Respiratory diseases recognition through respiratory sound with the help of deep neural network

1st Victor Basu
Department of Computer Science and Engineering
Jalpaiguri Government Engineering College
West Bengal, India
basu369victor@gmail.com

2nd Srinibas Rana,Assistant Professor
Department of Computer Science and Engineering
Jalpaiguri Government Engineering College
West Bengal, India
srinibas.rana123@gmail.com

***Abstract*—Prediction of respiratory diseases such as COPD(Chronic obstructive pulmonary disease), URTI(upper respiratory tract infection), Bronchiectasis, Pneumonia, Bronchiolitis with the help of deep neural networks or deep learning. We have constructed a deep neural network model that takes in respiratory sound as input and classifies the condition of its respiratory system. It not only classifies among the above-mentioned disease but also classifies if a person's respiratory system is healthy or not with higher accuracy and precision.**

***Index Terms*—Respiratory disease recognition, Deep neural network,GRU(Gated Recurrent Unit), sound data, data augmentation, feature extraction, classification**

I. Introduction

In this research paper, we are going to discuss how deep learning could be used in the recognition of respiratory disease just from the respiratory sound. Respiratory audios are important indicators of respiratory health and respiratory disorder. For example, a wheezing sound is a common sign that a patient has an obstructive airway disease like asthma or chronic obstructive pulmonary disease (COPD). We have approached the problem with different neural network model architecture, and choose the model with would give us the best possible results, we also performed data augmentation over the data-set. The data-set we have used consists of respiratory sounds taken from different patients from different locations around the chest. We have used Accuracy score, Precision score, Recall score, f1-score, Cohen's kappa score, Matthews correlation coefficient as metrics to evaluate and compare the performance of different models against the same data-set. With this model we have achieved an accuracy of $95.67\,\% \pm 0.77\,\%$,precision of $95.89\,\% \pm 0.8\,\%$,Sensitivity of $95.65\,\% \pm 0.753\,\%$,f1-score of $95.66\,\% \pm 0.79\,\%$, Cohen's kappa score of $94.74\,\% \pm 0.96\,\%$ and Matthews correlation coefficient of $94.79\,\% \pm 0.96\,\%$.

II. Model Implementation with the help of deep neural networks

A. *Why did we choose deep learning to approach the problem?*

Artificial Intelligence has always proven to solve any task or problem in the most efficient and enhanced way. We have used Deep learning which is a subpart of Artificial intelligence for the detection of respiratory diseases. A deep neural network model that would learn from the features extracted from the respiratory sound to classify the condition of a respiratory system. Using the A.I. model for the diagnosis would highly reduce the extract cost and time for manual diagnosis by doctors and pharmacists.

B. *About the Dataset*

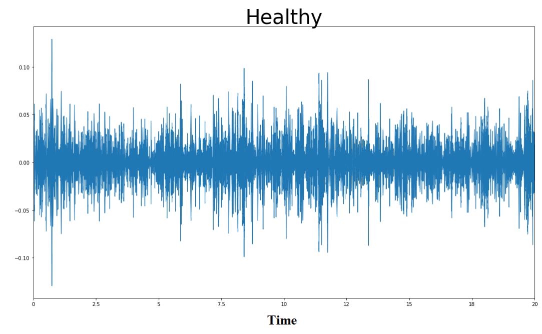
Details about the collector and maintainer of the database is explained in reference [1] and [11]. These recordings were taken from 126 patients. It includes 920 annotated recordings of varying length from 10s to 90s.There are a total of 5.5 hours of recordings containing 6898 respiratory cycles - 886 contain wheezes, 1864 contain crackles and 506 contain both crackles and wheezes. The patients span all age groups - children, adults and the elderly. The respiratory sounds in the data-set are of different category such as Healthy, COPD(Chronic obstructive pulmonary disease), URTI(upper respiratory tract infection), Bronchiectasis, Pneumonia, Bronchiolitis, Asthma, LRTI(Lower respiratory tract infection) which would be classified or predicted by out neural network model. Here healthy category means respiratory sound taken from a healthy person and other categories represents the sound taken from the patient suffering from their respective conditions. The data-set could be found at ICBHI 2017 Challenge[12].

