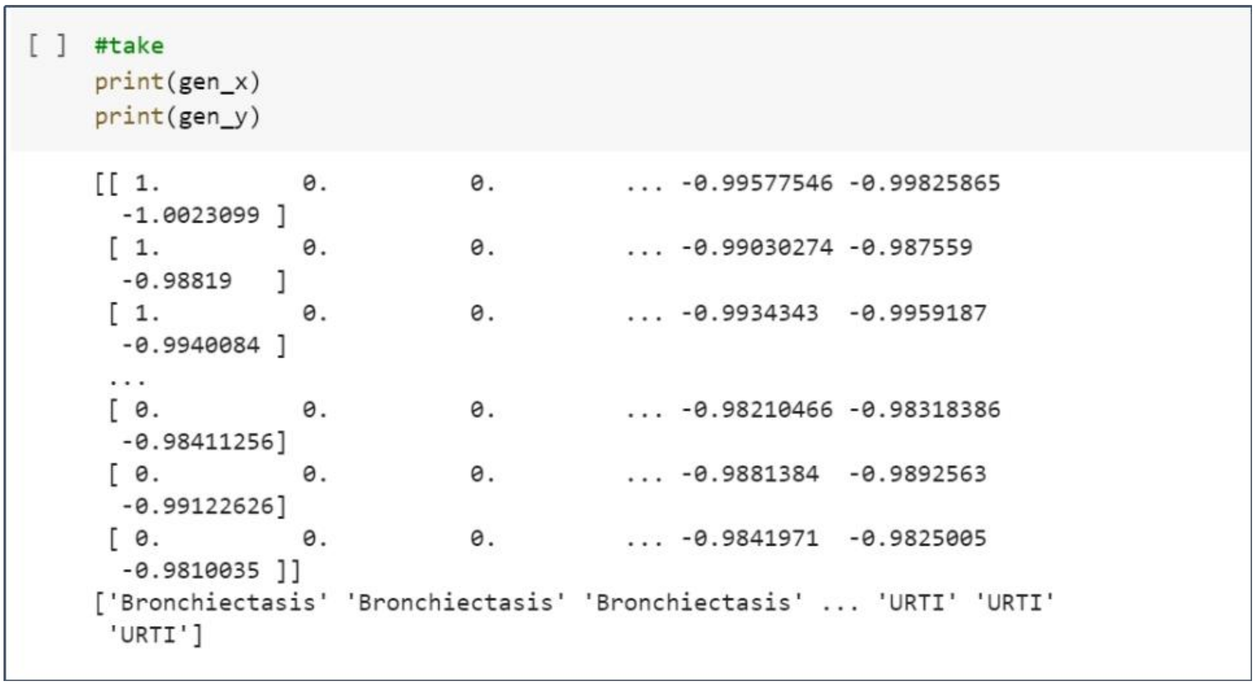


```
#take
gen_x=[]
gen_y=[]
df = pd.DataFrame(labels, columns=['Class'])
    # creating instance of labelencoder
labelencoder = LabelEncoder()
    # Assigning numerical values and storing in another column
df['Class_cat'] = labelencoder.fit_transform(df['Class'])
enc_labels = df[['Class_cat']].to_numpy().reshape(len(sounds))

for i in range(6):
  samples = get_samples(i)
  y = np.arange(len(samples))
  y.fill(i)
  if i==0:
    gen_x = np.array(samples)
    gen_y = np.array(y)
  else:
    gen_x = np.concatenate((gen_x, samples),axis=0)
    gen_y = np.concatenate((np.array(gen_y), np.array(y)), axis=0)
gen_x = np.array(gen_x).reshape(len(gen_x),47)
gen_y = labelencoder.inverse_transform(gen_y)
```



```
[ ] #take
    gen_x=infogan.g_model.predict(gen_x)
```

```
[ ] #take
    print(len(gen_x))

    11083
```

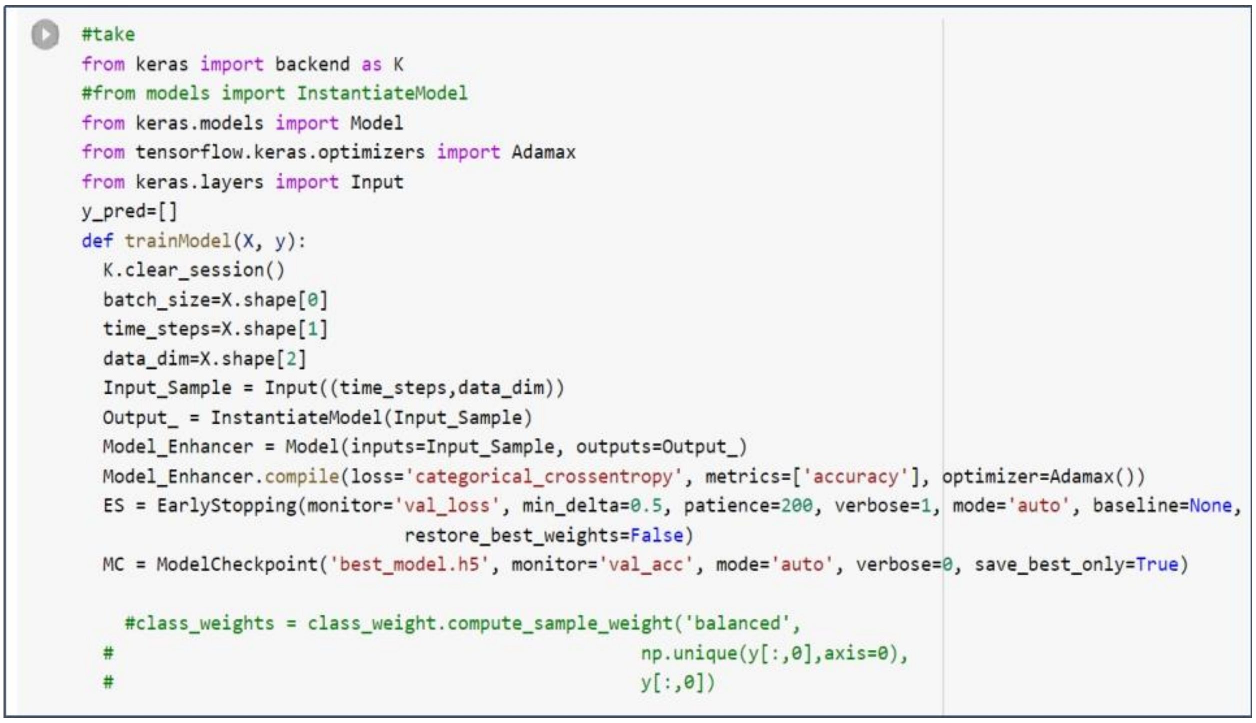


```
[ ]  #take
     print(gen_x)
     print(gen_y)

     [[ 1.          0.          0.         ... -0.99577546 -0.99825865
       -1.0023099 ]
      [ 1.          0.          0.         ... -0.99030274 -0.987559
       -0.98819    ]
      [ 1.          0.          0.         ... -0.9934343  -0.9959187
       -0.9940084 ]
      ...
      [ 0.          0.          0.         ... -0.98210466 -0.98318386
       -0.98411256]
      [ 0.          0.          0.         ... -0.9881384  -0.9892563
       -0.99122626]
      [ 0.          0.          0.         ... -0.9841971  -0.9825005
       -0.9810035 ]]
     ['Bronchiectasis' 'Bronchiectasis' 'Bronchiectasis' ... 'URTI' 'URTI'
      'URTI']
```

