JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

DEEP LEARNING SOLUTIONS FOR RESPIRATORY DISORDERS THROUGH SOUND PATTERNS

[1]Mallisetty Siva Mahesh,[2]Kattamuri B N Ayyappa,[3]Maddela Murali,

[4]Mididoddi Surendra Babu,[5]Nagababu Pachhala

[1,2,3,4]Student,[5]Assistant Professor
[1]Information Technology,
[1-5]Vasireddy Venkatadri Institute of Technology, Guntur, India

Abstract: The goal of our experiment is to investigate if deep learning can identify pulmonary conditions from electronically recorded lung sounds. While support vector machines and K-Nearest Neighbors algorithms are examples of machine learning techniques, they are not very accurate in diagnosing pulmonary disorders. Thus, we presented a deep neural network model that identifies the respiratory system's state based on breathing sound input. A number of performance assessment criteria, such as Cohen's kappa, accuracy, sensitivity, specificity, precision, and F1-score, were used to assess the model's training. By fusing CNN and GRU, the proposed algorithm was able to categorize patients based on the categories of pulmonary diseases with the highest average accuracy and precision. A total of 103 patients from locally recorded stethoscope lung sounds at Abdullah University Hospital, Jordan University of Science and Technology, Jordan, were included in the chosen data-set. use of deep neural networks or deep learning to predict respiratory conditions such as bronchiectasis, pneumonia, bronchiolitis, URTIs (upper respiratory tract infections), and COPD. This work opens the door for the application of deep learning models in clinical settings to support physicians in making diagnose-related decisions regarding pulmonary disorders.

***Index Terms*–Deep Learning, GRU, Respiratory Diseases, Classification**

I. INTRODUCTION

This research study will address the potential application of deep learning in respiratory illness detection based only on respiratory sounds. Respiratory audiographs are significant indicators of both respiratory health and respiratory diseases. Wheezing is a common sign of obstructive airway disease, such as asthma or chronic obstructive pulmonary disease (COPD). To solve the issue, we employed many neural network model topologies, choosing the one that would produce the greatest outcomes. Additionally, we enhanced the data across the dataset. The data-set we employed is made up of several patients' respiratory sounds that were recorded from different parts of the chest. To evaluate and compare, we have used a number of metrics, such as f1-score, accuracy, precision, recall, and recall.

II. LITERATURE SURVEY

[1] Only few modifications have been made to the stethoscope and its method of usage over the 200 years since Dr.Laennec designed it and its semantic of auscultatory observations. Nonetheless, in clinical practice, the ability to differentiate between normal and pathological sounds or noises (vesicular sounds, wheezes, crackles, etc.) is still crucial for accurate diagnosis and treatment. With a focus on intelligent communicative stethoscope devices in clinical practice, training, and telemedicine, it seeks to discuss current technical advancements and assess new developments and prospects in the field of auscultation.

[2] Recognizing typical and atypical respiratory sounds, such crackles and wheezes, is crucial for precise illness diagnosis. These sounds include a wealth of information on the physiologies and diseases of the lung structure, and they may be used to identify any obstructions in the airways.

[3] The ability of human hearing to detect crackles has been tested in a number of investigations. The study used synthetic crackles to overlay actual breathing and respiratory sounds. These studies revealed the most significant detection mistakes, such as variations in crackling strength and wavelength caused by distinct crackle kinds, among others. These investigations led us to the conclusion that conventional auscultation should not be used as a stand-alone method of verifying respiratory sounds.

Participants in the study by Nightingale, Wade, and Watson et al. [4] were asked to identify and find alterations in pictures depicting actual scenes. The findings demonstrated that people's capacity to recognize modified images was restricted, and they frequently failed to pinpoint the modification. Professionals working in a variety of areas using digital photographs should take note of this discovery.

Mesonet is a small facial video forgery detection network that was first presented by Afchar et al. [5]. This cutting-edge network architecture is especially made to effectively identify forgeries in face footage. Mesonet is useful for a variety of applications needing accurate fraud detection in face films because of its small size, which allows for efficient detection while consuming less processing power. Tokuda and associates.

Examined the use of a synergetic feature and classifier combination strategy to differentiate between digital pictures and computer-generated images [6]. This study, which was published in the Journal of Visual Communication and Image Representation, investigates techniques for accurately differentiating real images from computer-generated imagery. The authors' synergistic combination of characteristics and classifiers advances digital forensics and picture authentication systems.

III. EXISTING SYSTEM

In the realm of deep learning architectures for signal processing, the current system represents a novel approach, particularly with regard to the diagnosis of illnesses using lung sound data. Unlike typical techniques that need separate procedures for spatial and temporal feature extraction, our system integrates both spatial and temporal processing into a single network design. This integration is achieved by combining models of Bidirectional Long Short-Term Memory (BDLSTM) and Convolutional Neural Network (CNN). A deep learning model based on bidirectional long short-term memory (LSTM) and convolutional neural networks (CNNs) was used to classify lung sounds. The total average accuracy of the model used to classify lung sounds into different respiratory illnesses was 89%, as shown by the Cohen's kappa value of 86.26%.