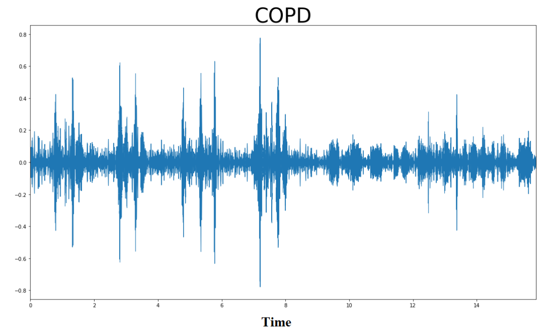
C. *Tools used for implementation*

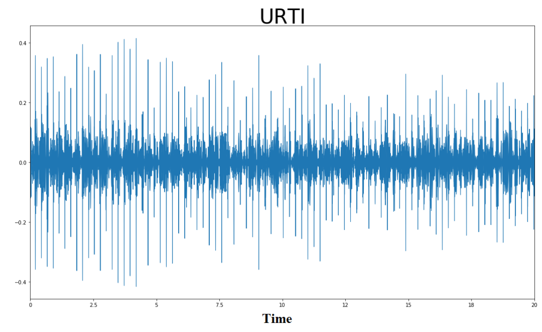
We have used python as the programming language to implement the model. For feature extraction from the audio data, we have used the library Librosa. For data visualization, we have used matplotlib library. For constructing the deep neural network we have used Keras, a python deep learning library with TensorFlow in the backend.

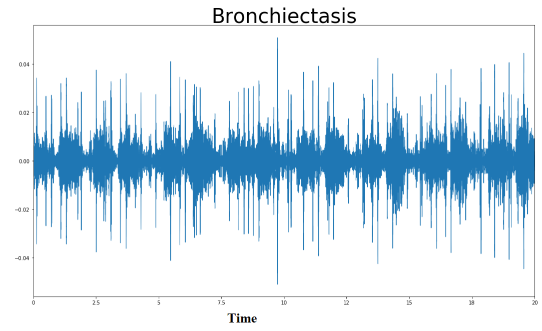
D. *Visualization of sound files of different category*

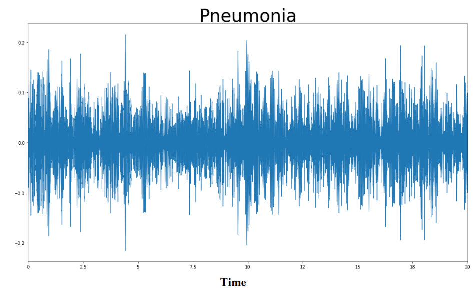
We have used librosa to load the audio files and displaying the wave-plot. Given below are the visualizations of the wave-plot of each category of the sound file we have used in our research.

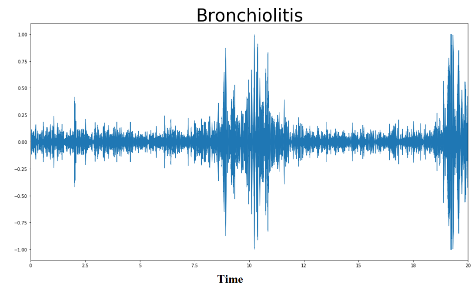












E. Data Pre-processing

The first problem we faced with the data-set was data imbalance. In the data-set, we got only one case for Asthma and only two cases for LRTI. Therefore we removed the rows corresponding to the respective classes for better performance of our model. Now, we are left with six classes.The data-set was still imbalanced. COPD had a much higher number of cases than other classes in the data-set, therefore we performed **data augmentation** with other classes in the data-set. In **data augmentation** we added random noise to the audio data, shifted the audio data both by using numpy library and also applied time-stretch using librosa to the audio data. In this data-set, the audio was recorded from different locations around the chest for every single patient so in case of COPD, we considered a maximum three locations for every single patient to balance the data-set.
We extracted **Mel-frequency cepstral coefficients(spectral features**), from the audio data. 40 features are extracted from each audio data and used to train the model.