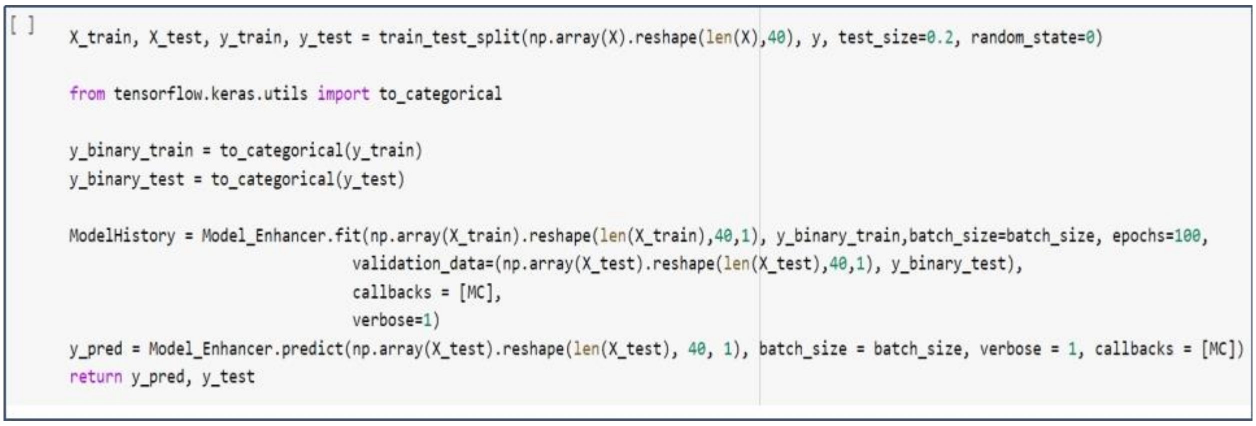
6.2.5 DEEP NEURAL NETWORK CLASSIFIER

The augmented data is fed into the deep neural network. The network is compiled using 5 layers.

Implementation of the neural network

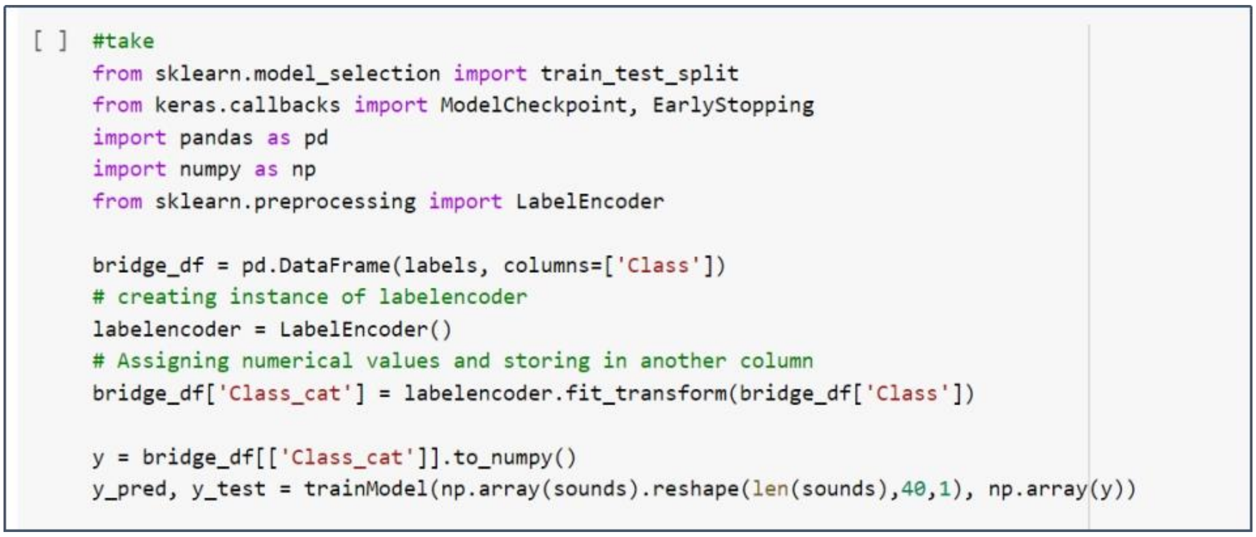


```
#take
from keras import backend as K
#from models import InstantiateModel
from keras.models import Model
from tensorflow.keras.optimizers import Adamax
from keras.layers import Input
y_pred=[]
def trainModel(X, y):
  K.clear_session()
  batch_size=X.shape[0]
  time_steps=X.shape[1]
  data_dim=X.shape[2]
  Input_Sample = Input((time_steps,data_dim))
  Output_ = InstantiateModel(Input_Sample)
  Model_Enhancer = Model(inputs=Input_Sample, outputs=Output_)
  Model_Enhancer.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=Adamax())
  ES = EarlyStopping(monitor='val_loss', min_delta=0.5, patience=200, verbose=1, mode='auto', baseline=None,
                             restore_best_weights=False)
  MC = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='auto', verbose=0, save_best_only=True)

    #class_weights = class_weight.compute_sample_weight('balanced',
  #                                                  np.unique(y[:,0],axis=0),
  #                                                  y[:,0])
```



