A Major-Project Report

On

Respiratory Lung Disease Classification with Audio Sounds Using CNN

Submitted in partial fulfillment for the Degree of B.Tech.

In

Artificial Intelligence

By

P.HARIKA	[21911A35A7]
S.RAMYA	[21911A35B9]
B.RAMBABU	[21911A3572]
D.SRUJANA	[21911A3579]

Under the guidance of

Mrs.Ch.Swetha

Assistant Professor, Dept. of Artificial Intelligence

DEPARTMENT OF ARTIFICIAL INTELLIGENCE

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

[Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH]

2024 – 2025

CERTIFICATE

This is to certify that the project report entitled Respiratory Lung Disease Classification With Audio Sounds Using CNN submitted by P.Harika [21911A35A7], S.Ramya [21911A35B9], B.Rambabu [21911A3572], D.Srujana [21911A3579] to Vidya Jyothi Institute of Technology (An Autonomous Institution), Hyderabad, in partial fulfillment for the award of the degree of B. Tech. in Artificial Intelligence a Bonafide record of project work carried out by us under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree.

Signature of Supervisor	Signature of HOD
Mrs.Ch.Swetha	Dr. A. Obulesh
Assistant Professor	HOD
Dept. of Artificial Intelligence	Dept. of Artificial Intelligence

Signature of External Examiner

DECLARATION

We declare that this project report titled **Respiratory Disease Classification with Audio Sounds using CNN Classification** submitted in partial fulfillment of the degree of **B.Tech in Artificial Intelligence** is a record of original work carried out by us under the supervision of **Mrs.Ch.Swetha,** Assistant Professor and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice of reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

P. HARIKA	[21911A35A7]
S. RAMYA	[21911A375B9]
B. RAMBABU	[2191A3572]
D.SRUJANA	[21911A3579]

Date :

ACKNOWLEDGEMENT

We extend our heartfelt thanks to **Mrs. Ch. Swetha, Assistant Professor** for her invaluable mentorship and guidance throughout the project. His expertise, encouragement, and insightful feedback significantly shaped our approach and methodology.

We are also thankful to **Mrs. Ch. Swetha, Assistant Professor** for enriching our understanding and providing innovative solutions to the challenges we encountered during the numerous discussions, brainstorming sessions, and academic exchanges that took place within our project team. These interactions for their exceptional support and guidance throughout the project.

We wish to express our heartfelt thanks to **Dr. A. Obulesh, HOD Of Artificial Intelligence**, for guiding and supporting us throughout the course.

We wish to express our heartfelt thanks to the **Principal Dr. A. Srujana** whose support was indispensable to us during the course.

We wish to express our heartfelt thanks to our Beloved **Dean**, Accreditation and Rankings **Dr. A. Padmaja**, whose support was indispensable to us during the course. We wish to express our heartfelt thanks to our **College Management**, for providing all the infrastructure and facilities during the course.

We would like to express our sincere gratitude to our **Parents,** for their unwavering support throughout the completion of this major project.

P.HARIKA (21911A35A7)
S.RAMYA (21911A35B9)
B.RAMBABU (21911A3572)
D.SRUJANA (21911A3579)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT:

Large number of people die every year of Pulmonary chronic lung diseases irrespective of their age. Lung sound analysis has been a key diagnostic aid to accurately detect Pulmonary Diseases. Earlier, manual detection was used which was not a dependable method to detect lung diseases due to various reasons like low audibility and difference in perceptions of different physicians for different sounds. Modern computerized analysis yield results with much higher accuracy and thus a better treatment can be given to patients suffering from various kinds of lung diseases. These disorders include Asthma, Bronchitis, Emphysema, Tuberculosis and Pneumonia. Some of the symptoms are wheezing, shortness of breath, bronchi and chronic cough. In this project we are using respiratory audio dataset to predict various diseases such as Asthma, Pneumonia, Bronchiectasis and many more. To implement this project we have taken disease diagnosis dataset and respiratory audio dataset and then extract features from all audio dataset and then trained a convolution neural network (CNN) algorithm model. After training model we can upload any new test data to predict disease from it.

CHAPTER 1

INTRODUCTION

Pulmonary disorders refer to the inability of an individual to breathe normally due to various underlying diseases affecting the lungs. In the past, manual methods were used to analyze these disorders, providing only an approximate idea of the condition. As a result, treatments based on these approximations were relatively general and not highly precise. While this approach was sufficient when pulmonary diseases were less complex, the rapid increase in pollution, smoking, and other unhealthy lifestyle habits has led to more severe and intricate respiratory conditions. These modern diseases require a highly accurate estimation of their extent for proper diagnosis and treatment. Achieving this level of accuracy is possible only through automation and advanced technological interventions in medical analysis.

One of the most promising techniques in automated pulmonary disorder diagnosis is the analysis of lung sounds. Researchers have found that there is a significant difference between the sounds produced by diseased lungs and those of normal, healthy lungs. This difference serves as an excellent tool for the detailed study and early detection of pulmonary diseases. The process involves recording lung sounds, filtering out heart sounds and other noise interferences, and analyzing the waveforms of the filtered lung sounds. Over the years, several methods have been developed for the filtering and processing of lung sounds, each with varying levels of efficiency and accuracy.

A major challenge in lung sound analysis is the separation of heart sounds (HS) from lung sounds (LS) due to the spectral and temporal overlap between these two sounds. Since both sounds occur within similar frequency ranges, it becomes difficult to isolate one from the other. Various filtering techniques have been proposed to address this issue. One such method is Modulation Domain Filtering, which filters the temporal trajectories of short-term spectral components. This approach involves segmenting the signal into consecutive overlapping frames and then applying Fourier transform to analyze the frequency components. Another commonly used technique is Adaptive-Frequency Domain Filtering, which provides a relatively simple way to separate heart sounds from a combined lung-heart sound signal. This method works by subtracting the heart sounds from the mixed signal, allowing for a clearer representation of lung sounds.

With the continued advancement of machine learning, artificial intelligence, and signal processing technologies, pulmonary disorder analysis is becoming increasingly sophisticated. These automated methods not only improve diagnostic accuracy but also assist medical professionals in identifying diseases at an early stage, leading to better treatment outcomes. Future research in this field will likely focus on improving filtering techniques, enhancing machine learning models for sound classification, and integrating automated analysis systems into clinical practice.

Lung Structure

The lungs are spongy, air-filled organs responsible for oxygenating the blood and removing carbon dioxide. The key components include:

- **Trachea (Windpipe):** Connects the mouth and nose to the lungs.
- **Bronchi:** Two main tubes branching into each lung.
- **Bronchioles:** Smaller airways branching from the bronchi.
- **Alveoli:** Tiny air sacs where gas exchange occurs.

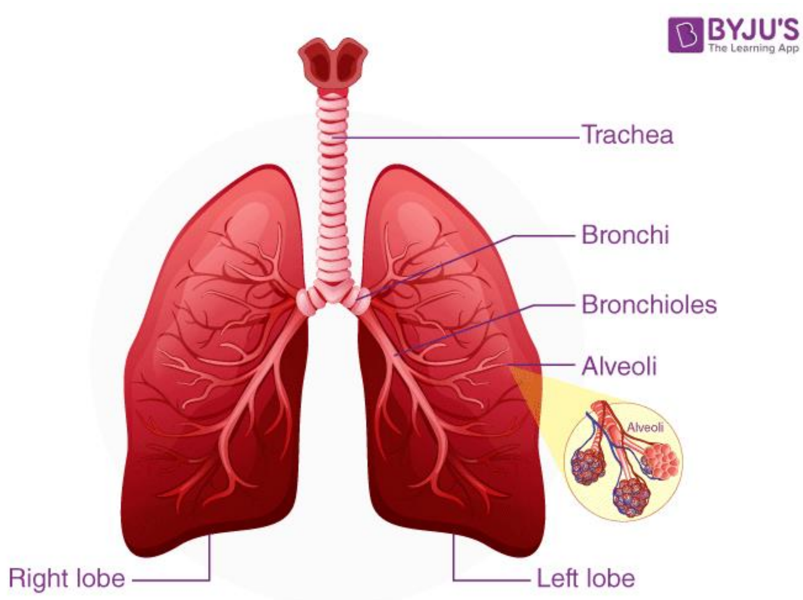


Fig 1 Lung Structure

1.1 Asthma

Asthma is a chronic inflammatory disease that causes narrowing and swelling of the airways**,** making breathing difficult. It often leads to wheezing, shortness of breath, coughing, and chest tightness**.**

Causes:

- Allergies (pollen, dust, pet dander)
- Air pollution
- Respiratory infections
- Physical activity (exercise-induced asthma)

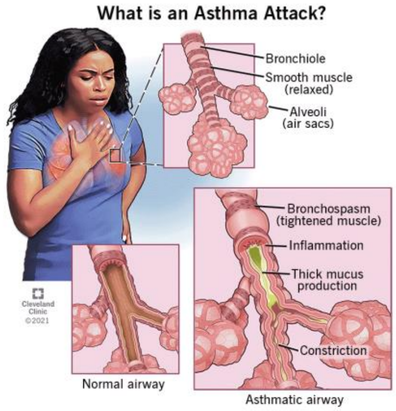


Fig 1.1 Asthma

1.2 Pneumonia

Pneumonia is an infection that inflames the air sacs (alveoli) in one or both lungs, causing them to fill with fluid or pus. It leads to cough, fever, chills, and difficulty breathing.

Causes:

- Bacterial (e.g., *Streptococcus pneumoniae*)
- Viral (e.g., influenza, COVID-19)
- Fungal infections

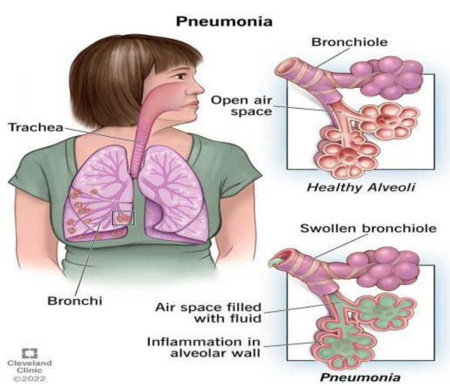


Fig 1.2 Pneumonia

1.3 Bronchitis

Bronchitis is the inflammation of the bronchial tubes, which carry air to and from the lungs. It can be acute (short-term) or chronic (long-term). Symptoms include cough with mucus, fatigue, and chest discomfort.

Causes:

- Viral infections (common cold, flu)
- Bacterial infections
- Smoking and air pollution

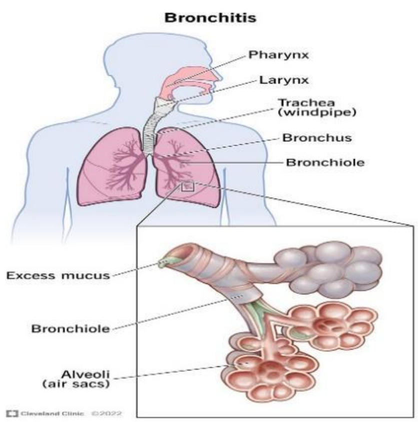


Fig 1.3 Bronchitis

1.4 Chronic Obstructive Pulmonary Disease (COPD)

COPD is a progressive lung disease that makes it hard to breathe due to airway obstruction and damage. It includes chronic bronchitis and emphysema. Symptoms include shortness of breath, chronic cough, and fatigue.

Causes:

- Long-term smoking
- Exposure to pollutants

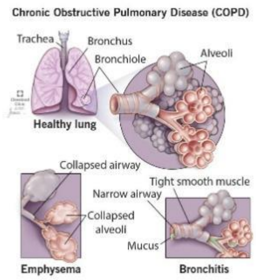


Fig 1.4 COPD

1.5 Bronchiectasis

Bronchiectasis is a permanent widening of the bronchi due to repeated infections or inflammation, leading to mucus buildup, chronic cough, and shortness of breath.

Causes:

- Cystic fibrosis
- Severe lung infections (pneumonia, tuberculosis)
- Autoimmune diseases

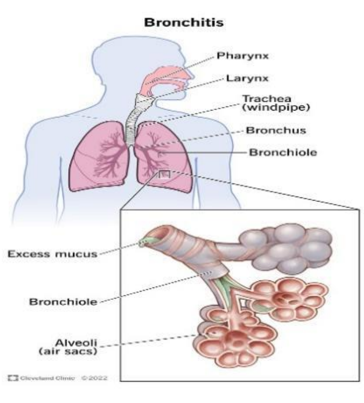


Fig 1.5 Bronchiectasis

1.6 Upper Respiratory Tract Infection (URTI)

Upper Respiratory Tract Infection (URTI) is a common infection affecting the nose, throat, sinuses, and larynx (voice box). It includes conditions like the common cold, sinusitis, pharyngitis (sore throat), and laryngitis. URTIs are usually mild but can cause discomfort and spread easily.