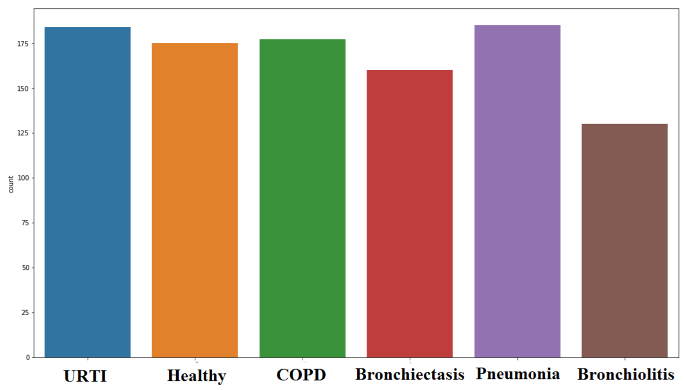


Fig. 1. Count plot after balancing imbalanced classes in the dataset

F. Implementation of neural network model

For construction of the neural network model we have used five types of layer 1) GRU(Gated Recurrent Unit), 2) Leaky Relu(Leaky version of a Rectified Linear Unit), 3) Dense Layer, 4) Dropout Layer, 5) Add Layer.

Function of each layer in our model in details.

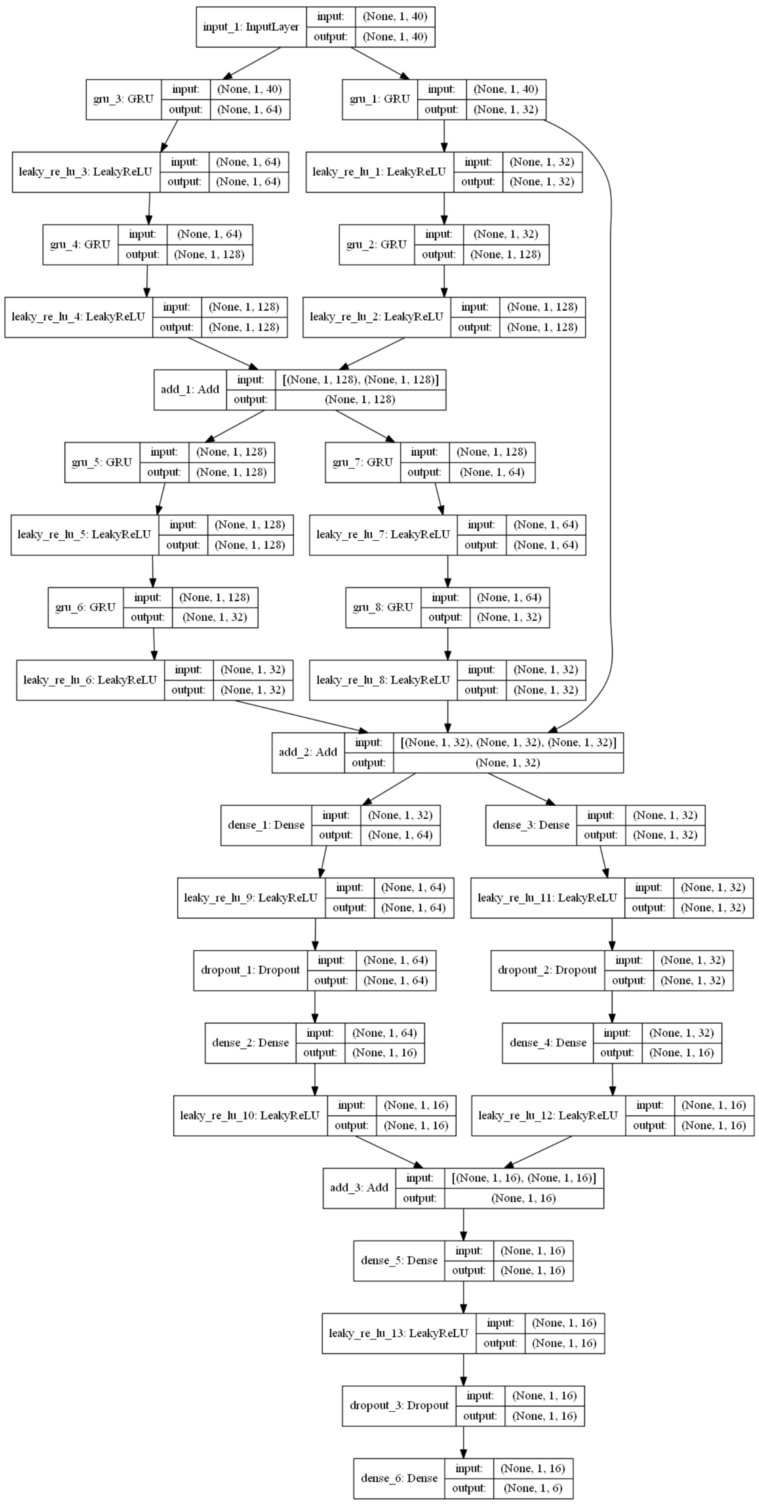

input_1: InputLayer | input: (None, 1, 40) | output: (None, 1, 40)
gru_3: GRU | input: (None, 1, 40) | output: (None, 1, 64)
gru_1: GRU | input: (None, 1, 40) | output: (None, 1, 32)
leaky_re_lu_3: LeakyReLU | input: (None, 1, 64) | output: (None, 1, 64)
leaky_re_lu_1: LeakyReLU | input: (None, 1, 32) | output: (None, 1, 32)
gru_4: GRU | input: (None, 1, 64) | output: (None, 1, 128)
gru_2: GRU | input: (None, 1, 32) | output: (None, 1, 128)
leaky_re_lu_4: LeakyReLU | input: (None, 1, 128) | output: (None, 1, 128)
leaky_re_lu_2: LeakyReLU | input: (None, 1, 128) | output: (None, 1, 128)
add_1: Add | input: [(None, 1, 128), (None, 1, 128)] | output: (None, 1, 128)
gru_5: GRU | input: (None, 1, 128) | output: (None, 1, 128)
gru_7: GRU | input: (None, 1, 128) | output: (None, 1, 64)
leaky_re_lu_5: LeakyReLU | input: (None, 1, 128) | output: (None, 1, 128)
leaky_re_lu_7: LeakyReLU | input: (None, 1, 64) | output: (None, 1, 64)
gru_6: GRU | input: (None, 1, 128) | output: (None, 1, 32)
gru_8: GRU | input: (None, 1, 64) | output: (None, 1, 32)
leaky_re_lu_6: LeakyReLU | input: (None, 1, 32) | output: (None, 1, 32)
leaky_re_lu_8: LeakyReLU | input: (None, 1, 32) | output: (None, 1, 32)
add_2: Add | input: [(None, 1, 32), (None, 1, 32), (None, 1, 32)] | output: (None, 1, 32)
dense_1: Dense | input: (None, 1, 32) | output: (None, 1, 64)
dense_3: Dense | input: (None, 1, 32) | output: (None, 1, 32)