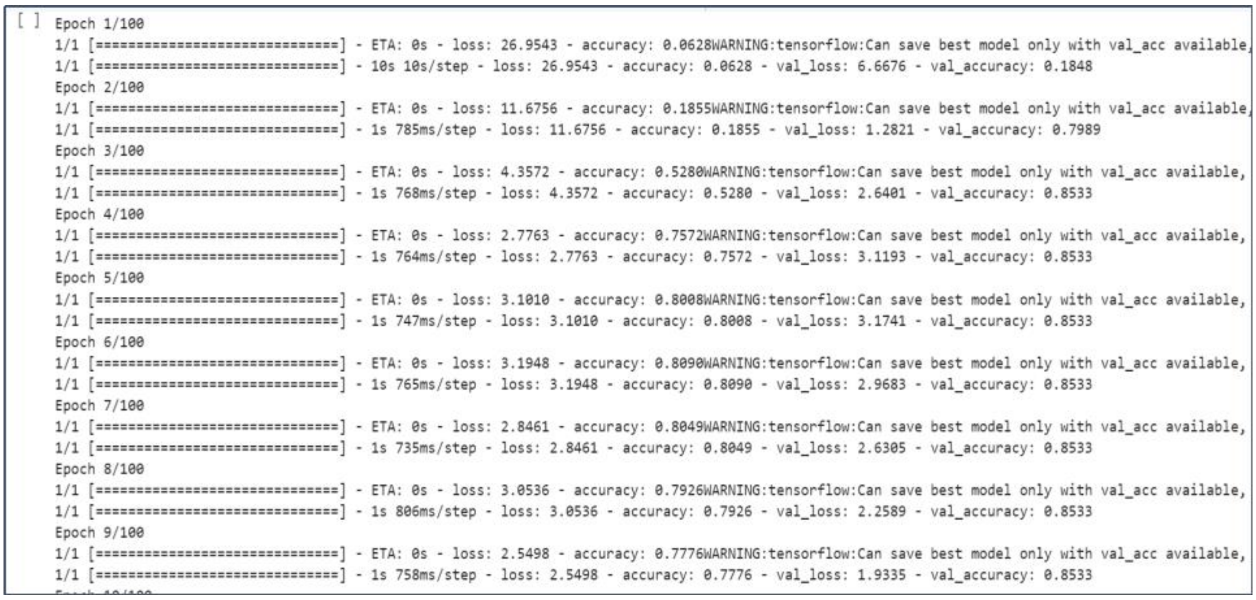
```
X_train, X_test, y_train, y_test = train_test_split(np.array(X).reshape(len(X),40), y, test_size=0.2, random_state=0)

from tensorflow.keras.utils import to_categorical

y_binary_train = to_categorical(y_train)
y_binary_test = to_categorical(y_test)

ModelHistory = Model_Enhancer.fit(np.array(X_train).reshape(len(X_train),40,1), y_binary_train,batch_size=batch_size, epochs=100,
                                  validation_data=(np.array(X_test).reshape(len(X_test),40,1), y_binary_test),
                                  callbacks = [MC],
                                  verbose=1)
y_pred = Model_Enhancer.predict(np.array(X_test).reshape(len(X_test), 40, 1), batch_size = batch_size, verbose = 1, callbacks = [MC])
return y_pred, y_test
```

Feeding into the network

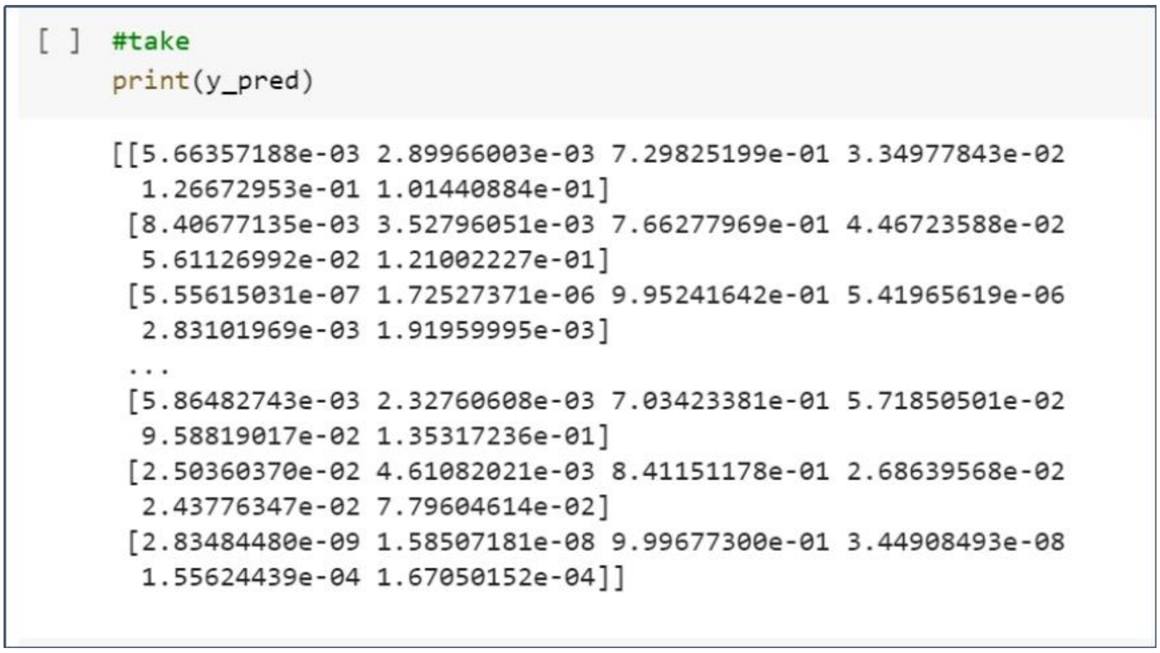


```
#take
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint, EarlyStopping
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder

bridge_df = pd.DataFrame(labels, columns=['Class'])
# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
bridge_df['Class_cat'] = labelencoder.fit_transform(bridge_df['Class'])

y = bridge_df[['Class_cat']].to_numpy()
y_pred, y_test = trainModel(np.array(sounds).reshape(len(sounds),40,1), np.array(y))
```

Received output:

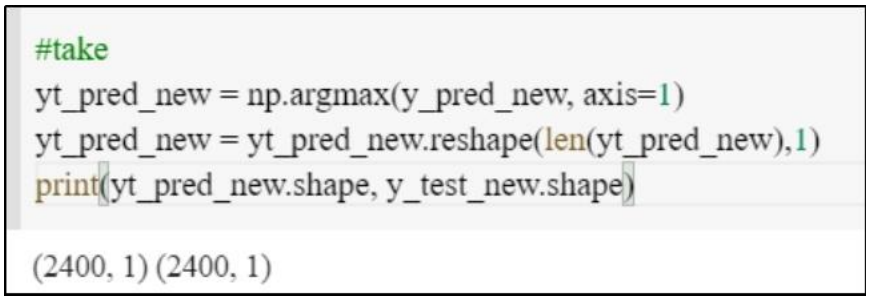


```
Epoch 1/100
1/1 [==============================] - ETA: 0s - loss: 26.9543 - accuracy: 0.0628WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 10s 10s/step - loss: 26.9543 - accuracy: 0.0628 - val_loss: 6.6676 - val_accuracy: 0.1848
Epoch 2/100
1/1 [==============================] - ETA: 0s - loss: 11.6756 - accuracy: 0.1855WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 785ms/step - loss: 11.6756 - accuracy: 0.1855 - val_loss: 1.2821 - val_accuracy: 0.7989
Epoch 3/100
1/1 [==============================] - ETA: 0s - loss: 4.3572 - accuracy: 0.5280WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 768ms/step - loss: 4.3572 - accuracy: 0.5280 - val_loss: 2.6401 - val_accuracy: 0.8533
Epoch 4/100
1/1 [==============================] - ETA: 0s - loss: 2.7763 - accuracy: 0.7572WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 764ms/step - loss: 2.7763 - accuracy: 0.7572 - val_loss: 3.1193 - val_accuracy: 0.8533
Epoch 5/100
1/1 [==============================] - ETA: 0s - loss: 3.1010 - accuracy: 0.8008WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 747ms/step - loss: 3.1010 - accuracy: 0.8008 - val_loss: 3.1741 - val_accuracy: 0.8533
Epoch 6/100
1/1 [==============================] - ETA: 0s - loss: 3.1948 - accuracy: 0.8090WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 765ms/step - loss: 3.1948 - accuracy: 0.8090 - val_loss: 2.9683 - val_accuracy: 0.8533
Epoch 7/100
1/1 [==============================] - ETA: 0s - loss: 2.8461 - accuracy: 0.8049WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 735ms/step - loss: 2.8461 - accuracy: 0.8049 - val_loss: 2.6305 - val_accuracy: 0.8533
Epoch 8/100
1/1 [==============================] - ETA: 0s - loss: 3.0536 - accuracy: 0.7926WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 806ms/step - loss: 3.0536 - accuracy: 0.7926 - val_loss: 2.2589 - val_accuracy: 0.8533
Epoch 9/100
1/1 [==============================] - ETA: 0s - loss: 2.5498 - accuracy: 0.7776WARNING:tensorflow:Can save best model only with val_acc available,
1/1 [==============================] - 1s 758ms/step - loss: 2.5498 - accuracy: 0.7776 - val_loss: 1.9335 - val_accuracy: 0.8533
```

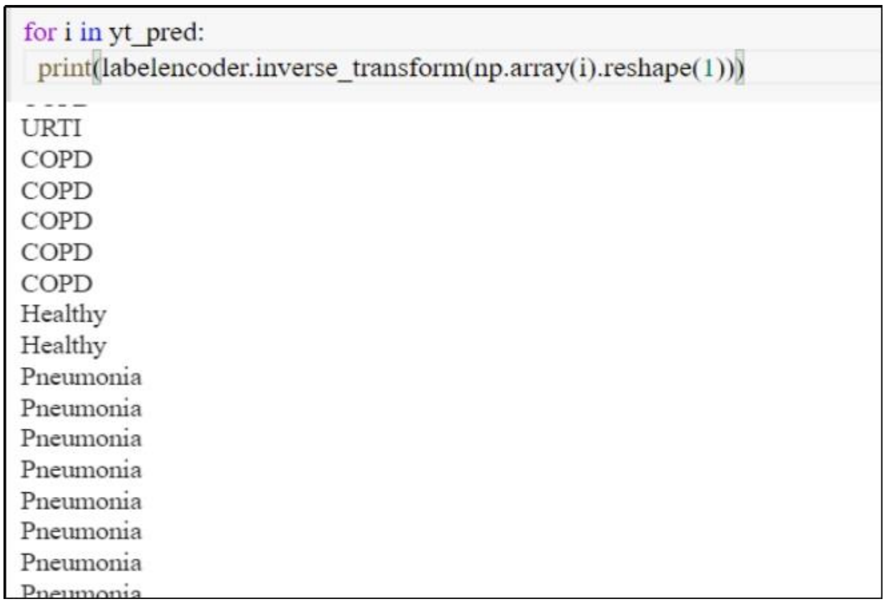


```
[ ]  #take
     print(y_pred)

     [[5.66357188e-03 2.89966003e-03 7.29825199e-01 3.34977843e-02
       1.26672953e-01 1.01440884e-01]
      [8.40677135e-03 3.52796051e-03 7.66277969e-01 4.46723588e-02
       5.61126992e-02 1.21002227e-01]
      [5.55615031e-07 1.72527371e-06 9.95241642e-01 5.41965619e-06
       2.83101969e-03 1.91959995e-03]
      ...
      [5.86482743e-03 2.32760608e-03 7.03423381e-01 5.71850501e-02
       9.58819017e-02 1.35317236e-01]
      [2.50360370e-02 4.61082021e-03 8.41151178e-01 2.68639568e-02
       2.43776347e-02 7.79604614e-02]
      [2.83484480e-09 1.58507181e-08 9.99677300e-01 3.44908493e-08
       1.55624439e-04 1.67050152e-04]]
```

Transformed output



```
#take
yt_pred_new = np.argmax(y_pred_new, axis=1)
yt_pred_new = yt_pred_new.reshape(len(yt_pred_new),1)
print(yt_pred_new.shape, y_test_new.shape)

(2400, 1) (2400, 1)
```



```
for i in yt_pred:
  print(labelencoder.inverse_transform(np.array(i).reshape(1)))
URTI
COPD
COPD
COPD
COPD
COPD
Healthy
Healthy
Pneumonia
Pneumonia
Pneumonia
Pneumonia
Pneumonia
Pneumonia
Pneumonia
Pneumonia
```