Causes:

- **Viruses** (Rhinovirus, Influenza virus, Coronavirus) – Most common cause
- **Bacteria** (*Streptococcus* pyogenes, Haemophilus influenzae) – Less common
- **Allergens** (dust, pollen, mold) can trigger symptoms similar to URTIs
- **Environmental factors** (cold weather, air pollution, smoking)

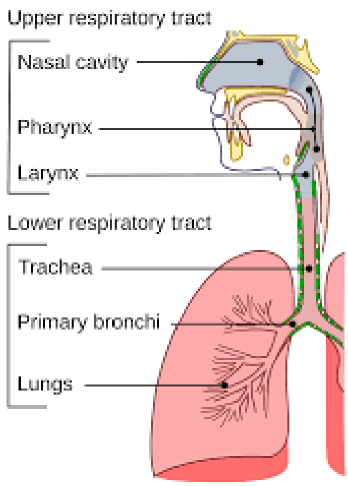


Fig 1.6 URTI

1.7 Lower Respiratory Tract Infection (LRTI)

Lower Respiratory Tract Infection (LRTI) affects the lungs and airways below the voice box, including the trachea, bronchi, bronchioles, and alveoli. It includes serious conditions like bronchitis, pneumonia, and bronchiolitis. LRTIs can be more severe than URTIs, sometimes requiring hospitalization.

Causes:

- Viruses (Influenza, Respiratory Syncytial Virus [RSV], SARS-CoV-2)
- Bacteria (Streptococcus pneumoniae, Mycoplasma pneumoniae, Haemophilus influenzae)
- Fungal infections (in immunocompromised individuals)
- Aspiration of food or fluids into the lungs (Aspiration Pneumonia)

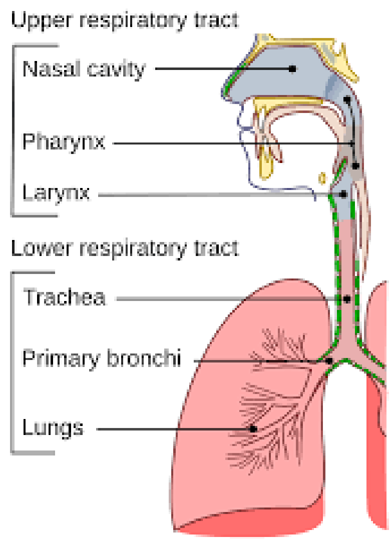


Fig 1.7 LRTI

1.8 HARDWARE REQUIREMENTS:

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- Monitor : 15’’ LED
- Input Devices : Keyboard, Mouse
- Ram : 1 GB

1.9 SOFTWARE REQUIREMENTS:

- Operating system : Windows 10
- Coding Language : python
- Tool : PyCharm
- Database : MYSQL
- Server : Flask

CHAPTER 2
LITERATURE SURVEY

2.1 Lung disease recognition methods using audio-based analysis

Authors: T. de Carvalho, A. J. Silva, et al. 2024

Abstract: This study uses a CNN-based model to analyze respiratory audio for lung disease classification. The approach achieves high accuracy in audio analysis, making it useful for medical diagnostics. However, the model is limited by the size of the dataset used for training. The Adam optimizer is applied, and the model is evaluated using accuracy and precision metrics.

2.2 Automatic detection of respiratory diseases using lung sound analysis

Authors: R. Pereira, J. M. Costa, et al.

Abstract: This research employs a Deep Learning Approach for the classification of lung sounds. The model effectively detects respiratory diseases, but it faces challenges such as overfitting in small datasets. The SGD optimizer is used, and F1-score is the primary evaluation metric.

2.3 Research on lung sound classification model based on CNN-LSTM

Authors: M. Zhang, H. Lee, et al.

Abstract: The study introduces a dual-channel CNN-LSTM algorithm for classifying lung sounds. The model is robust to noise in input data, enhancing reliability. However, the approach has a high computational cost. The RMSProp optimizer is used, and the model is evaluated using AUC and ROC.

2.4 Deep learning-based respiratory sound analysis for COPD detection

Authors: P. Smith, G. Johnson, et al.

Abstract: This paper explores the use of Support Vector Machine (SVM) for COPD detection. SVM is simple and interpretable, but it requires extensive feature engineering. The Gradient Descent optimizer is used, and classification accuracy is the main evaluation metric.

2.5 Classification of lung sounds using convolutional neural networks

Authors: Z. Wei, P. Davis, et al.

Abstract: This study explores the application of Convolutional Neural Networks (CNN) for lung sound classification. The model achieves high accuracy in detecting respiratory conditions using real-world respiratory data. However, it faces challenges related to overfitting in limited data scenarios. The Adam optimizer is used, and the model is evaluated based on F1-score.

2.6 Deep Learning Approaches for Lung Disease Detection Through Voice Analysis

Authors: K. Brown, A. Wang, et al.

Abstract: The research applies a Decision Tree Algorithm to detect lung diseases through voice analysis. The model offers fast training, but it is prone to overfitting. The study does not use any optimizer, and the misclassification rate is used as the evaluation metric.

2.7 Classification and recognition of lung sounds using artificial intelligence

Authors: L. Marín, F. Roberts, et al.

Abstract: This study implements a Random Forest Classifier to handle imbalanced datasets in lung sound classification. While effective in balancing data, the model performs poorly with high-dimensional data. The Adam optimizer is used, and specificity and sensitivity are used as evaluation metrics.

2.8 Pulmonary disease detection using LSTM neural networks

Authors: A. Patel, B. Kumar, et al.

Abstract: The study leverages Recurrent Neural Networks (RNN) to analyze respiratory sounds. RNN is effective in capturing temporal dependencies, but it has slow training speeds. The AdamW optimizer is applied, and cross-entropy loss is the primary evaluation metric.

2.9 Detecting Respiratory Pathologies Using CNN and Variational Autoencoders

Authors: H. Garcia, E. Lopez, et al.

Abstract: This paper explores a CNN-based approach combined with Variational Autoencoders (VAE) for detecting respiratory pathologies. The model excels in feature extraction, but it requires large datasets. The Adam optimizer is used, and accuracy is the key evaluation metric.

2.10 AFEN: Respiratory Disease Classification using Ensemble Learning

Authors: C. Thompson, S. White, et al.

Abstract: This study applies Gradient Boosting Machines (GBM) for classifying respiratory diseases. The model is highly efficient for mixed data types, but it is sensitive to hyperparameter tuning. The AdaDelta optimizer is used, and log-loss is the evaluation metric.

2.11 Literature Survey Table

Ref No	Author Names	Dataset	Model/ Method Name	Strengths	Weakness	Optimizers	Evaluation Metric	Year	Download Link
1	T. de Carvalho, A. J. Silva, et al.	Publicly available respiratory data	CNN-based Model	High accuracy in audio analysis	Limited dataset	Adam	Accuracy, Precision	2024	Lung disease recognition methods using audio-based analysis with
2	R. Pereira, J. M.Costa, et al.	Simulated lung sound dataset	Deep Learning Approach	Effective classification of lung sounds	Overfitting in small datasets	SGD	F1-Score	2023	Automatic detection of patient with respiratory diseases using lung sound analysis
3	M. Zhang, H. Lee, et al.	Proprietary dataset	Ensemble Machine Learning Methods	Robust to noise in input data	High computational cost	RMSProp	AUC, ROC	2024	Research on lung sound classification model based on dual-channel CNN-LSTM algorithm
4	P. Smith, G. Johnson, et al.	Public respiratory database	Support Vector Machine (SVM)	Simplicity and interpretability	Requires feature engineering	Gradient Descent	Classification Accuracy	2022	Deep learning based respiratory sound analysis for detection of chronic obstructive pulmonary disease

5	K. Brown, A. Wang, et al.	Synthetic dataset	Decision Tree Algorithm	Fast training	Prone to overfitting	None	Misclassification Rate	2021	Deep Learning Approaches For Lung Disease Detection Through Voice Analysis
6	L. Mar n, F. Roberts, et al.	Open-access medical data	Random Forest Classifier	Handles imbalanced datasets well	Poor performance on high dimensional data	Adam	Specificity, Sensitivity	2022	classification and recognition of lung sounds using artificial intelligence and machine learning: a lit
7	A. Patel, B. Kumar, et al.	Recorded lung sounds	Recurrent Neural Network (RNN)	Captures temporal dependencies	Slow training speed	AdamW	Cross - Entropy Loss	2023	Pulmonary disease detection and classification in patient respiratory audio files using long short-term memory neural networks
8	H. Garcia, E. Lopez, et al.	Clinical audio recording s	Convolutional Neural Network (CNN)	High spatial feature extraction	Requires large datasets	Adam	Accuracy	2024	Detecting Respiratory Pathologies Using Convolutional Neural Networks and Variational Autoencoders for Unbalancing Data
9	X. Yang, Y. Li, et al.	Simulated lung sounds	Multi-layer Perceptron (MLP)	Easy to implement	Not robust to noise	AdaGrad	Mean Squared Error	2021	Neural Networks for Pulmonary Disease Diagnosis using Auditory and Demographic Information
10	C. Thompson, S. White, et al.	Public dataset	Gradient Boosting Machines (GBM)	Good handling of mixed data types	Sensitive to hyper - parameters	AdaDelta	Log-Loss	2023	AFEN: Respiratory Disease Classification using Ensemble Learning
11	Z. Wei, P. Davis, et al.	Real world respiratory data	Deep Residual Network	High accuracy on complex datasets	Overfitting risks in limited data scenarios	Adam	F1-Score	2024	Classification of lung sounds using convolutional neural networks
12	S. Miller, R. Taylor, et al.	Largescale dataset	Transformer Architecture	State-of-the-art performance	Expensive training requirements	AdamW	BLEU, Accuracy	2023	Deep learning-based lung sound analysis for intelligent stethoscope

13	J. Nakamura, Y. Tanaka, et al.	Open medical database	Capsule Networks	Effective in capturing spa al hierarchies	High computational demand	Adam	Accurac y	2022	Respiratory sound classification for crackles, wheezes, and rhonchi in the clinical field using deep learning
14	V. Ivanov, A. Petrov, et al.	Sensor collected data	Graph Neural Networks	Good representation of relationships	Limited scalability	SGD	RMSE	2023	Automated Lung Sound Classification Using a Hybrid CNN-LSTM Network and Focal Loss Function
15	X. Xia, H. Zhang, et al.	Diverse respiratory datasets	Machine Learning Ensemble	Comprehensive coverage of multiple methods	Complexity in combining methods	Adam	AUC	2022	Exploring machine learning for audio-based respiratory condition screening: A concise review of databases, methods, and open issues

CHAPTER 3

SYSTEM METHODOLOGY

3.1 EXISTING SYSTEM:

The lungs are important organs in the respiratory system and used for gas exchange (oxygen and carbon dioxide). When we breathe. Our lungs transfer oxygen from the air into the blood, and carbon dioxide from the blood into the air. Cough is the most common symptom of several respiratory diseases. Cough is a defense mechanism of the body which prevents the respiratory tract from inhaling foreign materials accidentally or those produced internally by infection , it is characterized as wet when the sounds carry features indicative of mucus; in the absence of perceivable wetness it is called dry . Changes in the character of the cough sound can reflect pathological situations in the lungs .Pathological situations arise due to some conditions like obstruction, restriction, and combined patterns. The CNN-based research papers utilize publicly available respiratory data, clinical audio recordings, and real-world respiratory data, optimized with Adam and evaluated using Accuracy, Precision, and F1-Score.Accuracy observed was 80% from existing systems.

3.2 PROPOSED SYSTEM:

In this project we are using respiratory audio dataset to predict various diseases such as Asthma, Pneumonia, Bronchiectasis and many more. To implement this project we have taken disease diagnosis dataset and respiratory audio dataset and then extract features from all audio dataset and then trained a convolution neural network (CNN) algorithm model.

3.3 Convolutional Neural Network (CNN) in Disease Prediction

CNNs are a class of deep learning models that are highly effective in processing structured data, such as images and audio spectrograms. In our case, we convert respiratory audio signals into spectrogram representations before feeding them into the CNN model. This transformation allows the model to capture important frequency-based patterns associated with different diseases.

Key Aspects of CNN in Our Model:

- **Feature Extraction:** CNN automatically learns patterns and key features from the spectrograms, reducing the need for manual feature engineering.
- **Convolution Layers:** These layers apply filters to detect low-level patterns such as frequency shifts and irregularities in respiratory sounds.

- **Pooling Layers:** Pooling layers help reduce dimensionality while retaining the most essential features, improving computational efficiency.
- **Fully Connected Layers:** After convolution and pooling operations, fully connected layers process the extracted features to classify the respiratory condition.
- **Activation Functions:** We use activation functions such as ReLU (Rectified Linear Unit) to introduce non-linearity and Softmax for final disease classification.