leaky_re_lu_9: LeakyReLU | input: (None, 1, 64) | output: (None, 1, 64)
leaky_re_lu_11: LeakyReLU | input: (None, 1, 32) | output: (None, 1, 32)
dropout_1: Dropout | input: (None, 1, 64) | output: (None, 1, 64)
dropout_2: Dropout | input: (None, 1, 32) | output: (None, 1, 32)
dense_2: Dense | input: (None, 1, 64) | output: (None, 1, 16)
dense_4: Dense | input: (None, 1, 32) | output: (None, 1, 16)
leaky_re_lu_10: LeakyReLU | input: (None, 1, 16) | output: (None, 1, 16)
leaky_re_lu_12: LeakyReLU | input: (None, 1, 16) | output: (None, 1, 16)
add_3: Add | input: [(None, 1, 16), (None, 1, 16)] | output: (None, 1, 16)
dense_5: Dense | input: (None, 1, 16) | output: (None, 1, 16)
leaky_re_lu_13: LeakyReLU | input: (None, 1, 16) | output: (None, 1, 16)
dropout_3: Dropout | input: (None, 1, 16) | output: (None, 1, 16)
dense_6: Dense | input: (None, 1, 16) | output: (None, 1, 6)

Fig. 2. Architecture of the neural network model

- **Gated Recurrent Unit**(GRU)- It aims to solve the problem of vanishing gradient of a standard Recurrent neural layer. It has two types of gate - **update gate** and **reset gate**. These gates decide what information should be passed to the output. Update gate decides how much of the past memory is to need to be passed along to the future. Reset gate decides how much of past memory to forget. In our model, every GRU layer had linear activation function and the layers processed the input sequence backward and return the reverse sequence.
- **Leaky Relu**(Leaky version of a Rectified Linear Unit)- It is an activation layer. It allows a small gradient when the unit is active. In our model, we have a Leaky Relu layer after every GRU and Dense layer.
- **Dense Layer** - It is a fully connected layer. In our model the lower half of the neural network comprises of only dense layers.
- **Dropout** - By adding this layer some of the layer outputs