7. RESULT AND DISCUSSIONS

7.1 TEST SAMPLES

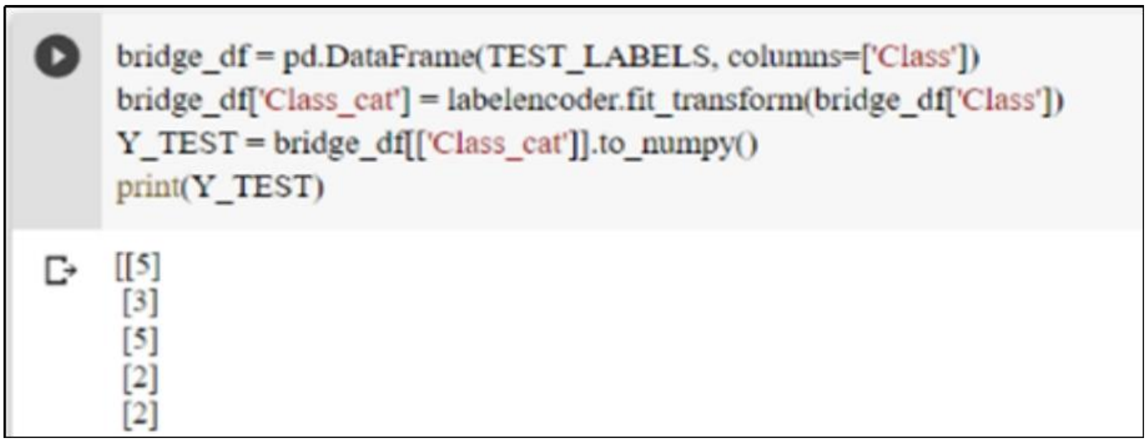
In order to cover test cases covering all possible demographics of the database, we chose test samples from each class representing each one of the seven recorded regions (Trachea (Tc), Anterior left (Al),Anterior right (Ar), Posterior left (Pl), Posterior right (Pr), Lateral left (Ll), Lateral right (Lr)) and recorded devices(AKGC417L Microphone (AKGC), 3M Littmann Classic II SE Stethoscope (LittC), 3M Litmmann 3200 Electronic Stethoscope (Litt3), WelchAllyn Meditron Master Elite Electronic Stethoscope (Med)) wherever possible.
[BRONCHIEC - Bronchiectasis, BRONCHIOL – Bronchiolitis]

	BRONCHIEC	**BRONCHIOL**	**COPD**	**HEALTHY**	**PNEUMONIA**	**URTI**
Tc	111_1b3_Tc_sc_Meditron.wav	NA	117_1b2_Tc_mc_LittC2SE.wav	121_1p1_Tc_sc_Meditron.wav	122_2b3_Tc_mc_LittC2SE.wav	105_1b1_Tc_sc_Meditron.wav
Al	168_1b1_Al_sc_Meditron.wav	173_1b1_Al_sc_Meditron.wav	113_1b1_Al_sc_Litt3200.wav	123_1b1_Al_sc_Meditron.wav	226_1b1_Al_sc_Meditron.wav	131_1b1_Al_sc_Meditron.wav
Ar	201_1b3_Ar_sc_Meditron.wav	206_1b1_Ar_sc_Meditron.wav	104_1b1_Ar_sc_Litt3200.wav	102_1b1_Ar_sc_Meditron.wav	219_2b2_Ar_mc_LittC2SE.wav	119_1b1_Ar_sc_Meditron.wav
Pl	116_1b2_Pl_sc_Meditron.wav	161_1b1_Pl_sc_Meditron.wav	107_2b4_Pl_mc_AKGC417L.wav	183_1b1_Pl_sc_Meditron.wav	191_2b1_Pl_mc_LittC2SE.wav	165_1b1_Pl_sc_Meditron.wav
Pr	196_1b1_Pr_sc_Meditron.wav	167_1b1_Pr_sc_Meditron.wav	106_2b1_Pr_mc_LittC2SE.wav	159_1b1_Pr_sc_Meditron.wav	191_2b1_Pr_mc_LittC2SE.wav	101_1b1_Pr_sc_Meditron.wav
Ll	169_1b2_Ll_s	NA	112_1p1_Ll_	187_1b1_Ll	140_2b3_Ll	137_1b1_

	c_Meditron.wav		sc_Litt3200.wav	_sc_Meditron.wav	_mc_LittC2SE.wav	Ll_sc_Meditron.wav
Lr	169_1b1_Lr_sc_Meditron.wav	149_1b1_Lr_sc_Meditron.wav	118_1b1_Lr_sc_Litt3200.wav	194_1b1_Lr_sc_Meditron.wav	NA	NA
AKGC	NA	NA	107_2b4_Pl_mc_AKGC417L.wav	NA	NA	NA
LITTC	NA	NA	106_2b1_Pr_mc_LittC2SE.wav	NA	All of the above	NA
LITT3	NA	NA	112_1p1_Ll_sc_Litt3200.wav	NA	NA	NA
MED	All of the above	All of the above	110_1p1_Lr_sc_Meditron.wav	All of the above	NA	All of the above

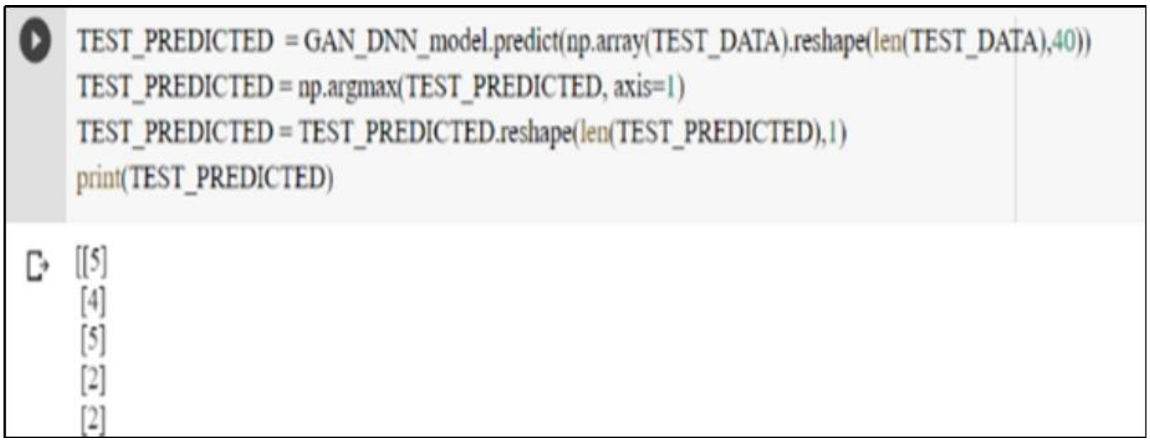
7.2 TEST CASES VALIDATION

The test set is fed to the deep neural network to get the predicted labels for the samples. Accordingly, the predicted labels are given below.



```
bridge_df = pd.DataFrame(TEST_LABELS, columns=['Class'])
bridge_df['Class_cat'] = labelencoder.fit_transform(bridge_df['Class'])
Y_TEST = bridge_df[['Class_cat']].to_numpy()
print(Y_TEST)
```