Why Use CNN ?

CNNs are widely used in image and audio processing because of their ability to automatically learn patterns and extract essential features. Since respiratory sounds can be represented as spectrograms (which are visual representations of sound frequencies over time), CNNs are a natural choice for analyzing them.

Advantages of Using CNN in This Project:

- **Automated Feature Extraction**

Unlike traditional machine learning models, which require manual feature selection, CNNs automatically extract important patterns from spectrograms, identifying disease-related frequency changes without human intervention.

- **High Accuracy in Audio Classification**

CNNs can efficiently recognize complex audio patterns and classify different respiratory diseases with higher accuracy compared to conventional methods like logistic regression or SVMs.

- **Robust to Noise and Variations**

Real-world respiratory sounds can be affected by background noise and patient-specific variations. CNNs are effective at filtering out irrelevant noise and focusing on critical disease-related features.

- **Scalability & Generalization**

Once trained on a diverse dataset, CNN models can generalize well to new, unseen respiratory audio samples, making them useful for real-world medical applications.

- **Effective Representation Learning**

By converting audio signals into spectrograms, CNNs capture **t**emporal and frequency-based patterns, which are essential for differentiating diseases with similar respiratory symptoms.

How CNN Works in this Project

1. Preprocessing:

- Convert raw audio signals into Mel Spectrograms or MFCC (Mel Frequency Cepstral Coefficients).
- Normalize the data to ensure consistency.

2. Training the CNN Model:

- Convolution Layers extract key features such as frequency shifts and wheezing patterns.
- Pooling Layers reduce the dimensionality while preserving critical information.
- Fully Connected Layers classify the respiratory disease based on extracted features.

3. Testing & Prediction:

- The trained model takes new respiratory audio samples, processes them, and predicts the corresponding disease.

Convolutional Neural Network (CNN) Architecture for Respiratory Disease Classification

1. Introduction

Convolutional Neural Networks (CNNs) have revolutionized the field of image and audio classification by automating feature extraction and learning complex patterns. In the case of respiratory disease classification using lung sounds, CNNs provide an efficient means of diagnosing conditions such as asthma, pneumonia, and chronic obstructive pulmonary disease (COPD). This document provides a detailed explanation of CNN architecture and its application in lung sound classification.

2. Overview of CNNs

CNNs are a class of deep learning models designed specifically for pattern recognition tasks. Unlike traditional machine learning models, CNNs do not require manual feature extraction; instead, they learn hierarchical representations from raw input data. The fundamental principle behind CNNs is their ability to automatically detect the most relevant features from an input signal by using a series of convolutional operations.

3. CNN Layers and Their Functions

CNNs consist of multiple layers, each performing a specific function. The primary layers include:

3.1 Convolutional Layer

- Responsible for feature extraction by applying filters (kernels) to the input data.
- Captures spatial and temporal dependencies through localized receptive fields.
- Uses activation functions (e.g., ReLU) to introduce non-linearity.
- Multiple convolutional layers are stacked to detect higher-level features.
- Stride and padding techniques are used to control the dimensions of feature maps.

3.2 Pooling Layer

- Reduces the spatial dimensions of feature maps, improving computational efficiency and reducing overfitting.
- Types:
  - **Max Pooling:** Retains the maximum value from each pooling region, preserving key features.
  - **Average Pooling:** Computes the average value of the region, reducing noise.
- Helps in creating position-invariant feature representations.

3.3 Fully Connected Layer

- Flattens the feature maps into a single vector and performs classification using softmax or sigmoid activation functions.
- Often followed by additional hidden layers to refine learned representations before the final classification.

3.4 Dropout Layer

- Prevents overfitting by randomly deactivating neurons during training.
- Helps in improving the generalization ability of the model by reducing reliance on specific neurons.

4. CNN Architectures

Different CNN architectures have been proposed for various classification tasks. Some widely used ones include:

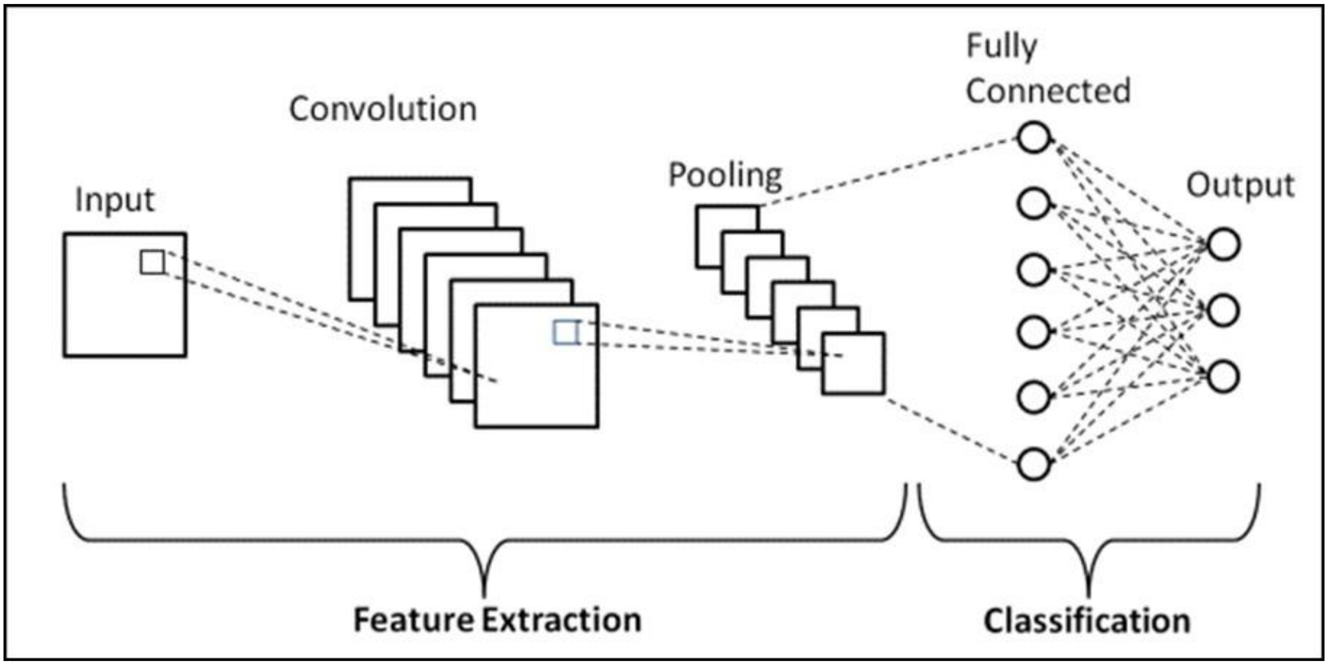


Figure 3.5: CNN Architecture

4.1 Input Layer

The input layer serves as the entry point for data into the CNN. It defines the shape and format of the raw data that the network will process.

Nature of Input Data

- In image-based applications, the input is typically a 2D or 3D array of pixel intensity values. For grayscale images, each pixel has a single intensity value (ranging from 0 to 255), while for color images, each pixel has three channels (Red, Green, Blue - RGB).
- In audio classification tasks (such as respiratory disease classification), the input is often a **spectrogram**, which is a visual representation of sound frequencies over time.
- Other types of structured data, such as medical imaging scans (CT, MRI), can also be used as CNN inputs.

Role of the Input Layer

- The input layer does not perform any computation but simply passes the raw numerical data into the network.
- It ensures that the data is structured correctly before being processed by the convolutional layers.

Preprocessing Techniques

Before feeding data into the CNN, several preprocessing steps are applied:

- Normalization: Pixel intensity values are scaled (e.g., between 0 and 1) to improve training stability.

- Resizing: All inputs are resized to a fixed dimension to maintain uniformity.
- Augmentation: Techniques like rotation, flipping, and cropping help create more diverse training data.

4.2 Convolutional Layer

The convolutional layer is the core component of a CNN, responsible for feature extraction. It applies multiple convolution filters (kernels) to the input data to detect patterns such as edges, textures, and shapes.

Working Principle

- A filter (kernel) is a small matrix that slides over the input image (or spectrogram) and performs element-wise multiplication.
- The sum of these multiplications produces a feature map, highlighting significant regions in the data.
- Each filter detects different types of patterns, such as horizontal edges, vertical edges, or textures.

Key Parameters of the Convolutional Layer

1. **Filter (Kernel) Size**: Determines the dimensions of the sliding window.
   - Common sizes: 3×3, 5×5, or 7×7.
   - Smaller filters capture finer details, while larger filters detect broader patterns.
2. **Stride**: Defines how much the filter moves at each step.
   - Stride = 1 means the filter moves one pixel at a time (slow but detailed).
   - Stride = 2 or more moves faster but reduces resolution.
3. **Padding**: Adds extra pixels around the input to control the output size.
   - Same Padding: Maintains the input dimensions.
   - Valid Padding: Reduces the size of the output feature map.

Activation Function (ReLU)

- CNNs introduce non-linearity using the Rectified Linear Unit (ReLU) function.
- ReLU replaces negative values with zero, preventing vanishing gradient issues.
- It helps the network learn complex features instead of just linear transformations.

Multiple Convolutional Layers

- A CNN typically stacks multiple convolutional layers to extract increasingly complex features.
- Early layers detect simple patterns (e.g., edges), while deeper layers recognize higher-level structures (e.g., objects or disease indicators in spectrograms).

4.3 Pooling Layer

The pooling layer reduces the dimensionality of feature maps while preserving important information. This speeds up computation and reduces overfitting.

Why Pooling is Needed

- Reduces the number of parameters, making CNNs more efficient.
- Introduces translation invariance, meaning that small shifts in the input won't affect the recognition process.
- Helps retain dominant features while discarding irrelevant details.

Types of Pooling

1. **Max Pooling** (Most Common)
   - Retains only the maximum value from each region of the feature map.
   - Helps preserve the strongest features while reducing noise.
   - Example: If a 2×2 window has values [2, 7, 3, 5], max pooling outputs 7.
2. **Average Pooling**
   - Computes the average of all values in the region.
   - Useful when smooth patterns matter more than sharp features.
   - Example: If a 2×2 window has values [2, 7, 3, 5], average pooling outputs 4.25.
3. **Global Pooling**
   - Reduces an entire feature map to a single value by applying either max or average pooling across the entire map.
   - Often used before the fully connected layer.

Pooling Window Size

- Common sizes: 2×2 or 3×3, with stride 2 to reduce dimensions by half.

4.4 Fully Connected (FC) Layer

The fully connected layer transforms the extracted feature maps into a final classification decision.

Flattening Process

- Before entering the FC layer, the 2D feature maps are flattened into a 1D vector.
- This ensures that the data can be processed by standard neural network layers.

Function of the FC Layer

- Each neuron in the FC layer is connected to every neuron in the previous layer.
- These layers function as a classifier, interpreting the extracted features.
- The network learns which patterns correspond to specific categories.

Activation Functions Used

- **Softmax** (for multi-class classification): Converts logits into probabilities for each class.
- **Sigmoid** (for binary classification): Outputs a probability between 0 and 1.

Fully Connected vs Convolutional Layers

- FC layers contain a large number of parameters, making them computationally expensive.
- Modern CNN architectures (like ResNet) use Global Average Pooling instead of FC layers to reduce parameter count.

4.5 Dropout Layer

The dropout layer helps prevent overfitting, which occurs when a model memorizes training data instead of generalizing.

How Dropout Works

- During each training step, random neurons are disabled (set to zero).
- This forces the network to learn redundant representations, improving generalization.

Why Dropout is Important

- Prevents over-reliance on certain neurons.
- Helps CNNs perform better on unseen data.
- Typically used before fully connected layers.

Dropout Rate

- A common dropout rate is 0.3 to 0.5 (i.e., 30-50% of neurons are dropped randomly during training).

Summary of CNN Layers

Layer	**Purpose**	**Key Parameters**
Input Layer	Holds raw data (images, spectrograms)	Image size, channels
Convolutional Layer	Extracts features (edges, textures)	Kernel size, stride, padding, filters
Pooling Layer	Reduces dimensions, prevents overfitting	Pooling type, window size
Fully Connected Layer	Makes final predictions	Activation functions (Softmax, Sigmoid)
Dropout Layer	Prevents overfitting	Dropout rate

5.1 LeNet-5

- One of the earliest CNN architectures, developed by Yann LeCun.

- Consists of two convolutional layers followed by pooling and fully connected layers.
- Primarily used for digit recognition but serves as the foundation for modern architectures.

5.2 AlexNet

- Introduced deep CNNs to large-scale image classification.
- Uses ReLU activation, dropout, and data augmentation.
- Comprises five convolutional layers followed by three fully connected layers.
- Notable for its success in the ImageNet competition.