This approach enables the use of models trained with deep learning in clinical settings to assist physicians in making decisions. Although some misclassification can happen, particularly when signals from people with chronic obstructive pulmonary disease (COPD) are misclassified as normal or asthmatic, the total detection error for each illness is still quite low.

IV. PROPOSED SYSTEM

Five different kinds of layers were utilized in the building of the neural network model. GRU (Gated Recurrent Unit), Leaky Relu (Leaky Rectified Linear Unit), Dense Layer, Dropout Layer, Add Layer, and 3) Dense Layer are the first five components.

The vanishing gradient problem in a conventional recurrent neural layer is the aim of the Gated Recurrent Unit (GRU). It has two different types of gates: update gates and reset gates. These gates decide which information should be forwarded to the output. The quantity of historical memory that has to be carried over into the future is decided by the update gate. The amount of history that should be deleted is decided by the reset gate. With a linear activation function, each GRU layer in our model processed the input sequence backward and produced the reverse sequence.

Leaky rectified linear unit, or leaky Relu, is the term used to describe an activation layer. A little gradient is allowed while the device is working. In our approach, a Leaky Relu layer comes after each GRU and Dense layer.

Dense Layer: There is total connection in this layer. In our model, the lowest half of the neural network consists only of thick layers.

Dropout: When this layer is added, some of the layer outputs are randomly eliminated or deleted during training. Overfitting is halted by that. Every two thick layers in our model has a dropout layer in between. Except for the last dropout layer, which has a 0.5 dropout rate between the last two thick layers, the dropout rate in this layer is 0.2.

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

V. OBJECTIVE

Using powerful machine learning methods like CNNs and RNNs, we will create a model that can reliably categorize a broad variety of respiratory disorders, from minor illnesses to life-threatening diseases. This tool will easily mesh with clinical knowledge, making it easier for medical practitioners to diagnose patients and facilitating fast treatment. Our goal is to improve the model's accuracy and reliability in illness prediction while optimizing learning through the use of robust data handling approaches and augmentation methods. Transparency and interpretability will be given top priority in this technology, which will give physicians useful information to help them make decisions. By means of thorough preprocessing and augmentation of data, we will guarantee the model's resilience and efficacy in a range of patient groups. We'll use model optimization approaches like regularization and hyperparameter tweaking to boost prediction accuracy and performance.

This strategy places a strong emphasis on inclusiveness, guaranteeing that the tool works well for a wide range of respiratory ailments and patient types. Our objective is to enhance patient outcomes in respiratory care by accelerating diagnosis, simplifying therapy, and integrating clinical integration with optimal learning methodologies.

VI. METHODOLOGY

1. **Data Collection:** To obtain the information for this module, audio files are collected, stored, and converted into text files. These are collected and recorded data from the lungs of different people. Virtualized files include audio and text.
2. **Data Preprocessing:** Data preparation is the process of converting the raw material into an understandable format. Preprocessing is one of the most crucial phases in data mining to boost data efficiency. The methods employed for data

preparation directly affect the output of any analytical program. We need to exclude any null values and unwanted values from these data because they have already undergone pre-processing.

3. **Model Selection:** We chose to base our deep learning model on the GRU+CNN architecture after giving the job specifications and available processing power some thought. A popular pre-trained model with great performance in image classification tasks is GRU+CNN.
4. **Model Training:** We utilized 70% of the preprocessed dataset for training and 30% for validation after splitting it into training and validation sets. After that, we used the training set to train the model, adjusting its parameters with the help of the binary cross-entropy loss function and the Adam optimizer.
5. **Model Evaluation:** After training was finished, we assessed the trained model's effectiveness using a test set that was held out and not used during training. We calculated a number of assessment measures, such as F1-score, recall, accuracy, and precision, to determine how well the model distinguished between actual and artificial intelligence (AI)-generated pictures.
6. **Performance Analysis:** Lastly, in order to understand the model's performance, we examined its assessment metrics and visualizations, including ROC curves and confusion matrices. Through this study, we were able to pinpoint regions that needed work and adjust the model's hyperparameters for increased resilience and accuracy. During the model assessment process, compute and present performance indicators such as Cohen's kappa, accuracy, sensitivity, specificity, precision, and F1-score.