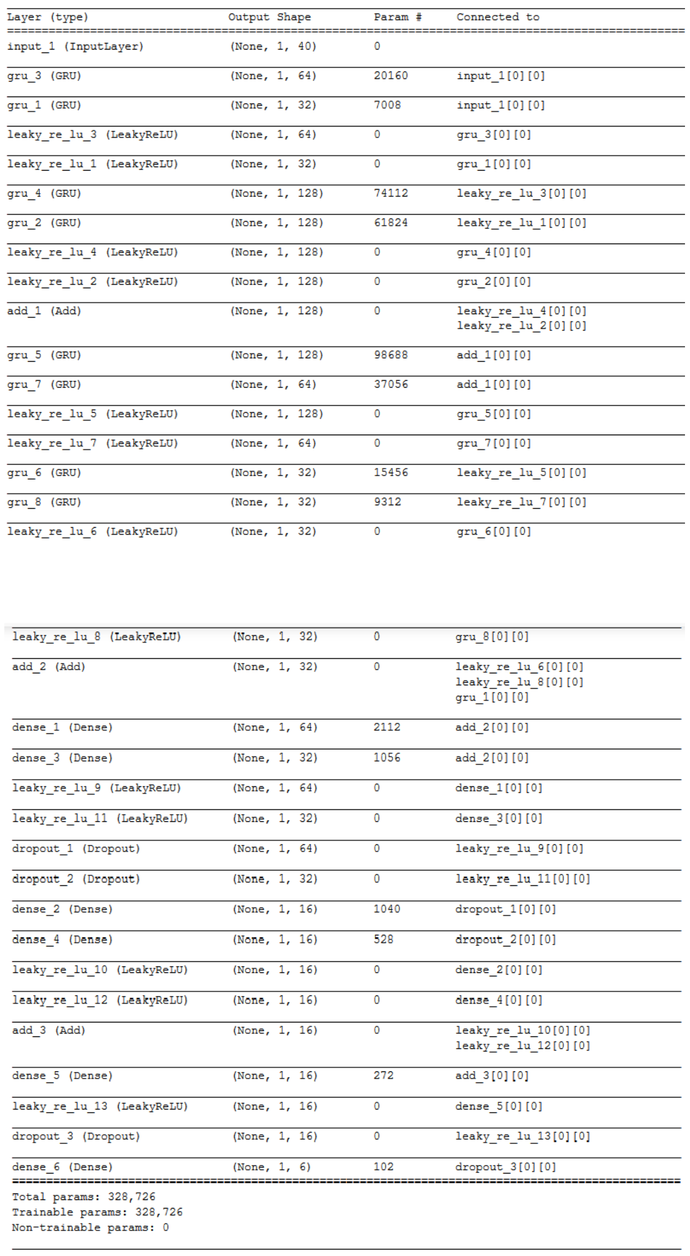
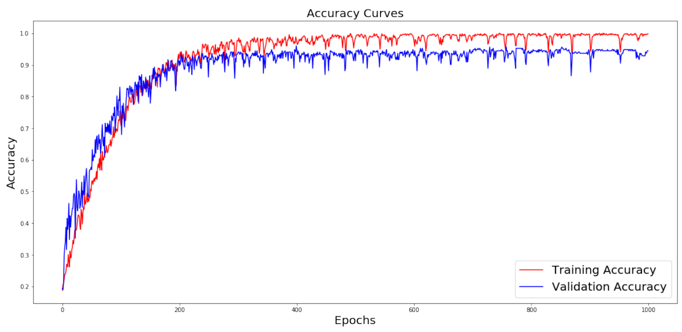

```
Layer (type)                 Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)         (None, 1, 40)        0

gru_3 (GRU)                  (None, 1, 64)        20160       input_1[0][0]

gru_1 (GRU)                  (None, 1, 32)        7008        input_1[0][0]

leaky_re_lu_3 (LeakyReLU)    (None, 1, 64)        0           gru_3[0][0]

leaky_re_lu_1 (LeakyReLU)    (None, 1, 32)        0           gru_1[0][0]

gru_4 (GRU)                  (None, 1, 128)       74112       leaky_re_lu_3[0][0]

gru_2 (GRU)                  (None, 1, 128)       61824       leaky_re_lu_1[0][0]

leaky_re_lu_4 (LeakyReLU)    (None, 1, 128)       0           gru_4[0][0]

leaky_re_lu_2 (LeakyReLU)    (None, 1, 128)       0           gru_2[0][0]

add_1 (Add)                  (None, 1, 128)       0           leaky_re_lu_4[0][0]
                                                              leaky_re_lu_2[0][0]

gru_5 (GRU)                  (None, 1, 128)       98688       add_1[0][0]

gru_7 (GRU)                  (None, 1, 64)        37056       add_1[0][0]

leaky_re_lu_5 (LeakyReLU)    (None, 1, 128)       0           gru_5[0][0]

leaky_re_lu_7 (LeakyReLU)    (None, 1, 64)        0           gru_7[0][0]

gru_6 (GRU)                  (None, 1, 32)        15456       leaky_re_lu_5[0][0]

gru_8 (GRU)                  (None, 1, 32)        9312        leaky_re_lu_7[0][0]

leaky_re_lu_6 (LeakyReLU)    (None, 1, 32)        0           gru_6[0][0]

leaky_re_lu_8 (LeakyReLU)    (None, 1, 32)        0           gru_8[0][0]

add_2 (Add)                  (None, 1, 32)        0           leaky_re_lu_6[0][0]
                                                              leaky_re_lu_8[0][0]
                                                              gru_1[0][0]

dense_1 (Dense)              (None, 1, 64)        2112        add_2[0][0]

dense_3 (Dense)              (None, 1, 32)        1056        add_2[0][0]

leaky_re_lu_9 (LeakyReLU)    (None, 1, 64)        0           dense_1[0][0]

leaky_re_lu_11 (LeakyReLU)   (None, 1, 32)        0           dense_3[0][0]

dropout_1 (Dropout)          (None, 1, 64)        0           leaky_re_lu_9[0][0]

dropout_2 (Dropout)          (None, 1, 32)        0           leaky_re_lu_11[0][0]

dense_2 (Dense)              (None, 1, 16)        1040        dropout_1[0][0]

dense_4 (Dense)              (None, 1, 16)        528         dropout_2[0][0]

leaky_re_lu_10 (LeakyReLU)   (None, 1, 16)        0           dense_2[0][0]

leaky_re_lu_12 (LeakyReLU)   (None, 1, 16)        0           dense_4[0][0]

add_3 (Add)                  (None, 1, 16)        0           leaky_re_lu_10[0][0]
                                                              leaky_re_lu_12[0][0]

dense_5 (Dense)              (None, 1, 16)        272         add_3[0][0]

leaky_re_lu_13 (LeakyReLU)   (None, 1, 16)        0           dense_5[0][0]

dropout_3 (Dropout)          (None, 1, 16)        0           leaky_re_lu_13[0][0]

dense_6 (Dense)              (None, 1, 6)         102         dropout_3[0][0]
==================================================================================================
Total params: 328,726
Trainable params: 328,726
Non-trainable params: 0
```