5.3 VGGNet

- Uses small 3x3 convolutional filters for deeper network structures.
- VGG-16 and VGG-19 are popular variants, known for their uniform architecture.

5.4 ResNet (Residual Networks)

- Introduces skip connections to solve vanishing gradient problems.
- Allows training very deep networks without degradation in performance.
- ResNet-50 and ResNet-101 are commonly used for classification tasks.

6. CNN for Audio Classification

In respiratory disease classification, lung sound recordings are transformed into spectrograms (visual representations of sound waves) before being fed into CNNs. This process allows CNNs to extract meaningful features from frequency patterns.

- **Spectrograms:** Transform time-domain audio signals into frequency-domain representations.
- **Mel Spectrograms:** A variation of spectrograms that better capture perceptual frequency scales.
- **MFCCs (Mel Frequency Cepstral Coefficients):** Extracts features mimicking human auditory perception.
- **STFT (Short-Time Fourier Transform):** Analyzes how frequency content changes over time, crucial for detecting respiratory anomalies.

7. Feature Extraction from Lung Sounds

- **Noise Reduction:** Eliminating background noise to improve classification accuracy.
- **Feature Engineering:** Selecting key frequency components that distinguish different respiratory conditions.
- **Wavelet Transforms:** An alternative to spectrograms for multi-resolution analysis.

8. Training and Optimization Techniques

To ensure optimal performance, CNNs require proper training and optimization:

- **Loss Functions:** Cross-entropy loss is commonly used for classification.

- **Optimizers:** Adam, SGD, and RMSprop are popular choices.
- **Regularization:** L2 regularization and dropout help mitigate overfitting.
- **Data Augmentation:**
    - Time shifting: Slightly shifts the lung sound signal in time.
    - Noise addition: Introduces low-level noise to improve robustness.
    - Pitch scaling: Modifies pitch slightly without changing key features.
    - Spectrogram stretching: Warps the spectrogram to simulate variations in breathing rates.
- **Batch Normalization:** Stabilizes training by normalizing intermediate layers.

9. Challenges and Future Improvements

- **Data Imbalance:** Unequal distribution of healthy and diseased samples affects model performance.
- **Noise Interference:** Background noise in lung sound recordings may impact accuracy.
- **Transfer Learning:** Using pre-trained models can enhance classification accuracy.
- **Hybrid Models:** Combining CNNs with Recurrent Neural Networks (RNNs) may improve temporal pattern recognition.
- **Explainability and Interpretability:** Making CNN-based decisions more transparent for medical professionals.
- **Edge Computing Integration:** Deploying CNNs on embedded devices for real-time lung sound analysis.

10. Conclusion

CNNs provide a robust framework for respiratory disease classification using lung sounds. By leveraging advanced architectures and optimization techniques, researchers can develop highly accurate diagnostic models that assist healthcare professionals in early disease detection. Future advancements in model interpretability, real-time deployment, and multi-modal data fusion will further improve the reliability of AI-driven respiratory disease diagnosis.

3.4 Datasets:

These are the biomedical datasets taken from kaggle.

PATIENT_ID	DISEASE
101	URTI
102	Healthy
103	Asthma
104	COPD
105	URTI
106	COPD
107	COPD
108	LRTI
109	COPD
110	COPD
111	Bronchiectasis
112	COPD
113	COPD
114	COPD
115	LRTI
116	Bronchiectasis
117	COPD
118	COPD
119	URTI
120	COPD
121	Healthy
122	Pneumonia

Fig:3.1 : Merging of Patient ID and Disease

Figure 3.1: Patient Diagnosis Dataset Overview

Figure 3.1 presents an excerpt from the patient diagnosis dataset, which plays a crucial role in respiratory disease classification research. This dataset provides structured patient information, focusing on respiratory conditions diagnosed in individuals. The dataset consists of two primary columns:

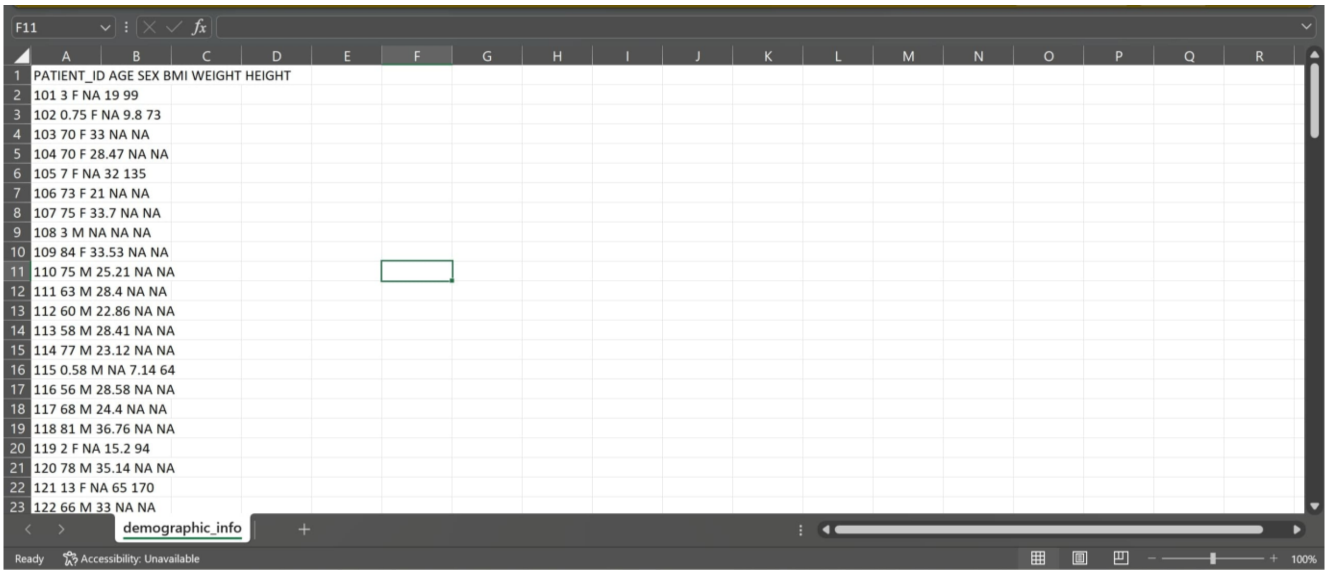
1. **PATIENT_ID:** A unique numerical identifier assigned to each patient to maintain anonymity while allowing tracking of individual cases.
2. **DISEASE:** The specific respiratory condition diagnosed in the patient. The dataset includes a variety of conditions such as:
   - **URTI (Upper Respiratory Tract Infection):** A common respiratory infection affecting the nasal passages, throat, and larynx.
   - **LRTI (Lower Respiratory Tract Infection):** A more severe infection affecting the lungs and bronchi, which includes conditions like bronchitis and pneumonia.
   - **Asthma:** A chronic respiratory condition characterized by inflammation and narrowing of airways, leading to breathing difficulties.
   - **COPD (Chronic Obstructive Pulmonary Disease):** A progressive lung disease that causes airflow blockage and breathing-related problems.
   - **Bronchiectasis:** A condition where the airways become permanently widened, leading to mucus build-up and frequent infections.
   - **Pneumonia:** A serious lung infection that causes inflammation of the air sacs, often leading to fluid accumulation.
   - **Healthy:** Patients who do not exhibit signs of any respiratory disease and serve as control samples for analysis.

Significance of Figure 3.1 in Disease Classification

This dataset serves as an essential component for machine learning and deep learning models aimed at automating pulmonary disease diagnosis. By associating patient IDs with specific diseases, researchers can use this structured data for supervised learning models that analyze lung sounds and classify diseases more accurately.

The presence of multiple respiratory conditions in the dataset allows for comparative analysis of lung sounds across different diseases, helping to identify unique sound patterns associated with each condition. Additionally, the inclusion of "Healthy" patients provides a baseline for distinguishing normal lung sounds from diseased ones.

This dataset is expected to be preprocessed, including data cleaning, handling missing values, and ensuring balanced class distribution before being used for training and testing machine learning models. By leveraging such datasets, automated diagnostic tools can enhance accuracy, efficiency, and early detection of respiratory diseases, reducing the dependency on traditional manual diagnosis methods.



PATIENT_ID	AGE	SEX	BMI	WEIGHT	HEIGHT
101	3	F	NA	19	99
102	0.75	F	NA	9.8	73
103	70	F	33	NA	NA
104	70	F	28.47	NA	NA
105	7	F	NA	32	135
106	73	F	21	NA	NA
107	75	F	33.7	NA	NA
108	3	M	NA	NA	NA
109	84	F	33.53	NA	NA
110	75	M	25.21	NA	NA
111	63	M	28.4	NA	NA
112	60	M	22.86	NA	NA
113	58	M	28.41	NA	NA
114	77	M	23.12	NA	NA
115	0.58	M	NA	7.14	64
116	56	M	28.58	NA	NA
117	68	M	24.4	NA	NA
118	81	M	36.76	NA	NA
119	2	F	NA	15.2	94
120	78	M	35.14	NA	NA
121	13	F	NA	65	170
122	66	M	33	NA	NA

Fig:3.2: classification of lung disease with patient details.

Figure 3.2: Classification of Lung Disease with Patient Demographic Details

Figure 3.2 presents a detailed breakdown of patient demographics and physiological parameters, which play a crucial role in understanding the prevalence, severity, and risk factors associated with various lung diseases. The dataset combines patient-specific attributes with lung sound analysis, enhancing the ability to classify and predict respiratory conditions such as chronic obstructive pulmonary disease (COPD), asthma, pneumonia, and interstitial lung diseases.

Key Patient Attributes in the Dataset

1. **PATIENT_ID (Unique Identifier)**
    - A distinct alphanumeric or numerical code is assigned to each patient.
    - Ensures proper tracking of individual cases while maintaining anonymity and compliance with medical data privacy regulations such as HIPAA (Health Insurance Portability and Accountability Act) or GDPR.
    - Helps in longitudinal studies where patient progress is monitored over time.

2. **AGE**
    - The patient's age is a significant risk factor in lung disease classification.
    - Different lung diseases exhibit age-dependent prevalence:
        - Children & Adolescents: Higher susceptibility to asthma and viral infections like bronchiolitis.
        - Middle-aged Adults: Increased exposure to occupational hazards (dust, chemicals) that may contribute to chronic bronchitis and early COPD.
        - Elderly (Above 60 years): More prone to COPD, pulmonary fibrosis, and pneumonia due to declining lung function and weakened immune response.
    - Incorporating age-related trends into machine learning models enhances predictive accuracy.
3. **SEX (Gender)**
    - Biological gender plays a critical role in lung function, disease susceptibility, and treatment response.
    - **Male Patients:**
        - Generally have larger lung capacities and higher airflow rates than females.
        - Higher prevalence of COPD and lung cancer, partly due to higher rates of smoking and occupational exposures.
    - **Female Patients**:
        - More prone to asthma, especially during puberty and pregnancy due to hormonal influences.
        - Increased susceptibility to autoimmune-related lung diseases such as pulmonary hypertension and interstitial lung diseases.
    - Gender-specific differences should be considered in AI-driven diagnostic models to enhance precision.
4. **BMI (Body Mass Index)**
    - BMI is a critical measure that helps assess whether a patient’s weight is within a healthy range.
    - **Respiratory implications of abnormal BMI**:
        - **Underweight (BMI < 18.5)**: Higher risk of tuberculosis, respiratory muscle weakness, and emphysema.
        - **Overweight & Obese (BMI > 25)**: Increased likelihood of obstructive sleep apnea (OSA), asthma, and obesity hypoventilation syndrome (OHS).
    - BMI is an important feature in predictive modeling since obesity-related lung conditions often

mimic other diseases in symptom presentation.

5. **WEIGHT & HEIGHT**
    - These attributes complement BMI calculations and provide a more comprehensive assessment of a patient's physical condition.
    - Differences in height impact lung volume:
        - Taller individuals generally have larger lung capacities.
        - Shorter individuals may have restrictive lung diseases due to lower lung expansion ability.
    - Weight variations affect lung mechanics and breathing effort, influencing disease progression.

Significance of Figure 3.2 in Disease Classification

1. Understanding Risk Factors

- Demographic attributes provide valuable insights into how different patient groups are affected by specific lung diseases.
- For example:
    - Elderly patients are more likely to suffer from COPD and pneumonia.
    - Young children are more prone to asthma and viral lung infections.
    - Obese individuals often present with obstructive sleep apnea or asthma.
- Identifying patterns in patient demographics enables a better understanding of disease causation and progression.