VII. DESIGN

7.1 ARCHITECTURE DIAGRAM:

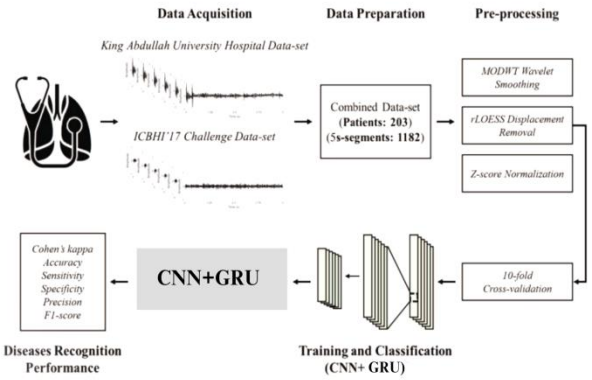


FIGURE 1: Architecture of the system

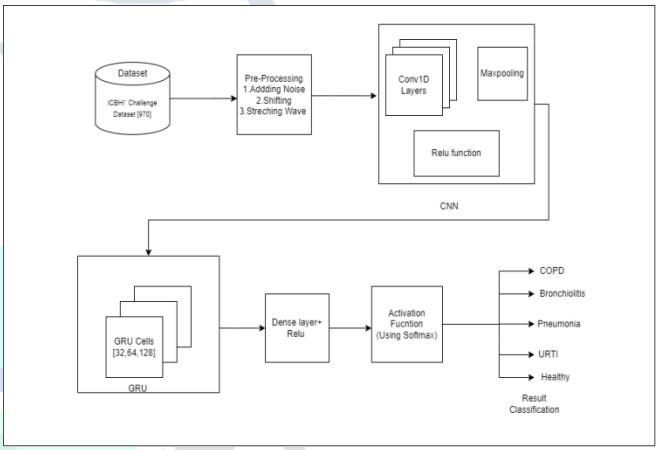


FIGURE 2: MODEL Architecture

VIII. ALGORITHM USED

1. GRU (GATED RECURRENT UNIT NETWORKS):

GRU's fundamental principle is to update the network's hidden state only on a chosen subset of time steps by use of gating methods. Information entering and leaving the network is managed by the gating mechanisms. The reset gate and the update gate are the two gating mechanisms of the GRU.The update gate defines how much of the new input should be utilized to update the hidden state, while the reset gate indicates how much of the prior hidden state should be ignored. The updated hidden state serves as the basis for calculating the GRU's output.The equations used to calculate the reset gate, update gate, and hidden state of a GRU are as follows:

Reset gate: r_t = sigmoid(W_r * [h_{t-1}, x_t])
Update gate: z_t = sigmoid(W_z * [h_{t-1}, x_t])
Candidate hidden state: h_t' = tanh(W_h * [r_t * h_{t-1}, x_t])
Hidden state: h_t = (1 – z_t) * h_{t-1} + z_t * h_t'

where W_r, W_z, and W_h are learnable weight matrices, x_t is the input at time step t, h_{t-1} is the previous hidden state, and h_t is the current hidden state.

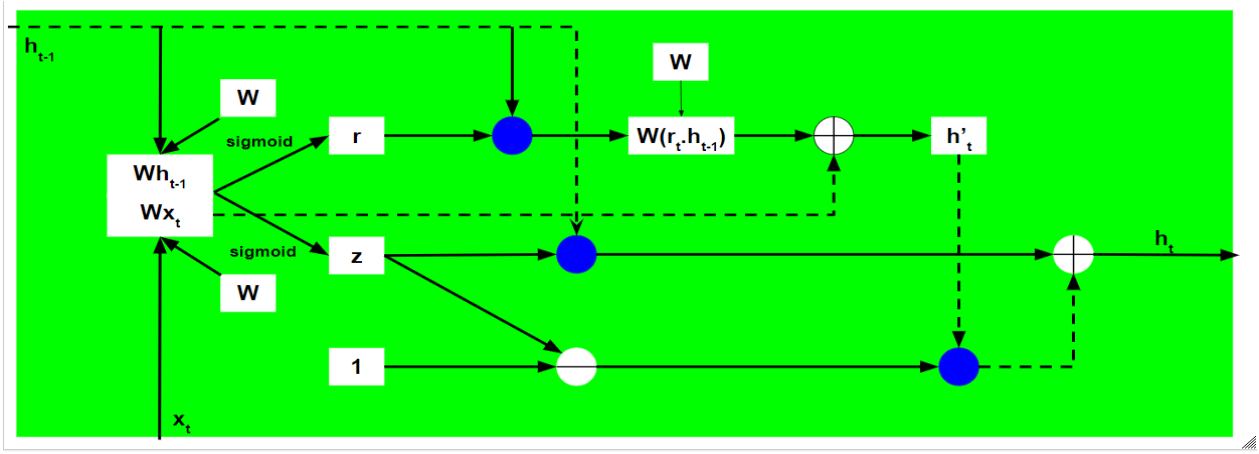


FIGURE 2: GRU Model Architecture [8].