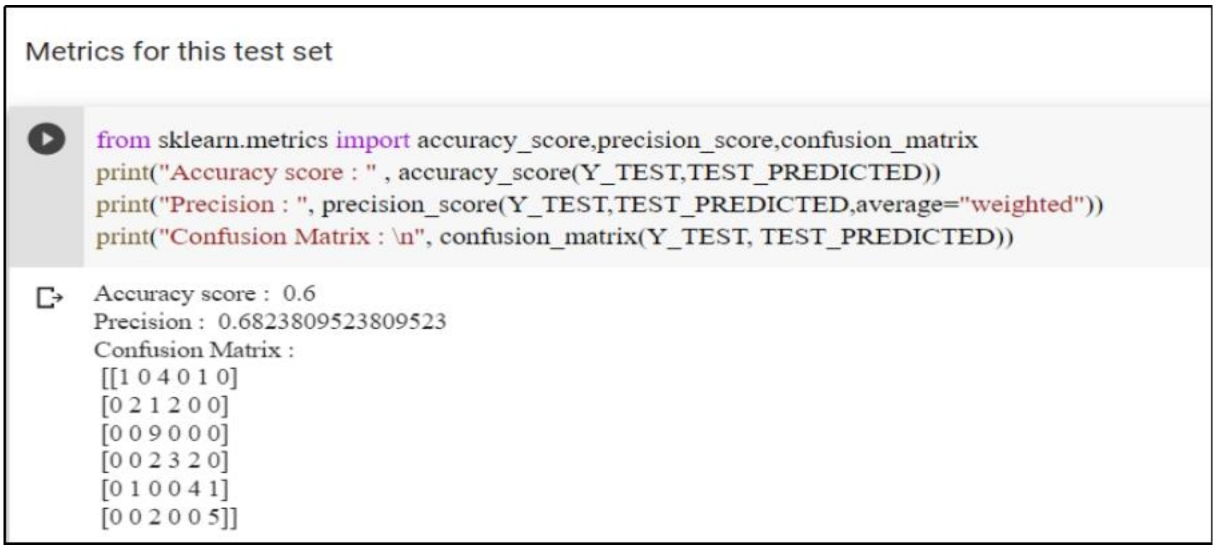
```
[[5]
 [3]
 [5]
 [2]
 [2]
```



```
TEST_PREDICTED  = GAN_DNN_model.predict(np.array(TEST_DATA).reshape(len(TEST_DATA),40))
TEST_PREDICTED = np.argmax(TEST_PREDICTED, axis=1)
TEST_PREDICTED = TEST_PREDICTED.reshape(len(TEST_PREDICTED),1)
print(TEST_PREDICTED)
```

```
[[5]
 [4]
 [5]
 [2]
 [2]
```

METRICS



Metrics for this test set

```
from sklearn.metrics import accuracy_score,precision_score,confusion_matrix
print("Accuracy score : " , accuracy_score(Y_TEST,TEST_PREDICTED))
print("Precision : ", precision_score(Y_TEST,TEST_PREDICTED,average="weighted"))
print("Confusion Matrix : \n", confusion_matrix(Y_TEST, TEST_PREDICTED))
```

```
Accuracy score :  0.6
Precision :  0.6823809523809523
Confusion Matrix :
 [[1 0 4 0 1 0]
 [0 2 1 2 0 0]
 [0 0 9 0 0 0]
 [0 0 2 3 2 0]
 [0 1 0 0 4 1]
 [0 0 2 0 0 5]]
```

PREDICTED LABELS

SOUND	ACTUAL LABEL	PREDICTED LABEL
102_1b1_Ar_sc_Meditron.wav	'Healthy'	'Pneumonia'
101_1b1_Pr_sc_Meditron.wav	'URTI'	'URTI'
104_1b1_Ar_sc_Litt3200.wav	'COPD'	'COPD'
107_2b4_Pl_mc_AKGC417L.wav	'COPD'	'COPD'
106_2b1_Pr_mc_LittC2SE.wav	'COPD'	'COPD'
169_1b1_Lr_sc_Meditron.wav	'Bronchiectasis'	'COPD'
118_1b1_Lr_sc_Litt3200.wav	'COPD'	'COPD'
119_1b1_Ar_sc_Meditron.wav	'URTI'	'URTI'
194_1b1_Lr_sc_Meditron.wav	'Healthy'	'Pneumonia'
165_1b1_Pl_sc_Meditron.wav	'URTI'	'URTI'
117_1b2_Tc_mc_LittC2SE.wav	'COPD'	'COPD'
113_1b1_Al_sc_Litt3200.wav	'COPD'	'COPD'
112_1p1_Ll_sc_Litt3200.wav	'COPD'	'COPD'
196_1b1_Pr_sc_Meditron.wav	'Bronchiectasis'	'COPD'
201_1b3_Ar_sc_Meditron.wav	'COPD'	'COPD'
110_1p1_Lr_sc_Meditron.wav	'Bronchiectasis'	'COPD'
116_1b2_Pl_sc_Meditron.wav	'COPD'	'COPD'
131_1b1_Al_sc_Meditron.wav	'URTI'	'COPD'
183_1b1_Pl_sc_Meditron.wav	'Healthy'	'COPD'
191_2b1_Pr_mc_LittC2SE.wav	'Pneumonia'	'Pneumonia'
159_1b1_Pr_sc_Meditron.wav	'Healthy'	'COPD'
137_1b1_Ll_sc_Meditron.wav	'URTI'	'COPD'
219_2b2_Ar_mc_LittC2SE.wav	'URTI'	'URTI'
167_1b1_Pr_sc_Meditron.wav	'Pneumonia'	'Pneumonia'
122_2b3_Tc_mc_LittC2SE.wav	'Bronchiolitis'	'Bronchiolitis'
226_1b1_Al_sc_Meditron.wav	'Pneumonia'	'Pneumonia'
191_2b1_Pl_mc_LittC2SE.wav	'Pneumonia'	'URTI'
206_1b1_Ar_sc_Meditron.wav	'Pneumonia'	'Bronchiolitis'
111_1b3_Tc_sc_Meditron.wav	'Bronchiolitis'	'Bronchiolitis'

121_1p1_Tc_sc_Meditron.wav	'Bronchiectasis'	'COPD'
123_1b1_Al_sc_Meditron.wav	'Healthy'	'Healthy'
168_1b1_Al_sc_Meditron.wav	'Healthy'	'Healthy'
169_1b2_Ll_sc_Meditron.wav	'Bronchiectasis'	'Pneumonia'
140_2b3_Ll_mc_LittC2SE.wav	'Bronchiectasis'	'Bronchiectasis'
161_1b1_Pl_sc_Meditron.wav	'Pneumonia'	'Pneumonia'
173_1b1_Al_sc_Meditron.wav	'Bronchiolitis'	'Healthy'
149_1b1_Lr_sc_Meditron.wav	'Bronchiolitis'	'COPD'
187_1b1_Ll_sc_Meditron.wav	'Bronchiolitis'	'Healthy'