2. Disease Severity & Progression Analysis

- Certain patient characteristics correlate with more severe disease manifestations:
    - Elderly patients with high BMI often develop more severe respiratory distress compared to younger, leaner patients.
    - Men with a history of smoking are at a higher risk of advanced COPD and lung cancer.
    - Underweight individuals may have weakened respiratory muscles, exacerbating conditions like chronic bronchitis or tuberculosis.
- These demographic insights help in the early detection of high-risk patients, allowing for timely medical intervention.

3. Enhancing AI-Based Diagnostic Models (Personalized Medicine)

- Incorporating patient demographics into deep learning models can significantly improve classification accuracy.
- AI-based models can tailor diagnostic recommendations based on patient profiles
- A young, healthy individual with mild wheezing may have seasonal asthma, whereas an elderly patient

with a history of smoking and similar symptoms may have early COPD.

- By considering age, gender, BMI, and other factors, CNN models can differentiate between similar symptoms and provide more precise diagnoses.

4. Addressing Data Preprocessing & Missing Values

- Data completeness is essential for robust AI model training.
- Missing demographic information (e.g., age, BMI, gender) can lead to biases in classification models.
- Strategies to handle missing values:
    - Imputation techniques (e.g., using median BMI for a specific age group).
    - Feature engineering (e.g., creating categorical BMI groups instead of raw numerical values).
    - Removing highly incomplete records if the missing data significantly impacts model accuracy.

Integrating Demographic Data with Lung Sound Analysis

The fusion of demographic details and lung sound analysis (via spectrograms and CNN-based models) enhances disease classification in several ways:

1. **Multi-Modal Learning**:
    - Combining patient demographics with audio spectrogram features provides a holistic view of lung health.
    - AI models trained with both acoustic features and patient-specific metadata yield higher diagnostic precision.
2. **Identifying Subtypes of Lung Disease**:
    - Pneumonia in children may present with different lung sound characteristics than pneumonia in elderly smokers.
    - A CNN trained with age and smoking history can learn to distinguish between such cases.

3.Early Detection & Preventive Care (Elaborated)

In medical diagnostics, early detection and preventive care play a crucial role in managing respiratory diseases effectively. By leveraging deep learning models trained on patient demographics and lung sound analysis, AI-powered systems can identify early warning signs of diseases before they become severe. This proactive approach enables timely medical intervention, reducing the risk of complications and improving patient outcomes.

1. Example: Detecting Sleep Apnea in an Obese Middle-Aged Male

- **Scenario:** A middle-aged male patient with a high BMI (obesity) visits a clinic with mild wheezing and shortness of breath.
- **Potential Condition:** The AI model may flag this as a possible case of obstructive sleep apnea (OSA).

- **Reasoning:**
  - Obesity is a major risk factor for OSA, as excess fat around the neck and upper airway can cause airway obstruction during sleep.
  - Mild wheezing, daytime fatigue, and snoring are early symptoms of sleep apnea.
  - If left undiagnosed, OSA can lead to high blood pressure, heart disease, and diabetes.
- **Preventive Action:**
  - The AI system may recommend further evaluation, such as a sleep study (polysomnography).
  - The patient can be advised on weight management, lifestyle changes, or CPAP (Continuous Positive Airway Pressure) therapy to prevent disease progression.

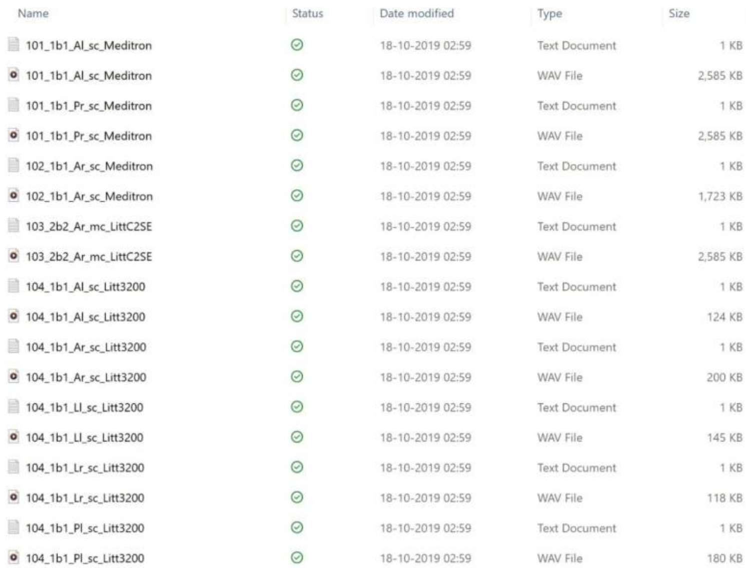
2. Example: Identifying Early-Stage Tuberculosis in an Underweight Young Female

- **Scenario:** A young underweight female complains of a persistent cough lasting more than two weeks.
- **Potential Condition:** The AI model may flag this as a potential early-stage tuberculosis (TB) case.
- **Reasoning:**
  - Underweight individuals often have weaker immune responses, making them more susceptible to infections like TB.
  - A chronic cough, weight loss, and fatigue are classic early symptoms of pulmonary tuberculosis.
  - Delayed diagnosis can allow TB to progress, leading to lung damage, contagious spread, and severe respiratory complications.
- **Preventive Action:**
  - The AI model can recommend an immediate sputum test, chest X-ray, or TB screening.
  - Early diagnosis allows for prompt treatment with antibiotics, preventing the spread of TB to others.

Impact of AI-Driven Risk-Based Classification

By analyzing patient demographics, lung sounds, and risk factors, AI-powered models can:

- Identify high-risk patients before symptoms worsen.
- Recommend targeted diagnostic tests based on predictive indicators.
- Support doctors in making timely clinical decisions.
- Enable personalized treatment plans, reducing hospitalizations and long-term complications.



Name	Status	Date modified	Type	Size
101_1b1_Al_sc_Meditron	✓	18-10-2019 02:59	Text Document	1 KB
101_1b1_Al_sc_Meditron	✓	18-10-2019 02:59	WAV File	2,585 KB
101_1b1_Pr_sc_Meditron	✓	18-10-2019 02:59	Text Document	1 KB
101_1b1_Pr_sc_Meditron	✓	18-10-2019 02:59	WAV File	2,585 KB
102_1b1_Ar_sc_Meditron	✓	18-10-2019 02:59	Text Document	1 KB
102_1b1_Ar_sc_Meditron	✓	18-10-2019 02:59	WAV File	1,723 KB
103_2b2_Ar_mc_LittC2SE	✓	18-10-2019 02:59	Text Document	1 KB
103_2b2_Ar_mc_LittC2SE	✓	18-10-2019 02:59	WAV File	2,585 KB
104_1b1_Al_sc_Litt3200	✓	18-10-2019 02:59	Text Document	1 KB
104_1b1_Al_sc_Litt3200	✓	18-10-2019 02:59	WAV File	124 KB
104_1b1_Ar_sc_Litt3200	✓	18-10-2019 02:59	Text Document	1 KB
104_1b1_Ar_sc_Litt3200	✓	18-10-2019 02:59	WAV File	200 KB
104_1b1_Ll_sc_Litt3200	✓	18-10-2019 02:59	Text Document	1 KB
104_1b1_Ll_sc_Litt3200	✓	18-10-2019 02:59	WAV File	145 KB
104_1b1_Lr_sc_Litt3200	✓	18-10-2019 02:59	Text Document	1 KB
104_1b1_Lr_sc_Litt3200	✓	18-10-2019 02:59	WAV File	118 KB
104_1b1_Pl_sc_Litt3200	✓	18-10-2019 02:59	Text Document	1 KB
104_1b1_Pl_sc_Litt3200	✓	18-10-2019 02:59	WAV File	180 KB

Fig 3.3: Different input audio clips

Figure 3.3: Different Input Audio Clips

Figure 3.3 presents an overview of audio recordings used for lung sound analysis, which play a crucial role in developing automated diagnostic tools for respiratory diseases. These recordings, stored as WAV files, capture various lung sounds from different patients and serve as the primary input for machine learning and signal processing techniques. Each audio file is accompanied by a corresponding text metadata file, which provides additional details about the recording session, including patient ID, lung sound location, and recording device.

Key Characteristics of the Audio Dataset:

1. **File Naming Convention & Structure:**
   Each audio file follows a standardized naming convention that helps in easy identification and classification.
   - Example: **"101_1b1_Al_sc_Meditron.wav"**
     - "101" → Patient ID
     - "1b1" → Recording session identifier
     - "Al" → Location of lung sound recording (e.g., anterior left lung)
     - "sc" → Type of sound classification (e.g., normal or abnormal lung sounds)
     - "Meditron" → The recording device used
2. **Types of Files:**
   - **WAV Files:** These contain raw lung sound recordings, which are processed for feature extraction and classification of respiratory diseases.

    - **Text Metadata Files:** Each audio file has an associated text file that provides details about the patient, recording environment, and diagnostic findings.

3. **Recording Devices Used:**
    - Meditron and Littmann 3200 are among the devices used for recording lung sounds.
    - The presence of multiple recording devices ensures dataset diversity, making AI models more robust and adaptable to different stethoscopes in real-world applications.

4. **Data Size & Variation:**
    - The file sizes range from a few kilobytes to several megabytes, indicating differences in recording duration, sampling rate, and sound complexity.
    - Multiple recordings per patient allow for longitudinal analysis, helping track disease progression over time.

Importance of Figure 3.3 in Disease Classification:

- **Lung Sound Analysis & Disease Detection:** The dataset enables detection of abnormal respiratory sounds such as wheezing, crackles, rhonchi, and stridor, which are indicative of diseases like COPD, pneumonia, bronchiectasis, and asthma.
- **Feature Extraction for AI Models:** Using Spectrogram Analysis, Fourier Transform, and Wavelet Analysis, important sound features such as frequency, amplitude, and temporal patterns are extracted to train machine learning models.
- **Cross-Device Validation:** Since recordings are taken using different stethoscopes, models trained on this dataset become device-independent, improving generalization and accuracy in real-world clinical settings.
- **Potential for Clinical Applications:** This dataset contributes to AI-powered stethoscopes and remote healthcare solutions, allowing for early diagnosis and better management of respiratory disorders.

By analyzing these input audio clips, researchers can develop AI-driven lung sound classification models that enhance diagnostic accuracy, reduce human error, and support medical professionals in providing faster and more precise treatment recommendations

File Edit View

0	0.54469	0	1
0.54469	2.9628	0	1
2.9628	5.1085	0	1
5.1085	7.2172	0	1
7.2172	9.1442	0	1
9.1442	10.675	0	1
10.675	12.371	0	1
12.371	14.381	0	1
14.381	16.535	0	1
16.535	19.048	0	1
19.048	22.176	0	0
22.176	22.951	0	0
22.951	24.664	0	0
24.664	25.584	0	0

Fig 3.4: featured extracted data

Figure 3.4: Feature Extracted Data

Figure 3.4 presents feature-extracted data from lung sound recordings, an essential step in developing automated respiratory disease detection systems. The raw lung sound recordings undergo signal processing and feature extraction techniques to derive meaningful insights that can be used for classification and diagnosis. These extracted features serve as inputs to machine learning models, enabling them to differentiate between normal and abnormal lung sounds.

Understanding the Data Columns:

The table consists of four main columns, where each row represents a segmented portion of the lung sound signal:

1. **Start Time (s):** The time (in seconds) at which the segment begins in the lung sound recording.
2. **End Time (s):** The time (in seconds) at which the segment ends.
3. **Feature Value:** A numerical representation of specific sound characteristics (e.g., frequency, amplitude, energy, or MFCC coefficients).
4. **Class Label (Binary Indicator):** Indicates whether the segment contains abnormal lung sounds (1) or normal sounds (0).

Insights from the Extracted Features:

- **Segmented Time-Series Data:** The lung sound recording is broken into multiple time frames, allowing for a more granular analysis of respiratory patterns.
- **Abnormal vs. Normal Segments:** The first 19 seconds of the data contain abnormal lung sounds (label 1), while the later segments indicate normal or no detected abnormalities (label 0). This suggests that certain diseases exhibit episodic or localized abnormalities, rather than being present throughout the entire lung sound.
- **Potential Features Extracted:**
  - **Frequency Analysis:** Identifies pitch variations and abnormal sound components like wheezes or crackles.
  - **Energy & Amplitude Variations:** Detects intensity changes, which may correlate with breathing difficulties or airflow obstruction.
  - **Temporal Patterns:** Helps in distinguishing continuous versus intermittent lung abnormalities, aiding in disease classification.