IX. DATA SET

Respiratory Sound Database: Important markers of respiratory health and diseases are respiratory sounds. The flow of air, changes in lung tissue, and the location of secretions in the lung are all intimately connected to the sound made during breathing. For instance, wheezing is frequently indicative of an obstructive airway condition such as asthma or chronic obstructive pulmonary disease (COPD).Other recording methods including digital stethoscopes can be used to capture these sounds on tape. With the use of this digital data, machine learning may be able to identify respiratory conditions including bronchiolitis, pneumonia, and asthma automatically.Two research teams in Greece and Portugal developed the Respiratory Sound Database. There are 920 annotated recordings total, ranging in duration from 10 to 90 minutes. There were 126 patients who provided these recordings. A total of 5.5 hours of recordings totaling 6898 respiratory cycles are available; of these, 1864 have crackles, 886 have wheezes, and 506 have both. Both clear respiratory sounds and loud recordings that mimic actual environments are included in the data. All age categories of patients are included: adults, children, and the elderly.

X. IMPLEMENTATION

10.1Import all necessary Libraries

Seaborn, NumPy, OpenCV, TensorFlow, TensorFlow Keras models, TensorFlow Keras Adam optimizer, TensorFlow Keras Activation, TensorFlow Keras EarlyStopping callback, Matplotlib pyplot, TensorFlow Keras GRU, Scikit-learn confusion_matrix function, Scikit-learn classification_report function, and TensorFlow Keras GRU are among the libraries imported in this project.

10.2 GPU Availability and Memory Growth Configuration

As we go, the code verifies that we are utilizing GPUs to their full potential by determining which GPUs are available and adjusting memory growth accordingly. To begin with, it retrieves a list of physical GPU devices using TensorFlow's "list_physical_devices" function. Then, it uses "set_memory_growth" to iteratively activate memory growth for each GPU in the list. TensorFlow can now dynamically allocate RAM as needed, resulting in effective GPU resource use during model training thanks to this setup.

10.3 Data Loading and Preprocessing

```
[11] path='/content/drive/MyDrive/Respiratory_Sound_Database/Respiratory_Sound_Database/audio_and_txt_files/'
     files=[s.split('.')[0] for s in os.listdir(path) if '.txt' in s]
```

```
[12] def add_noise(data,x):
         noise = np.random.randn(len(data))
         data_noise = data + x * noise
         return data_noise

     def shift(data,x):
         return np.roll(data, x)

     def stretch(data, rate):
         data = librosa.effects.time_stretch(data, rate=rate)
         return data

     def pitch_shift (data , rate):
         data = librosa.effects.pitch_shift(data, sr=220250, n_steps=rate)
         return data
```

The designated directory is where audio files are loaded. To make model training, validation, and assessment easier, the normalized dataset is split into training, validation, and test sets in the right ratios. Moreover, during model training, batches of the data are easily accessed using a NumPy iterator. By doing this, the data is properly preprocessed, laying the groundwork for the construction and evaluation of the model that follows.

10.4 Model Architecture Definition

```
model_conv = BatchNormalization()(model_conv)
model_conv = Conv1D(512, kernel_size=5, strides=1, padding='same', activation='relu')(model_conv)
model_conv = MaxPooling1D(pool_size=2, strides = 2, padding = 'same')(model_conv)
model_conv = BatchNormalization()(model_conv)
model_2_1 = GRU(32,return_sequences=True,activation='tanh',go_backwards=True)(model_conv)
model_2 = GRU(128,return_sequences=True, activation='tanh',go_backwards=True)(model_2_1)
model_3 = GRU(64,return_sequences=True,activation='tanh',go_backwards=True)(model_conv)
model_3 = GRU(128,return_sequences=True, activation='tanh',go_backwards=True)(model_3)
model_x = GRU(64,return_sequences=True,activation='tanh',go_backwards=True)(model_conv)
model_x = GRU(128,return_sequences=True, activation='tanh',go_backwards=True)(model_x)
model_add_1 = add([model_3,model_2,model_x])
model_5 = GRU(128,return_sequences=True,activation='tanh',go_backwards=True)(model_add_1)
model_5 = GRU(32,return_sequences=True, activation='tanh',go_backwards=True)(model_5)
model_6 = GRU(64,return_sequences=True,activation='tanh',go_backwards=True)(model_add_1)
model_6 = GRU(32,return_sequences=True, activation='tanh',go_backwards=True)(model_6)
model_add_2 = add([model_5,model_6,model_2_1])
model_7 = Dense(32, activation=None)(model_add_2)
model_7 = LeakyReLU()(model_7)
model_7 = Dense(128, activation=None)(model_7)
model_7 = LeakyReLU()(model_7)
model_9 = Dense(64, activation=None)(model_add_2)
model_9 = LeakyReLU()(model_9)
model_9 = Dense(128, activation=None)(model_9)
model_9 = LeakyReLU()(model_9)
model_add_3 = add([model_7,model_9])
model_10 = Dense(64, activation=None)(model_add_3)
model_10 = LeakyReLU()(model_10)
model_10 = Dense(32, activation=None)(model_10)
model_10 = LeakyReLU()(model_10)
model_10 = Dense(5, activation="softmax")(model_10)
gru_model = Model(inputs=Input_Sample, outputs = model_10)
gru_model.summary()
```

The model architecture is specified in this cell. First, weights pretrained on ImageNet are put into the pre-trained GRU model basis. Then, that network is expanded using the Conv1D layer. After adding the underlying GRU model, other layers are added, such as output layers, batch normalization, flattening, dense, and LeakyRelu.