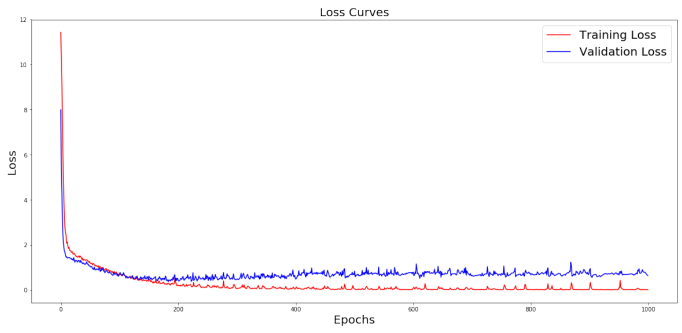
Fig. 3. Summary of the neural network model

are randomly ignored or dropped out during training. It prevents overfitting. In our model, there is a dropout layer between every two Dense layers, the dropout rate between every two dense layers is 0.2 except the last dropout layers between last two dense layer where the dropout rate is 0.5.

- **Add Layers** - It adds features generated by layers of same shape.

In our neural network model, we have also added layer generating an output of similar dimension in between our neural network. We have used **Model Checkpoint** to monitor our model performance and saved the best model weights with respect to the validation accuracy for predictions while training.
While compiling the model **Categorical Cross-entropy** is used as the loss function. We have also used **Adamax** as our Optimizer or optimizing algorithm.