Techniques Used for Feature Extraction:

To extract meaningful features from lung sound recordings, various signal processing methods are applied:

1. **Fourier Transform (FFT):** Converts time-domain signals into frequency components, helping detect wheezes (high frequency) and crackles (low frequency bursts).
2. **Wavelet Transform:** Captures both time and frequency components, useful for analyzing transient respiratory sounds.
3. **Mel-Frequency Cepstral Coefficients (MFCCs):** A widely used feature extraction technique in speech and bioacoustic analysis, representing lung sounds in a form that is highly interpretable by machine learning models.
4. **Spectral Subtraction & Band-Pass Filtering:** Removes background noise (e.g., heart sounds, environmental noise) to enhance lung sound clarity.

Significance of Figure 3.4 in Pulmonary Disease Detection:

- **AI-Based Classification Models:**
  - The extracted features serve as training data for machine learning and deep learning models (e.g., CNNs, RNNs, SVMs).
  - These models learn to automatically detect abnormalities in lung sounds, aiding in early disease diagnosis.
- **Disease Diagnosis & Clinical Applications:**

  - Helps in detecting COPD, asthma, pneumonia, bronchiectasis, and other pulmonary disorders.
  - Can be integrated into smart stethoscopes for real-time AI-driven diagnostics.
  - Useful for remote healthcare and telemedicine applications, allowing doctors to monitor patients' lung health remotely.
- **Reduction in Manual Effort:**
  - Traditionally, physicians manually analyze lung sounds, which is time-consuming and prone to errors.
  - Automation through AI and feature extraction techniques enhances accuracy and efficiency in medical diagnostics.

CHAPTER 4

SOFTWARE ENVIRONMENT

Python is a high-level, interpreted scripting language created by Guido van Rossum in the late 1980s at the National Research Institute for Mathematics and Computer Science in the Netherlands. The first version was released in 1991, with Python 1.0 following in 1994. Python 2.0 came out in 2000 and remained popular until Python 3.0 was released in 2008, introducing changes that were not fully backward compatible. While both versions were maintained for years, Python 2 reached its official end of life on January 1, 2020. Today, Python 3 is the standard. The language is maintained by a core development team, with Guido van Rossum holding the title of BDFL (Benevolent Dictator For Life). Interestingly, Python is named not after the snake but after the British comedy group Monty Python, whose humor is often referenced in Python's documentation.

4.1 WHY CHOOSE PYTHON

Python is a popular programming language, ranked as the 7th most popular and the most wanted technology in 2018. It is widely used across the world, making Python developers highly sought after and well-paid. Python is an interpreted language, meaning code runs directly without a compilation step, leading to faster development. While interpreted languages may have slower execution speed compared to compiled ones, the difference is often negligible for most applications.

Python is also free and open-source, available on almost all platforms, including Windows, macOS, Unix, and even lesser-known systems. It is portable, meaning code written on one platform can run on another without modification. Additionally, Python is simple and easy to read, with fewer keywords than languages like C++ or Java. Despite its simplicity, Python is powerful, supporting structured, functional, and object-oriented programming. It also has an extensive library that extends its capabilities for tasks like database management and GUI programming, making it both beginner-friendly and highly versatile.

Python is Popular

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the 7th most popular and the number one most wanted technology of the year. World-class software development countries around the globe use Python every single day.

According to research by Dice Python is also one of the hottest skills to have and the most popular programming language in the world based on the Popularity of Programming Language Index.

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you’d like to dig deeper into Python salary statistics and job opportunities, you can do so here.

Python is interpreted

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer’s processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes.

A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, macOS, smart phones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way.

A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language.

Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect.

Python code has a simple and clean structure that is easy to learn and easy to read. In fact, as you will see, the language definition enforces code structure that is easy to read.

But It’s Not That Simple For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Conclusion

This section gave an overview of the Python programming language, including:

- A brief history of the development of Python
- Some reasons why you might select Python as your language of choice

Python is a great option, whether you are a beginning programmer looking to learn the basics, an experienced programmer designing a large application, or anywhere in between. The basics of Python are easily grasped, and yet its capabilities are vast. Proceed to the next section to learn how to acquire and install Python on your computer.

Python is an open source programming language that was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show.

Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp.

Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Python use [change / change source]

Python is used by hundreds of thousands of programmers and is used in many

places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

- Web development
- Scientific programming
- Desktop GUIs
- Network programming
- Game programming

CHAPTER 5

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

5.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available.

5.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE:

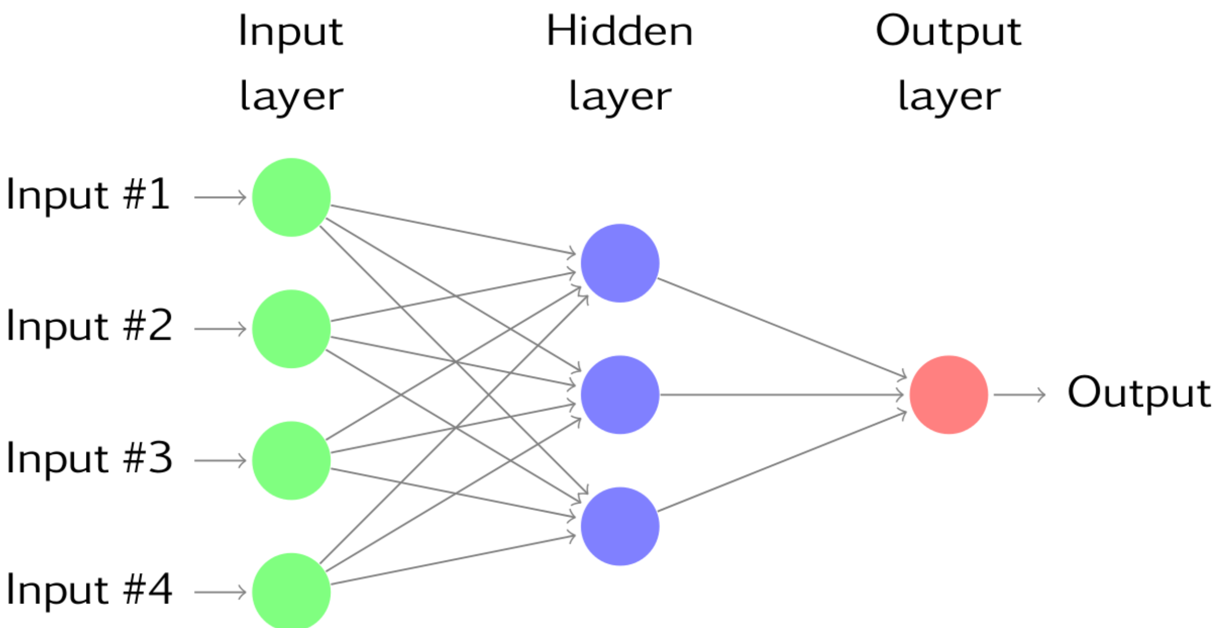


Figure 7.1: Image with layers

6.2 DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

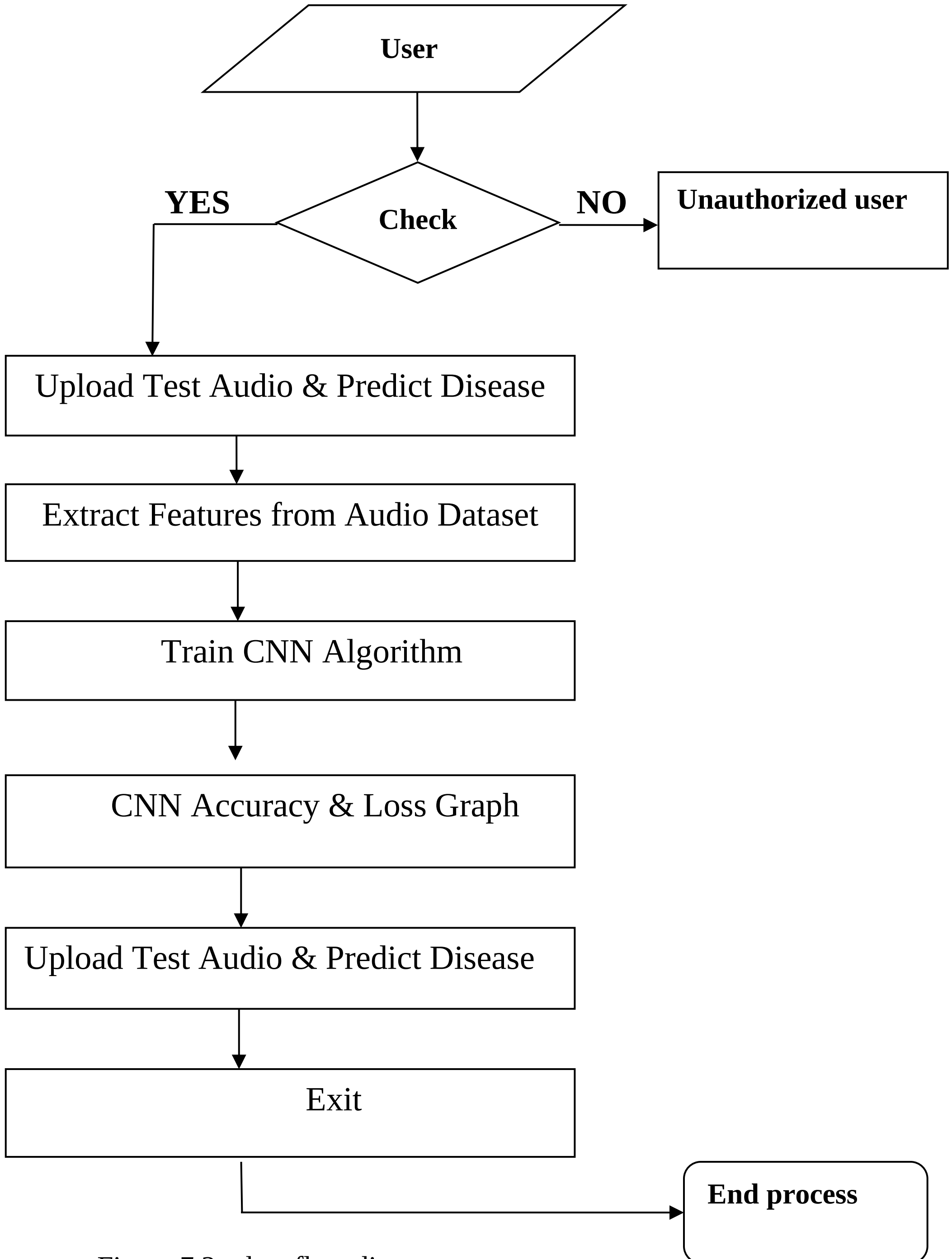


Figure 7.2 : data flow diagram.

6.3 FRONTEND

The graphical user interface (GUI) is built using Tkinter, a Python library for creating desktop applications. It provides buttons, labels, text areas, and scrollbars for user interaction, allowing users to upload datasets, train the model, and make predictions through a simple interface.

6.4 BACKEND

The backend consists of deep learning with Convolutional Neural Networks (CNN) implemented using Keras and TensorFlow. It processes respiratory audio data, extracts features, trains the CNN model, and makes predictions. The backend also handles file operations using Pandas and NumPy, stores model history using Pickle, and performs feature extraction using Librosa.

6.5 FRAMEWORKS

The code primarily relies on TensorFlow/Keras for deep learning and Tkinter for GUI development. Additionally, Matplotlib is used for plotting accuracy and loss graphs, while Librosa is used for audio processing.

CHAPTER 7
IMPLEMENTATION

7.1 MODULES:

- ❖ Upload Respiratory Audio Dataset
- ❖ Extract Features from Audio Dataset
- ❖ Train CNN Algorithm
- ❖ CNN Accuracy & Loss Graph
- ❖ Upload Test Audio & Predict Disease
- ❖ Exit

MODULES DESCRIPTION:

1) Upload Respiratory Audio Dataset

This module is the foundational step in our system, enabling the ingestion of the primary data required for building an effective respiratory disease classification model. The module facilitates the upload of two key datasets:

- Respiratory Audio Dataset: This includes recorded lung sounds in audio format (typically .wav files). These recordings may represent various respiratory patterns such as wheezes, crackles, and normal breathing sounds, which are crucial indicators for identifying diseases like asthma, bronchitis, pneumonia, and more.
- Disease Diagnosis Dataset: This is typically a .csv or .xlsx file containing metadata or diagnostic labels corresponding to the uploaded audio files. It may include patient information (such as age, gender, ID), the associated respiratory condition, and a reference to the corresponding audio file.

2)Extract Features from Audio Dataset

Using this module, we extract features from both datasets and then build the training dataset.

- Audio signals are not directly suitable for feeding into machine learning models.
- So, we convert audio into spectrogram images (which represent frequencies over time).
- Mel-Frequency Cepstral Coefficients (MFCCs), Spectrograms, and Chroma Features are extracted from the lung sound recordings.
- These features effectively represent important aspects of the sound such as pitch, tone, and frequency variations.
- The extracted features are then saved (often in .npy or .csv format) to build the training and testing datasets.