10.5 Setting Up Learning Rate Scheduler and Optimizer

```
optimiser = tf.keras.optimizers.Adam(learning_rate = 0.0001)
gru_model.compile(optimizer=optimiser, loss='categorical_crossentropy',metrics=['accuracy'])

cb = [EarlyStopping(patience=300,monitor='accuracy',mode='max',restore_best_weights=True),
      ModelCheckpoint("/diagnosis_GRU_CNN_1.h5",save_best_only=True)]
```

We create a learning rate scheduler and a model optimizer in this stage. 0.001 is the starting learning rate. `lr_scheduler`, the learning rate scheduler function, is designed to reduce the learning rate by 5% at the end of each epoch. The Adam optimizer is also instantiated with the given starting learning rate. This gets the model ready for compilation in the next stages.

10.6 Model Training

```
history = gru_model.fit(x_train_gru, y_train_gru, batch_size=8, epochs=800, validation_data=(x_val_gru, y_val_gru), callbacks = cb)
98/98 [==============================] - 2s 24ms/step - loss: 0.0891 - accuracy: 0.9655 - val_loss: 0.6640 - val_accuracy: 0.8833
Epoch 773/800
98/98 [==============================] - 3s 33ms/step - loss: 0.1129 - accuracy: 0.9552 - val_loss: 0.6165 - val_accuracy: 0.8722
Epoch 774/800
98/98 [==============================] - 2s 24ms/step - loss: 0.1240 - accuracy: 0.9540 - val_loss: 0.6810 - val_accuracy: 0.8556
Epoch 775/800
98/98 [==============================] - 2s 22ms/step - loss: 0.1428 - accuracy: 0.9425 - val_loss: 0.6363 - val_accuracy: 0.8778
Epoch 776/800
98/98 [==============================] - 2s 23ms/step - loss: 0.1475 - accuracy: 0.9450 - val_loss: 0.5524 - val_accuracy: 0.8944
Epoch 777/800
98/98 [==============================] - 2s 23ms/step - loss: 0.1140 - accuracy: 0.9552 - val_loss: 0.6221 - val_accuracy: 0.8722
Epoch 778/800
98/98 [==============================] - 3s 28ms/step - loss: 0.1034 - accuracy: 0.9578 - val_loss: 0.5713 - val_accuracy: 0.8889
Epoch 779/800
98/98 [==============================] - 3s 29ms/step - loss: 0.0859 - accuracy: 0.9668 - val_loss: 0.6357 - val_accuracy: 0.8667
Epoch 780/800
98/98 [==============================] - 2s 23ms/step - loss: 0.0961 - accuracy: 0.9616 - val_loss: 0.6048 - val_accuracy: 0.8889
Epoch 781/800
98/98 [==============================] - 2s 23ms/step - loss: 0.0745 - accuracy: 0.9719 - val_loss: 0.6952 - val_accuracy: 0.8944
Epoch 782/800
98/98 [==============================] - 2s 23ms/step - loss: 0.1170 - accuracy: 0.9604 - val_loss: 0.7791 - val_accuracy: 0.8833
Epoch 783/800
98/98 [==============================] - 2s 22ms/step - loss: 0.1028 - accuracy: 0.9565 - val_loss: 0.6106 - val_accuracy: 0.8889
Epoch 784/800
98/98 [==============================] - 3s 30ms/step - loss: 0.0978 - accuracy: 0.9655 - val_loss: 0.5896 - val_accuracy: 0.8833
```

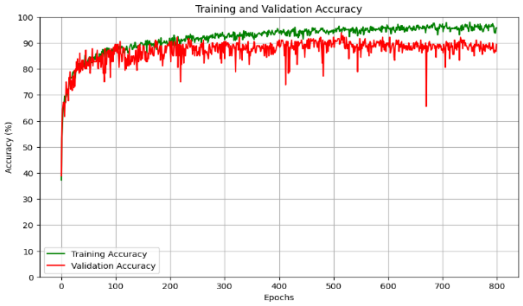
changes to the learning rate scheduler and optimizer design, improving the robustness and generalization capacity of the model. After that, the model is trained on the enhanced data for ten epochs, with early stopping put in place to avoid overfitting. This all-inclusive method includes both model training and data augmentation, which are essential phases in the creation of reliable machine learning models.