7.3 COMPARATIVE ANALYSIS

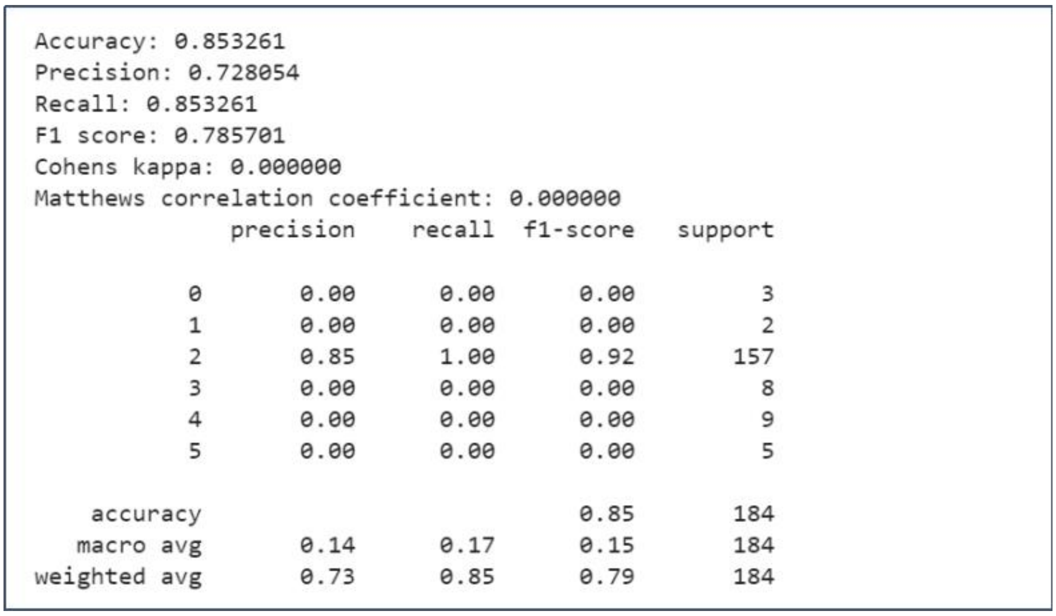
In order to assess the efficiency of the model, we use the metrics Accuracy, Precision, Recall, F1 score, Kohen's kappa score and Matthews correlation coefficient. The dataset suffers from imbalanced data distribution among classes. The initial method, however ensured data augmentation but has used random noise to generate the augmenting data. As a result, the metrics that reflects cooperation among classes is very low.

The present method using GAN, in spite of giving low accuracy as compared to the previous method, shows uniform distribution and equal cooperation between classes.

The comparatively higher accuracy results as a fact of imbalance in data. If a class is exceptionally more than the others, the model shows great accuracy even if it doesn't guess at least one sample of the other classes.

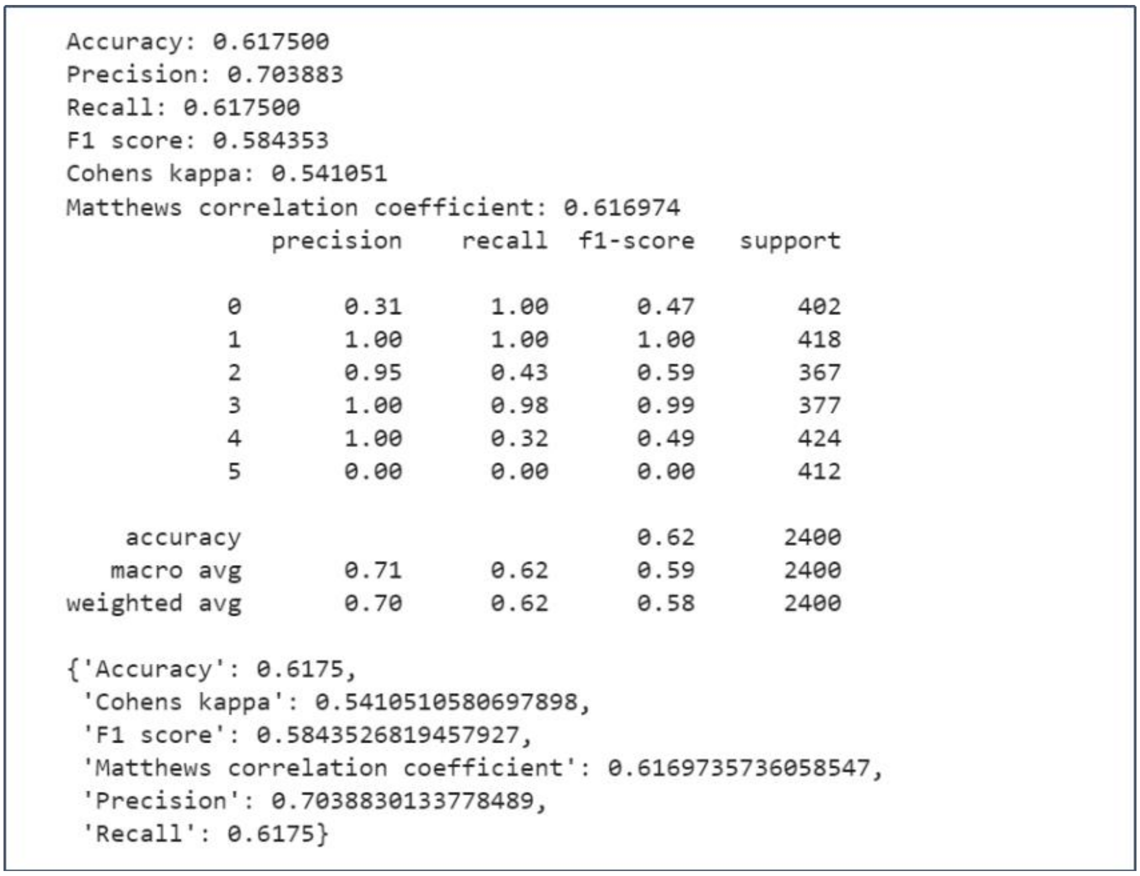
Moreover, the generated augmenting data resembles the domain space in case of GAN but shows no correlation to the domain space I the former method creating a lot of unwanted noise for the data. Thus, the proposed model gives clearer and efficient classification.