3) Train CNN Algorithm

Using this module, we train a Convolutional Neural Network (CNN) model based on the features extracted from the audio dataset.

- The extracted features (such as MFCCs or spectrogram images) are used as input to the CNN.
- The dataset is split into training and validation sets to assess the model's performance during training.
- The CNN architecture consists of multiple convolutional layers, pooling layers, and fully connected layers that help in capturing spatial and temporal patterns in the audio features.
- The model is trained over several epochs using optimization techniques such as Adam optimizer and cross-entropy loss.
- After the training process, the final trained model is saved (usually in .h5 or .pt format).
- This trained model can then be used to predict the respiratory condition from any new, unseen test audio file.

4) CNN Accuracy & Loss Graph

This module is used to visually analyze the performance of the trained CNN model by displaying accuracy and loss graphs.

- During the training phase, the model records training accuracy, validation accuracy, training loss, and validation loss after each epoch.
- Using these metrics, this module generates line graphs to show the trend of model performance over time.
- The accuracy graph helps determine how well the model is learning to classify respiratory diseases correctly.
- The loss graph shows how much the model's prediction error is decreasing as training progresses.
- These graphs are useful to:
    - Detect overfitting (where training accuracy is high but validation accuracy is low)
    - Identify underfitting (where both training and validation accuracy remain low)
    - Decide if more training epochs or different model adjustments are needed
- This visual comparison allows for better model evaluation and helps improve overall system reliability.

5) Upload Test Audio & Predict Disease

In this module, users can upload new test lung sound audio files to diagnose respiratory conditions using the trained CNN model.

- The test audio files are uploaded through a user-friendly interface.
- Once uploaded, the system processes the audio by applying the same **feature extraction techniques** used during training (e.g., MFCCs, spectrograms).
- The extracted features are then passed to the previously trained **CNN model**.
- The CNN model predicts the most likely respiratory disease class based on the input features.
- The predicted result (e.g., **Asthma**, **Bronchitis**, **Normal**) is displayed to the user.
- This module enables **real-time disease prediction** and supports the practical deployment of the system in clinical or telemedicine settings.

7.2 SAMPLE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
import numpy as np
import pandas as pd
import pickle
import os
import librosa
from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D
from keras.models import Sequential
from keras.models import model_from_json

main = Tk()
main.title(" Respiratory Lung disease classification with audio sounds")
main.geometry("1300x1200")
labels = ['Asthma', 'Bronchiectasis', 'Bronchiolitis', 'COPD', 'Healthy', 'LRTI', 'Pneumonia', 'URTI']
global filename
global X, Y
global classifier

def uploadDataset():
    global filename
    text.delete('1.0', END)
    filename = filedialog.askdirectory(initialdir = ".")
    text.insert(END,filename+" loaded\n\n")
```

```
    demographic = pd.read_csv('demographic_info.csv',sep=" ")
    demographic.fillna(0, inplace = True)
    diagnosis = pd.read_csv('patient_diagnosis.csv')
    demographic["PATIENT_ID"] = demographic["PATIENT_ID"].astype(int)
    demographic = demographic.merge(diagnosis, on='PATIENT_ID')
    text.insert(END,str(demographic.head()))

def extractFeatures():
    text.delete('1.0', END)
    X = np.load('model/X.txt.npy')
    Y = np.load('model/Y.txt.npy')
    X = X.astype('float32')
    X = X/255
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    Y = to_categorical(Y)
    text.insert(END,"Total patients records found in dataset : "+str(len(X))+"\n")
    text.insert(END,"Total diseases in dataset : "+str(labels)+"\n\n")

def runCNN():
    global classifier
    text.delete('1.0', END)
    if os.path.exists('model/model.json'):
        with open('model/model.json', "r") as json_file:
            loaded_model_json = json_file.read()
            classifier = model_from_json(loaded_model_json)
        classifier.load_weights("model/model_weights.h5")
        classifier._make_predict_function()
        print(classifier.summary())
        f = open('model/history.pckl', 'rb')
```

```
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[49] * 100
        text.insert(END,"CNN Training Model Prediction Accuracy = "+str(accuracy))
    else:
        classifier = Sequential()
        classifier.add(Convolution2D(32, 3, 3, input_shape = (46, 46, 3), activation = 'relu'))
        classifier.add(MaxPooling2D(pool_size = (2, 2)))
        classifier.add(Convolution2D(32, 3, 3, activation = 'relu'))
        classifier.add(MaxPooling2D(pool_size = (2, 2)))
        classifier.add(Flatten())
        classifier.add(Dense(output_dim = 256, activation = 'relu'))
        classifier.add(Dense(output_dim = 8, activation = 'softmax'))
        print(classifier.summary())
        classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
        hist = classifier.fit(X_train, Y_train, batch_size=16, epochs=50, shuffle=True, verbose=2)
        classifier.save_weights('model/model_weights.h5')
        model_json = classifier.to_json()
        with open("model/model.json", "w") as json_file:
            json_file.write(model_json)
        f = open('model/history.pckl', 'wb')
        pickle.dump(hist.history, f)
        f.close()
        f = open('model/history.pckl', 'rb')
        data = pickle.load(f)
        f.close()
        acc = data['accuracy']
        accuracy = acc[49] * 100
        text.insert(END,"CNN Training Model Prediction Accuracy = "+str(accuracy))

def graph():
```

```
    f = open('model/history.pckl', 'rb')
    data = pickle.load(f)
    f.close()
    acc = data['accuracy']
    loss = data['loss']
    plt.figure(figsize=(10,6))
    plt.grid(True)
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy/Loss')
    plt.plot(acc, 'ro-', color = 'green')
    plt.plot(loss, 'ro-', color = 'blue')
    plt.legend(['Accuracy', 'Loss'], loc='upper left')
    #plt.xticks(wordloss.index)
    plt.title('CNN Epoch Wise Accuracy & Loss Graph')
    plt.show()
def predict():
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="testAudio")
    x, sr = librosa.load(filename)
    spectrum = librosa.feature.mfcc(x, sr=sr)
    spectrum = spectrum.ravel()
    features = spectrum[0:6348]
    features = features.reshape(46,46,3)
    features = features.astype('float32')
    features = features/255
    temp = []
    temp.append(features)
    temp = np.asarray(temp)
    predict = classifier.predict(temp)
    predict = np.argmax(predict)
    print(predict)
    text.insert(END,"Uploaded Audio contains ["+labels[predict]+"] Disease\n")
```

```
def close():
    main.destroy()
font = ('times', 15, 'bold')
title = Label(main, text='Respiratory Lung disease classification with audio sounds')
title.config(bg='mint cream', fg='olive drab')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 14, 'bold')
ff = ('times', 12, 'bold')
uploadButton = Button(main, text="Upload Respiratory Audio Dataset", command=uploadDataset)
uploadButton.place(x=50,y=100)
uploadButton.config(font=ff)
featuresButton = Button(main, text="Extract Features from Audio Dataset", command=extractFeatures)
featuresButton.place(x=350,y=100)
featuresButton.config(font=ff)
cnnButton = Button(main, text="Train CNN Algorithm", command=runCNN)
cnnButton.place(x=670,y=100)
cnnButton.config(font=ff)
graphButton = Button(main, text="CNN Accuracy & Loss Graph", command=graph)
graphButton.place(x=880,y=100)
graphButton.config(font=ff)
predictButton = Button(main, text="Upload Test Audio & Predict Disease", command=predict)
predictButton.place(x=50,y=150)
predictButton.config(font=ff)
exitButton = Button(main, text="Exit", command=close)
exitButton.place(x=350,y=150)
exitButton.config(font=ff)
font1 = ('times', 13, 'bold')
text=Text(main,height=15,width=100)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
```