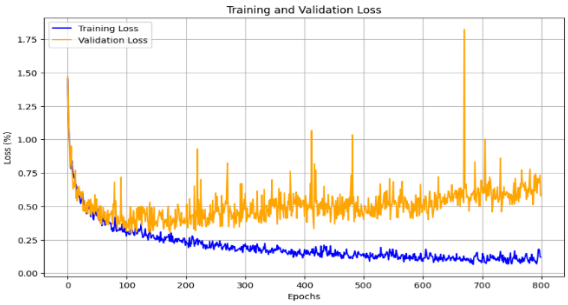
10.7Model Evaluation and Training Visualization

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Assuming gru_model and x_train_gru, y_train_gru are defined

# 1. Evaluate the model
evaluation_result = gru_model.evaluate(x_train_gru, y_train_gru)

# Extracting loss and accuracy
loss = evaluation_result[0]
accuracy = evaluation_result[1]

# Print the loss and accuracy
print("Loss: {:.2f}%".format(loss * 100))
print("Accuracy: {:.2f}%".format(accuracy * 100))

# 2. Define functions to plot individual graphs
def plot_loss(history):
    plt.figure(figsize=(10, 6))
    plt.plot(history.history['loss'], label='Training Loss', color='blue')
    plt.plot(history.history['val_loss'], label='Validation Loss', color='orange')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss (%)')  # Modify ylabel
    plt.legend()
    plt.grid(True)
    plt.show()

# Plot individual graphs
plot_loss(history)
```

Following model training, the next phase involves evaluating the model's performance on the test dataset and visualizing the training and validation accuracy and loss. This step provides insights into the model's generalization capability and training progress.

10.8Model Performance Evaluation on Test Dataset:





At this point, we use measures for accuracy, recall, and precision on the test dataset to evaluate the model's performance. First, the precision, recall, and binary accuracy metrics are initialized for the evaluation. The test dataset's labels are then predicted by the model, and the evaluation metrics are updated appropriately. Ultimately, the model's accuracy, recall, and precision values are presented for examination and interpretation.

10.9Model Persistence and Image Prediction:

```
def gru_diagnosis_prediction(test_audio):
    data_x, sampling_rate = librosa.load(test_audio)
    data_x = stretch (data_x,1.2)

    features = np.mean(librosa.feature.mfcc(y=data_x, sr=sampling_rate, n_mfcc=52).T,axis = 0)

    features = features.reshape(1,52)

    test_pred = gru_model.predict(np.expand_dims(features, axis = 1))
    classpreds = classes[np.argmax(test_pred[0], axis=1)[0]]
    confidence = test_pred.T[test_pred[0].mean(axis=0).argmax()].mean()

    print (classpreds , confidence)
```

```
gru_diagnosis_prediction("/content/drive/MyDrive/Respiratory_Sound_Database/Respiratory_Sound_Database/audio_and_txt_files/117_1b2_Tc_mc_LittC2
```

```
1/1 [==============================] - 0s 69ms/step
COPD 0.999977
```

Using TensorFlow's `save` function, we store the trained model to disk in the format ('tf' for TensorFlow Saved Model format) in this step. Next, we utilize TensorFlow's `load_model} function to load the stored model back into memory for later usage or deployment. Furthermore, we use the loaded model on a sample Audio to forecast diseases. After loading, preprocessing, displaying, and resizing the audio to fit the model's input format, the model is run through to make predictions. During this procedure, any errors that are found are handled and shown correctly.

10.10 Generating Confusion Matrix and Classification Report

```
classes = ["COPD" ,"Bronchiolitis ", "Pneumoina", "URTI", "Healthy"]

preds = gru_model.predict(x_test_gru)
classpreds = [np.argmax(t) for t in preds ]
y_testclass = [np.argmax(t) for t in y_test_gru]
cm = confusion_matrix(y_testclass, classpreds)

plt.figure(figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')
ax = sns.heatmap(cm, cmap='Blues', annot=True, fmt='d', xticklabels=classes, yticklabels=classes)