III. Model performance evaluation

The metrics that we have used to evaluate the performance of the model are **Accuracy**, **Precision**, **Recall/sensitivity**, **F1-score**, **Cohens kappa score**(CK), **Matthews correlation coefficient**(MCC). For all the above metrics if the score is 1, then the model is performing in the best way possible and predicting every class perfectly. Lower the score more is the model facing difficulty while predicting the correct class and also indicates poor model performance.
The test data was 25% of the entire data-set and the rest was used for training the model. The train-test split was always random split therefore there is a little fluctuation of the score of evaluation metrics. We have performed our experiment twenty times, and selected unique readings from them for this research paper as shown in TABLE-1. We validated our model performance with the best model weight concerning the validation score during training the model. The model was trained for 1000 iterations.

TABLE I
Model evaluation

	accuracy	precision	recall	f1-score	CK	MCC
1	0.9525	0.9548	0.9526	0.9522	0.9426	0.9432
2	0.9567	0.9589	0.9565	0.9566	0.9474	0.9479
3	0.9605	0.9621	0.9605	0.9604	0.9521	0.9525
4	0.9490	0.9507	0.9486	0.9485	0.9378	0.9383
5	0.9486	0.9509	0.9486	0.9487	0.9378	0.9382
6	0.9604	0.9623	0.9605	0.9603	0.9522	0.9526
7	0.9565	0.9589	0.9565	0.9566	0.9474	0.9479
8	0.9604	0.9618	0.9605	0.9604	0.9521	0.9525
9	0.9565	0.9599	0.9565	0.9566	0.9474	0.9480
10	0.9604	0.9625	0.9605	0.9601	0.9520	0.9534

The readings in TABLE-1 are in the score format(between 0 and 1), to get the equivalent percentage multiply the score with 100.
Our model has achieved a **training accuracy** of 99.9% - 100%. TABLE-1 represents the list of **validation accuracy** we have achieved throughout the experiment.

Therefore the overall performance of our model over unknown data or validation data are:

- Accuracy: $95.67\,\% \pm 0.77\,\%$
- Precision: $95.89\,\% \pm 0.8\,\%$
- Recall/Sensitivity: $95.65\,\% \pm 0.753\,\%$
- f1-score: $95.66\,\% \pm 0.79\,\%$
- Cohens kappa score: $94.74\,\% \pm 0.96\,\%$
- Matthews correlation coefficient: $94.79\,\% \pm 0.96\,\%$

A. *Classification report*

TABLE II
Classification report

	precision	recall	f1-score	Support
Bronchiectasis	0.98	1.00	0.99	44
Bronchiolitis	1.00	1.00	1.00	30
COPD	1.00	0.91	0.95	57
Healthy	0.94	0.92	0.93	36
Pneumonia	0.92	1.00	0.96	36
URTI	0.94	0.96	0.94	53
macro avg	0.96	0.97	0.96	253
weighted avg	0.96	0.96	0.96	253

B. *Comparison of our model performance with previous research papers*

First of all the model that we have created is a multi-class classifier that classifies six different classes and also we have used six different metrics to evaluate the performance of our

model. We are proud to declare that our model is the very class six class classifier model compared to the previous research papers, which itself is an improvement in the field of medical science and A.I.
The research papers shown below might not be a multi-label classifier or might not use the same parameters for evaluating their respective models.

- **Detection of explosive cough events in audio recordings by internal sound analysis**. [1]. sensitivity - 92.3 % ± 2.3 %; specificity - 84.7 % ± 3.3 %
- **Detection of crackle events using a multi-feature approach** .[2]. sensitivity - 76 %; precision - 77 %.
- **Integrated approach for automatic crackle detection based on fractal dimension and box filtering** .[11]. Sensitivity - 89 % ± 10 %; positive predictive value - 95 % ± 11 %; F-score - 92 % ± 10 %
- **Detection of Lungs Status Using Morphological Complexities of Respiratory Sounds**.[5]. Accuracy - 92.86 %; sensitivity - 86.30 %; specificity - 86.90 %.
- **An Expert Diagnostic System to Automatically Identify Asthma and Chronic Obstructive Pulmonary Disease in Clinical Settings**. [9]. Sensitivity - 96.45 %; sensitivity - 96 %; specificity - 98.71 %.
- **Lung disease detection using feature extraction and extreme learning machine**.[10]. Accuracy - 96 %

C. Device used to compute the model and evaluate it

We have implemented the model on a machine whose configurations are given below,
Processor: Intel(R) Core(TM)i5-5200U
GPU: GeForce 920M,compute capability: 3.5
CPU: 2.20 GHz

D. Future Scopes

This research could be taken to further improvements with different neural network architecture or maybe with different machine learning or deep learning algorithm. A different feature extraction technique could be applied for further improvement in performance. Using Generative Adversarial Networks for data augmentation could also be a very good approach.