NAÏVE IMPLEMENTATION WITH ONLY DEEP NEURAL NETWORK



```
Accuracy: 0.853261
Precision: 0.728054
Recall: 0.853261
F1 score: 0.785701
Cohens kappa: 0.000000
Matthews correlation coefficient: 0.000000
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         2
           2       0.85      1.00      0.92       157
           3       0.00      0.00      0.00         8
           4       0.00      0.00      0.00         9
           5       0.00      0.00      0.00         5

    accuracy                           0.85       184
   macro avg       0.14      0.17      0.15       184
weighted avg       0.73      0.85      0.79       184
```

DEEP NEIRAL NETWORK WITH GAN



```
Accuracy: 0.617500
Precision: 0.703883
Recall: 0.617500
F1 score: 0.584353
Cohens kappa: 0.541051
Matthews correlation coefficient: 0.616974
              precision    recall  f1-score   support

           0       0.31      1.00      0.47       402
           1       1.00      1.00      1.00       418
           2       0.95      0.43      0.59       367
           3       1.00      0.98      0.99       377
           4       1.00      0.32      0.49       424
           5       0.00      0.00      0.00       412

    accuracy                           0.62      2400
   macro avg       0.71      0.62      0.59      2400
weighted avg       0.70      0.62      0.58      2400

{'Accuracy': 0.6175,
 'Cohens kappa': 0.5410510580697898,
 'F1 score': 0.5843526819457927,
 'Matthews correlation coefficient': 0.6169735736058547,
 'Precision': 0.7038830133778489,
 'Recall': 0.6175}
```

8. CONCLUSION AND FUTURE WORK

In this study, respiratory disease detection is carried over with respiratory sounds. We experimented noise reduction and lung sound extraction to improve the quality of data. Data augmentation was also performed using InfoGAN to create samples that maximize the mutual information between samples. Classification using Deep neural network was then performed to detect the respiratory disease. On evaluation, our improved model performed better in terms of representing correlation between classes and uniform distribution of classification.

8.1 FUTURE WORK

Classification metrics with regard to the domain of lung sounds are greatly affected by the interference of sounds from other internal organs. More efficient noise reduction techniques can be implemented to improve performance through this dimension. Noise masking techniques can be implemented to localize and identify respiratory cycles. Long Short-term memory units can be incorporated to record both spatial and temporal dependencies.

9. REFERENCES

[1] Acharya, J., & Basu, A. (2020). Deep neural network for respiratory sound classification in wearable devices enabled by patient specific model tuning. *IEEE transactions on biomedical circuits and systems*, *14*(3), 535-544.

[2] Ayari, F., Ksouri, M., & Alouani, A. T. (2012, March). Lung sound extraction from mixed lung and heart sounds FASTICA algorithm. In *2012 16th IEEE Mediterranean Electrotechnical Conference* (pp. 339-342). IEEE.

[3] Badnjevic, A., Gurbeta, L., & Custovic, E. (2018). An expert diagnostic system to automatically identify asthma and chronic obstructive pulmonary disease in clinical settings. *Scientific reports*, *8*(1), 1-9.

[4] Basu, V., & Rana, S. (2020, February). Respiratory diseases recognition through respiratory sound with the help of deep neural network. In *2020 4th International Conference on Computational Intelligence and Networks (CINE)* (pp. 1-6). IEEE.

[5] Chamberlain, D., Kodgule, R., Ganelin, D., Miglani, V., & Fletcher, R. R. (2016, August). Application of semi-supervised deep learning to lung sound analysis. In *2016 38th annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 804-807). IEEE.

[6] Guntupalli, K. K., Alapat, P. M., Bandi, V. D., & Kushnir, I. (2008). Validation of automatic wheeze detection in patients with obstructed airways and in healthy subjects. *Journal of asthma*, *45*(10), 903-907.

[7] Jayalakshmy, S., & Sudha, G. F. (2021, November). Respiratory signal Classification by cGAN augmented EMD-Scalograms. In *2021 IEEE 2nd International Conference on Applied Electromagnetics, Signal Processing, & Communication (AESPC)* (pp. 1-5). IEEE.

[8] Kochetov, K., Putin, E., Balashov, M., Filchenkov, A., & Shalyto, A. (2018, October). Noise masking recurrent neural network for respiratory sound classification. In *International Conference on Artificial Neural Networks* (pp. 208-217). Springer, Cham.

[9] Kochetov, K., Putin, E., Balashov, M., Filchenkov, A., & Shalyto, A. (2018, October). Noise masking recurrent neural network for respiratory sound classification. In *International Conference on Artificial Neural Networks* (pp. 208-217). Springer, Cham.

[10] Lartillot, O., & Toiviainen, P. (2007, September). A Matlab toolbox for musical feature extraction from audio. In *International conference on digital audio effects* (Vol. 237, p. 244).

[11] Mondal, A., Bhattacharya, P., & Saha, G. (2014). Detection of lungs status using morphological complexities of respiratory sounds. *The scientific world journal*, *2014*.

[12] Mendes, L., Vogiatzis, I. M., Perantoni, E., Kaimakamis, E., Chouvarda, I., Maglaveras, N., ... & Paiva, R. P. (2016, August). Detection of crackle events using a multi-feature approach. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (pp. 3679-3683). IEEE.

[13] Pinho, C., Oliveira, A., Jácome, C., Rodrigues, J. M., & Marques, A. (2016). Integrated approach for automatic crackle detection based on fractal dimension and box filtering. *International Journal of Reliable and Quality E-Healthcare (IJRQEH)*, *5*(4), 34-50.

[14] Ramalho, G. L. B., Rebouças Filho, P. P., Medeiros, F. N. S. D., & Cortez, P. C. (2014). Lung disease detection using feature extraction and extreme learning machine. *Revista Brasileira de Engenharia Biomédica*, *30*(3), 207-214.

[15] Rocha, B. M., Filos, D., Mendes, L., Vogiatzis, I., Perantoni, E., Kaimakamis, E., ... & Maglaveras, N. (2017, November). A respiratory sound database for the development of automated classification. In *International Conference on Biomedical and Health Informatics* (pp. 33-37). Springer, Singapore.

[16] Rocha, B. M., Mendes, L., Couceiro, R., Henriques, J., Carvalho, P., & Paiva, R. P. (2017, July). Detection of explosive cough events in audio recordings by internal sound analysis. In *2017 39th Annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 2761-2764). IEEE.

[17]Dataset link - https://bhichallenge.med.auth.gr/