plt.title('')
plt.xlabel('Prediction')
plt.ylabel('Truth')
plt.show(ax)
```

By calculating the confusion matrix and producing a classification report based on the model's predictions on the test dataset, we evaluate the model's performance in this phase. From the test set, the real labels and expected probabilities are taken out. The probabilities are then transformed into binary predictions by using a 0.5 threshold. The `confusion_matrix` function from scikit-learn is then used to generate the confusion matrix, which offers insights into the model's performance in various classes. Furthermore, the 'classification_report' function generates a classification report that includes comprehensive metrics for every class, including accuracy, recall, and F1-score. Lastly, a heatmap is used to represent the confusion matrix, and the categorization report is shown for in-depth examination.

XI. RESULT ANALYSIS

In the result analysis, we evaluate the model's performance through multiple metrics and visualizations:

Confusion Matrix: The confusion matrix provides a tabular representation of the model's predictions versus the actual labels . It helps us understand how well the model is distinguishing between the five classes, showing the counts of true positives, false positives, true negatives, and false negatives.

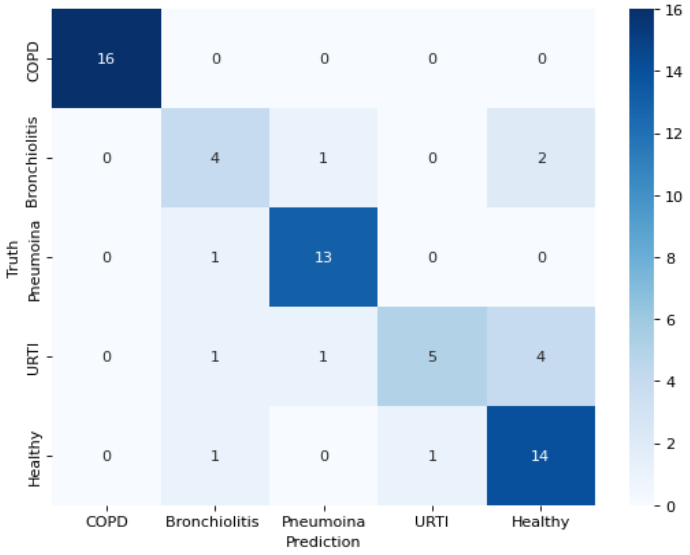


FIGURE 3: Confusion Matrix

1. **Audio Prediction Accuracy:** The accuracy of audio predictions gives an overall indication of how well the model is classifying audios correctly. It is a crucial metric to assess the model's overall performance on the test dataset.

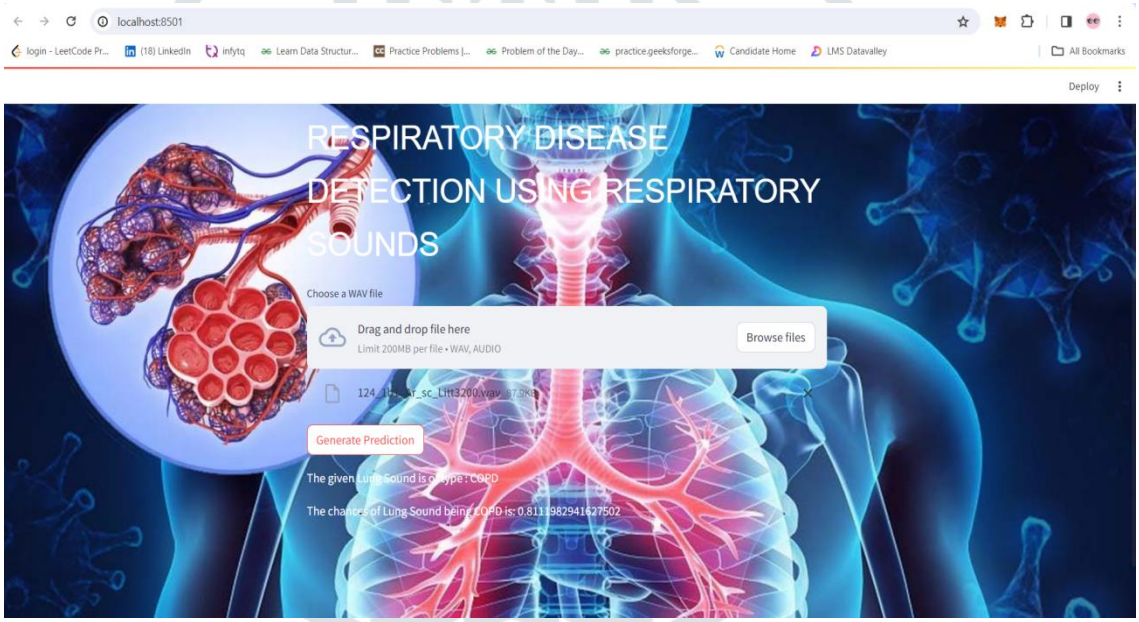


Fig: It displays the audio signal and the prediction of disease along with accuracy.

2. **Classification Report:** The classification report summarizes various classification metrics such as precision, recall, F1-score, and support for each class . It provides insights into the model's performance for each class, including its ability to correctly classify AI-generated and real images.
3.