IV. Conclusion

Medical research and medical science could be progressed further with the help of artificial intelligence. The neural network architecture has performed better than our expectations. But still, it needs a lot of improvements to achieve higher accuracy during prediction. We hope our research would inspire future researchers to work on this subject and wish they would approach with a better and encouraging solution.

V. Acknowledgement

We are highly thankful to the organizations that created the dataset and maintained it. We are also thankful to Int. Conf. on Biomedical Health Informatics - ICBHI for making the dataset available which saved a lot of search time. Our source of inspiration was always the researchers and their researches who worked on this subject.

VI. References

List of references which helped us a lot throughout the research are given below:

[1] Rocha BM, Mendes L, Couceiro R, Henriques J, Carvalho P, Paiva R. Detection of Explosive Cough Events in Audio Recordings by Internal Sound Analysis Internal Sound Analysis. 39th Annu Int Conf IEEE Eng Med Biol Soc 2017;2761-6

[2] L. Mendes, Ioannis Vogiatzis, Eleni Perantoni, Evangelos Kaimakamis, Ioanna Chouvarda, N. Maglaveras, Jorge Henriques, Paulo de Carvalho, Rui Pedro Paiva. Detection of crackle events using a multi-feature approach. 38th Annu Int Conf IEEE Eng Med Biol Soc. IEEE; 2016;367983.

[3] Guntupalli KK, Alapat PM, Bandi VD, Kushnir I. Validation of automatic wheeze detection in patients with obstructed airways and in healthy subjects. J Asthma. 2008;45(10):9037.

[4] B. M. Rocha1, D.Filos, L. Mendes, I. Vogiatzis, E.Perantoni, E.Kaimakamis, P.Natsiavas, A. Oliveira, C. Jcome, A. Marques, R. P. Paiva, I. Chouvarda,P. Carvalho, N. Maglaveras. Respiratory Sound Database for the Development of Automated Classification. International Conference on Biomedical and Health Informatics ICBHI 2017: Precision Medicine Powered by pHealth and Connected Health pp 33-37

[5] Ashok Mondal, Parthasarathi Bhattacharya, and Goutam Saha. Detection of Lungs Status Using Morphological Complexities of Respiratory Sounds. ScientificWorldJournal. 2014; 2014: 182938. doi: 10.1155/2014/182938.PMCID: PMC3933370

[6] Chamberlain D, Kodgule R, Ganelin D, Miglani V, Fletcher RR. Application of Semi-Supervised Deep Learning to Lung Sound Analysis. 38th Annu Int Conf IEEE Eng Med Biol Soc; 2016;8047.

[7] Lartillot O, Lartillot O, Toiviainen P, Toiviainen P. A matlab toolbox for musical feature extraction from audio. Int Conf Digit AudioEffects2007;(Ii):18.

[8] Olivier Lartillot, Petri Toiviainen. A MATLAB TOOLBOX FOR MUSICAL FEATURE EXTRACTION FROM AUDIO. Proc. of the 10thInt. Conference on Digital Audio Effects (DAFx-07), Bordeaux, France, September 10-15, 200

[9] Almir Badnjevic, Lejla Gurbeta, and Eddie Custovic. An Expert Diagnostic System to Automatically Identify Asthma and Chronic Obstructive Pulmonary Disease in Clinical Settings. SCIENTIFIC REPORTS — (2018) 8:11645 — DOI:10.1038/s41598-018-30116-2

[10] Geraldo Luis Bezerra Ramalho, Pedro Pedrosa Rebouas Filho, Ftima Nelsizeuma Sombra de Medeiros, Paulo

Csar Cortez.Lung disease detection using feature extraction and extreme learning machine. Sociedade Brasileira de Engenharia Biomdica.(Brazilian journal of biomedical engineering) version ISSN 1517-3151

[11] Ctia Pinho, Ana Oliveira, Cristina Jcome, Joo Manuel Rodrigues, Alda Marques.Integrated Approach for Automatic Crackle Detection Based on Fractal Dimension and Box Filtering. International Journal of Reliable and Quality E-Healthcare. Volume 5 Issue 4, October 2016. doi - 10.4018/IJRQEH.2016100103

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