```
[ ]  print(classification_report(y_testclass, classpreds, target_names=classes))
```

	precision	recall	f1-score	support
COPD	1.00	1.00	1.00	16
Bronchiolitis	0.57	0.57	0.57	7
Pneumoina	0.87	0.93	0.90	14
URTI	0.83	0.45	0.59	11
Healthy	0.70	0.88	0.78	16
accuracy			0.81	64
macro avg	0.79	0.77	0.77	64
weighted avg	0.82	0.81	0.80	64

FIGURE 4: Classification Report

These analyses collectively offer insights into the model's performance, highlighting areas where it excels and areas where it may need improvement. They guide further iterations of model development and fine-tuning to enhance its effectiveness for real-world applications.

XII. CONCLUSION

In conclusion, In this system, we have predicted respiratory diseases from the respiratory sounds database using Deep learning Models like GRU with CNN layers. In the preprocessing, we are handling the missing data, normalizing the values and eliminating any unwanted data from our dataset and creating a new preprocessed data. The prediction with GRU and CNN gives an accuracy rate of 95 percent.

XIII. FUTURE WORK

The future work of the application consists of mostly collecting more data and trying to implement the model with higher accuracy. Also instead of manual annotation of the audio files, teach the model to automatically annotate the recordings. This web application can add storage functionalities where the users can access their previous breath sound checks and also the automatic annotation process of sounds which helps the users to easily identify the disease. A desktop or mobile application can also be built to make the process easier for the users.

XIV. REFERENCES

[1] Andrs, E. (2012) 'Advances and Perspectives in the Field of Auscultation, with a Special Focus on the Contribution of New Intelligent Communicating Stethoscope Systems in Clinical Practice, in Teaching and Telemedicine'. In Hajjam El Hassani,A. (ed.) EHealth and Remote Monitoring. InTech.
[2] Sandra Reichert, Gass Raymond, Christian Brandt, Emmanuel Andres 'Analysis of Respiratory Sounds: State of the Art'. Clinical Medicine. Circulatory, Respiratory and Pulmonary Medicine.
[3] H Kiyokawa, M Greenberg, K Shirota, H Pasterkamp 'Auditory detection of simulated crackles in breath sounds' Chest. 2001 Jun; 119(6):1886-92.
[4] Welsby, P.D., Parry, G. and Smith, D. 'The Stethoscope: Some Preliminary Investigations'. Postgraduate Medical Journal (2003)
[5] Swarup, S. and Makaryus, A.N. 'Digital Stethoscope: Technology Update'.Medical Devices (2018)
[6] Sandra Reichert, Gass Raymond, Christian Brandt, Emmanuel Andres 'Analysis of Respiratory Sounds: State of the Art'. Clinical Medicine. Circulatory, Respiratory and Pulmonary Medicine, (2008)
[7] H Kiyokawa, M Greenberg, K Shirota, H Pasterkamp 'Auditory detection of simulated crackles in breath sounds' Chest. 2001 Jun;119(6):1886-92.doi: 10.1378/chest.119.6.1886.(2001)
[8] Zimmerman, B. and Williams, D. 'Lung Sounds'. In StatPearls. Treasure Island (FL): StatPearls Publishing. Available at: http://www.ncbi.nlm.nih.gov/books/NBK537253/ . (2020a)
[9] Zimmerman, B. and Williams, D 'Lung Sounds'. In StatPearls. Treasure Island (FL): StatPearls Publishing. Available at: http://www.ncbi.nlm.nih.gov/books/NBK537253/ (2020b)
[10] Emmanuel Andres, Amir Hajjam. 'Advances and Perspectives in the Field of Auscultation, with a Special Focus on the Contribution of New Intelligent Communicating Stethoscope Systems in Clinical Practice, in Teaching and Telemedicine'.EHealth and Remote Monitoring. InTech. (2012)
[11] B. M. Rocha, D. Filos, L. Mendes, I. Vogiatzis, E. Perantoni, E. Kaimakamis, P. Natsiavas, A. Oliveira, C. Jácome, A. Marques, R. P. Paiva, I. Chouvarda, P. Carvalho, N. MaglaverasA. 'Respiratory Sound Database for the Development of Automated Classification' Precision Medicine Powered by PHealth and Connected Health. IFMBE Proceedings. Singapore: Springer Singapore(2018)
[12] Renars Xaviero Adhi Pramono, Stuart bowyer, Esther Rodriguez-Villegas 'Automatic Adventitious Respiratory Sound Analysis: A Systematic Review' (2017)
[13] Singh, A., Thakur, N. and Sharma, A. 'A Review of Supervised Deep learning Algorithms'3rd International Conference on Computing for Sustainable Global Development (INDIACom) (2016)
[14] Georgios Ntritsos, Jacob Franek, Lazaros Belbasis, Maria A Christou, Georgios Markozannes, Pablo Altman, Robert Fogel, Tobias Sayre,Evangelia E Ntzani, Evangelos Evangelou. 'Gender-specific estimates of COPD prevalence: a systematic review and meta-analysis' International Journal of Chronic Obstructive Pulmonary Disease (2018).
[15] Pachhala, N, Jothilakshmi, S & Battula, BP 2021, 'A Comprehensive Survey on Identification of Malware Types and Malware Classification Using Machine Learning Techniques,' Proceedings of 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, pp. 1207-1214, doi: 10.1109/ICOSEC51865.2021.9591763.