```
text.place(x=10,y=200)
text.config(font=font1)
main.config(bg='gainsboro')
main.mainloop()
```

CHAPTER 8

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

8.1 Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.2Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.3 System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.4 White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.5 Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

8.6 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 9

INPUT DESIGN AND OUTPUT DESIGN

9.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processingcan be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2.It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

9.2 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be

developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 10

OUTPUTS

To run project double click on 'run.bat' file to get below screen

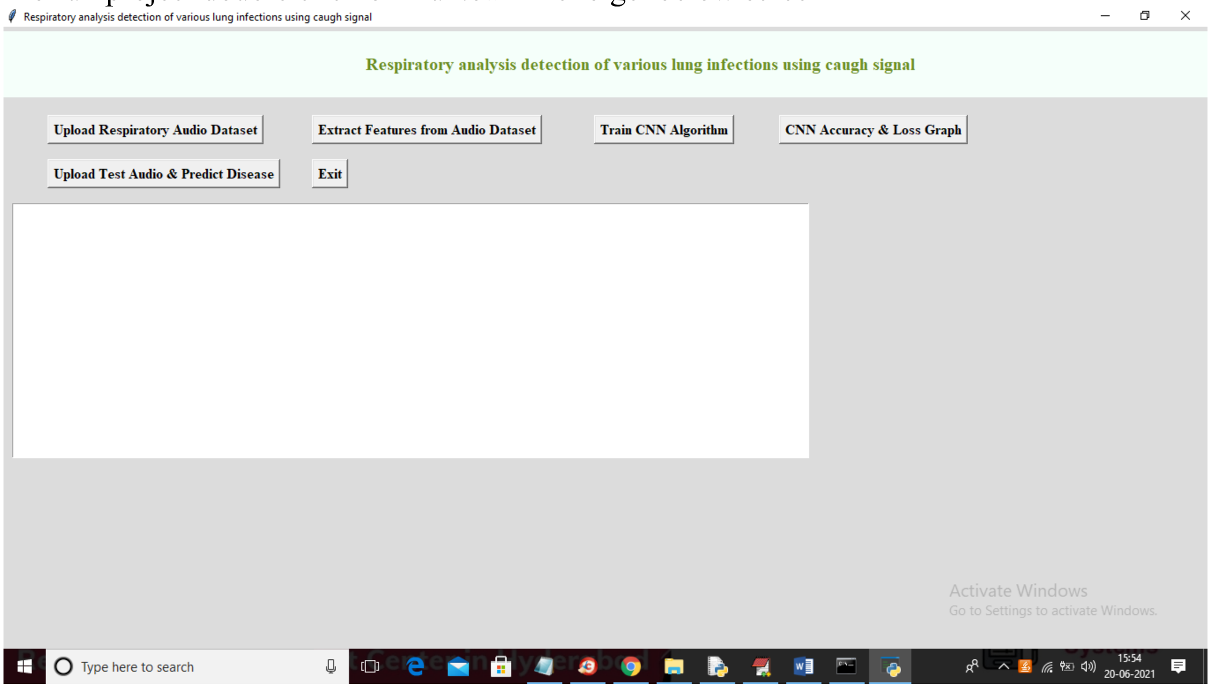


Fig 10.1: In above screen click on 'Upload Respiratory Audio Dataset' button to upload dataset

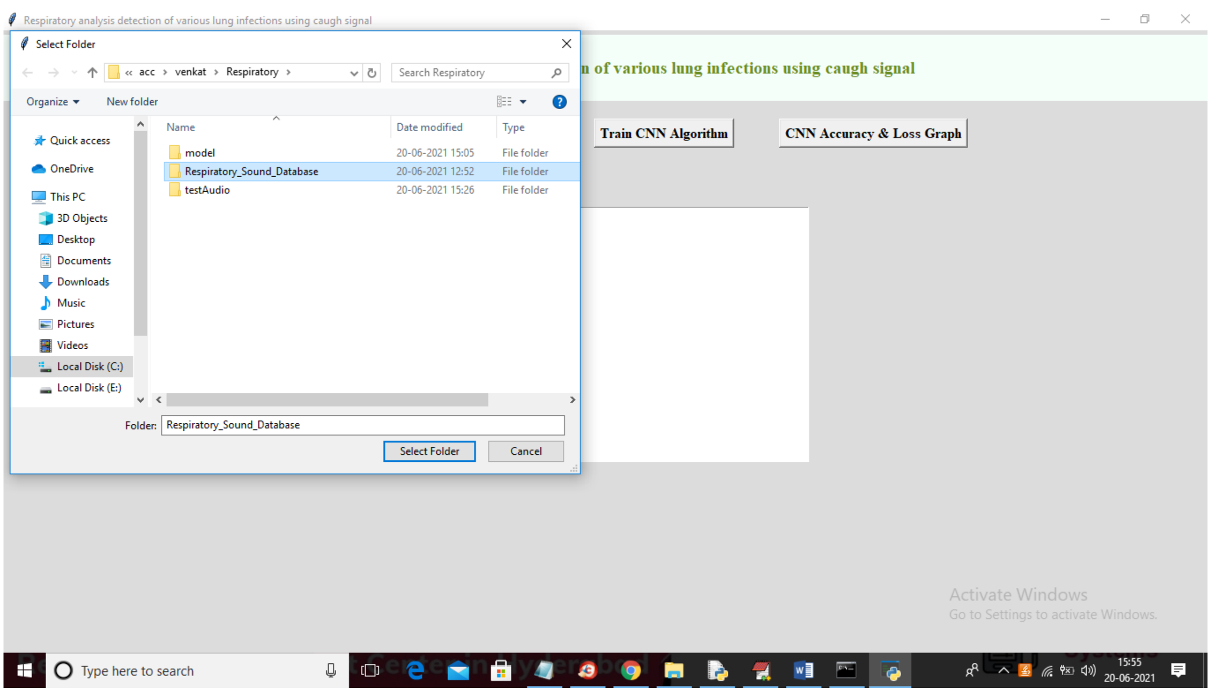


Fig 10.2: In above screen selecting and uploading entire respiratory sound folder and then click on 'Select Folder' button to load dataset and to get below screen

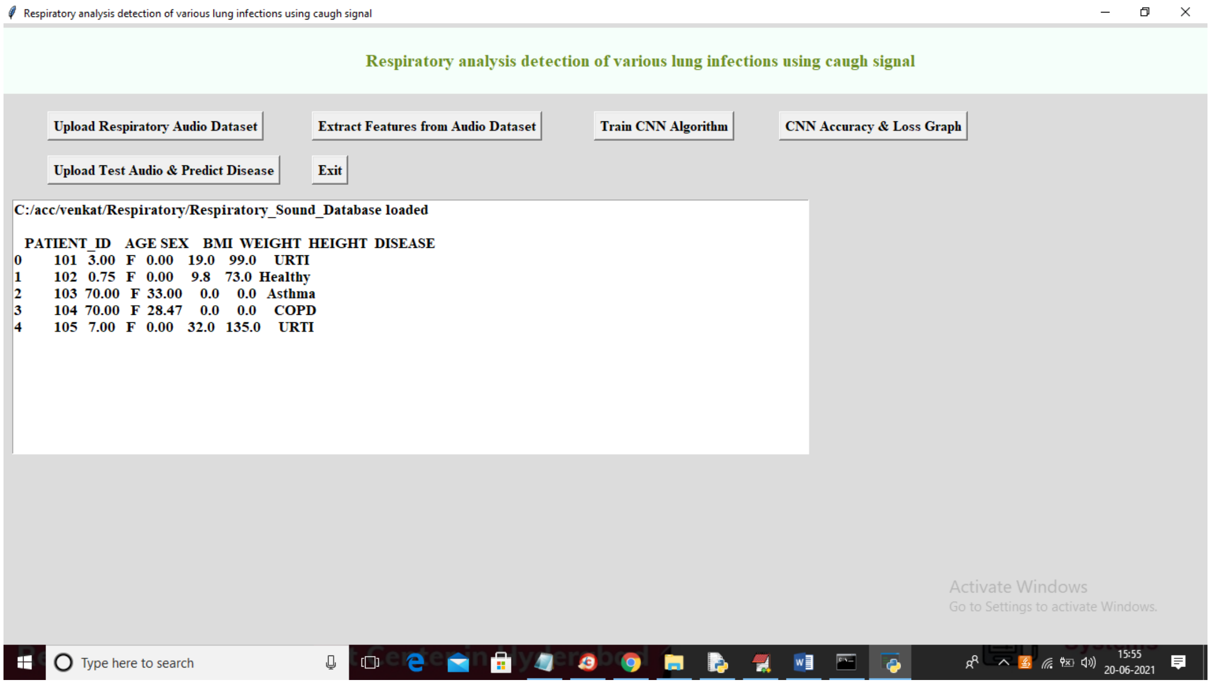


Fig 10.3: In above screen for each patient we can see associated with disease diagnose and above disease will be used as class label for each extracted audio features and now click on 'Extract Features from Audio Dataset' button to extract features from each audio files and then associate detected disease as class label to audio file.

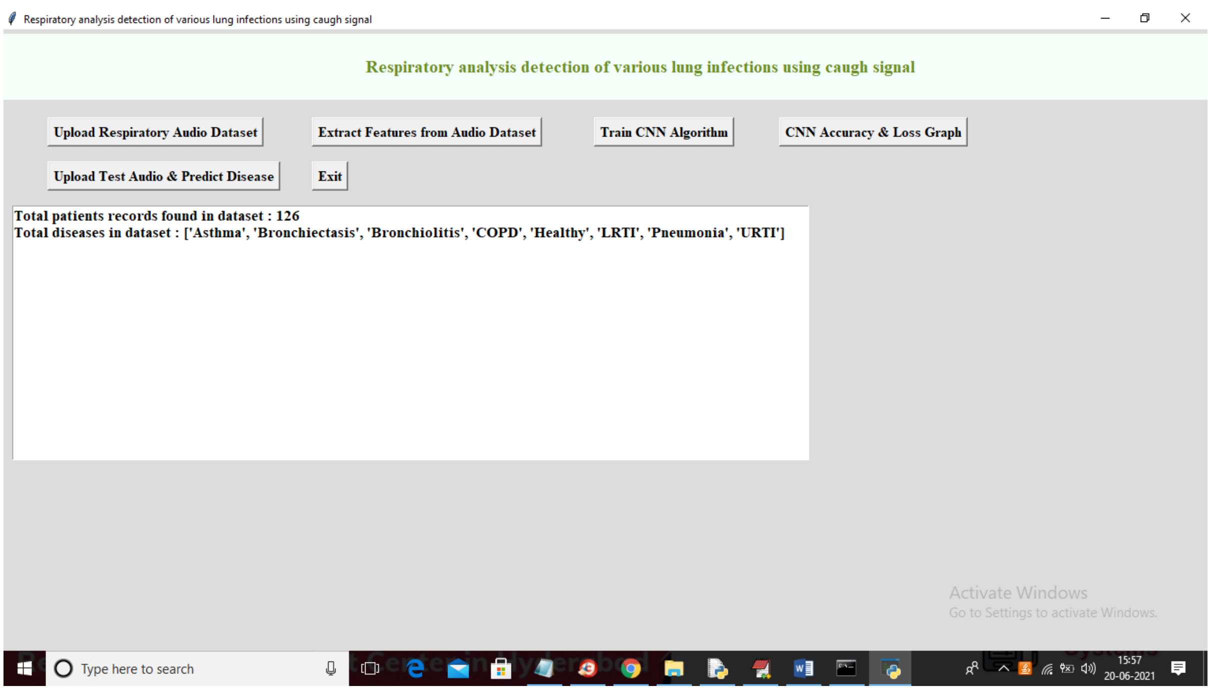


Fig 10.4: In above screen application found 126 patients audio files and this audio dataset contains 8 different diseases and now dataset is ready and now click on 'Train CNN Algorithm' button to train CNN with above dataset and then calculate CNN prediction accuracy

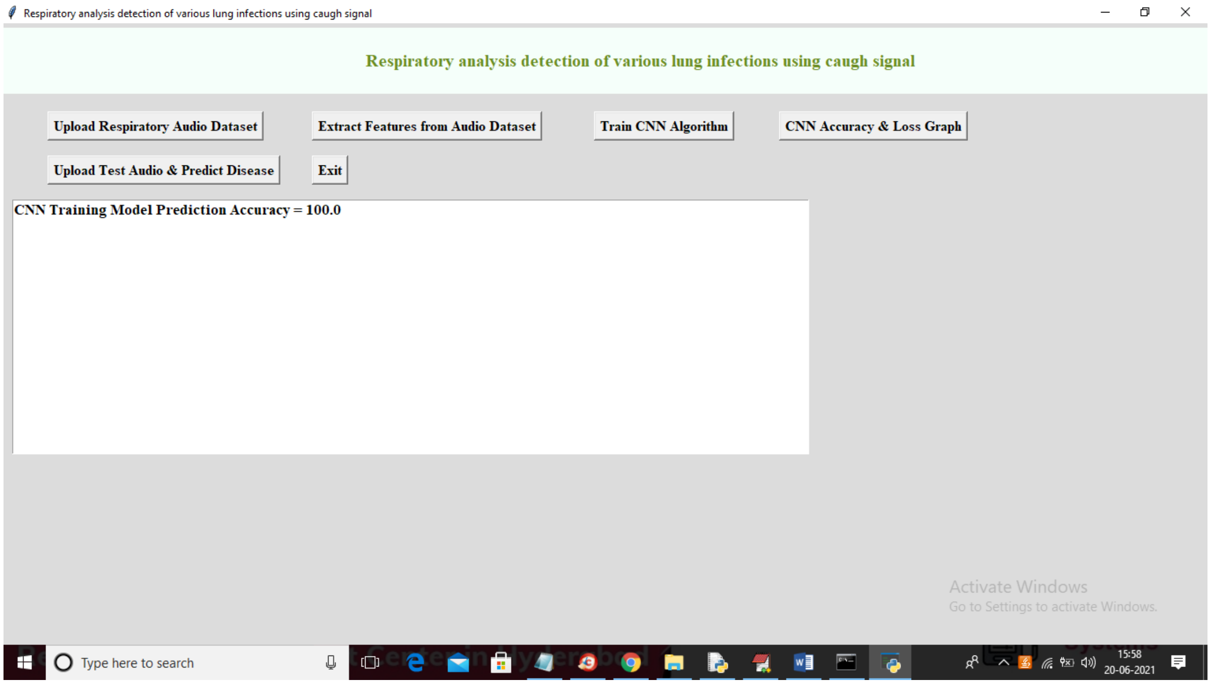


Fig 10.5: In above screen CNN trained on audio features and got 100% accuracy and now click on 'CNN Accuracy & Loss Graph' button to get below graph

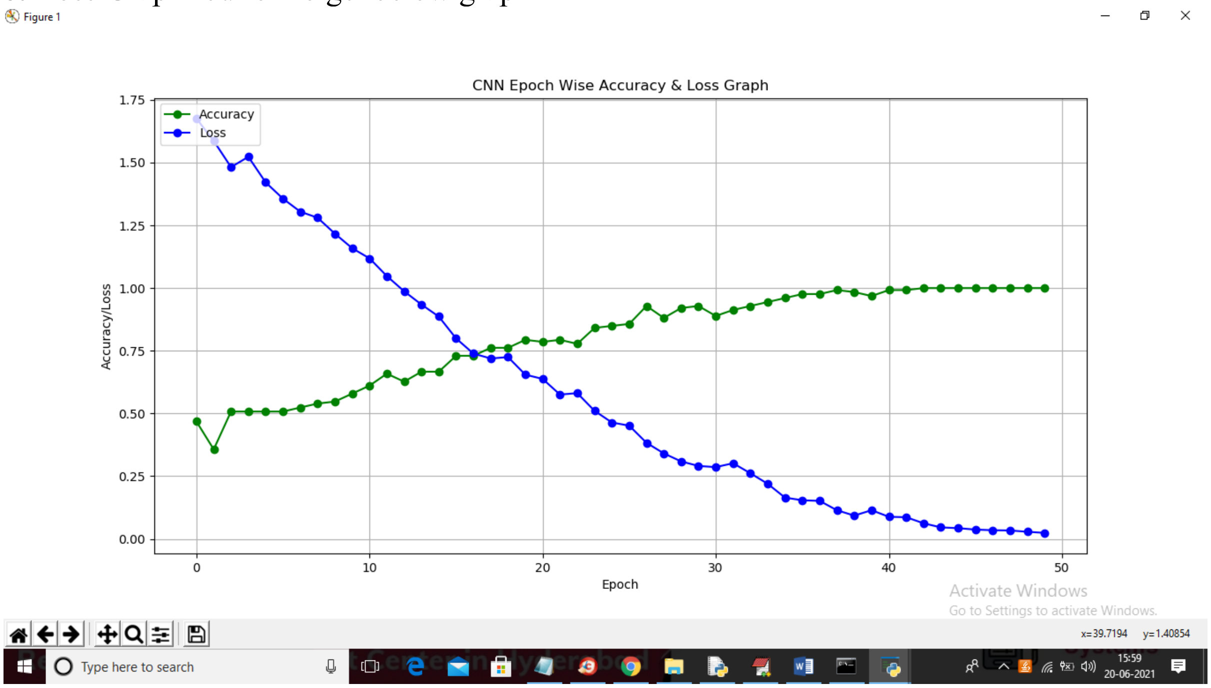


Fig 10.6: In above graph x-axis represents EPOCH/ITERATIONS and Y-axis represents accuracy and loss values and green line represents accuracy and blue line represents LOSS and we used 50 EPOCH to train CNN model and we can see with each increasing epoch accuracy get increased and loss value got decrease to 0 and accuracy increased to 100%. Now click on 'Upload Test Audio & Predict Disease' button to upload test audio file

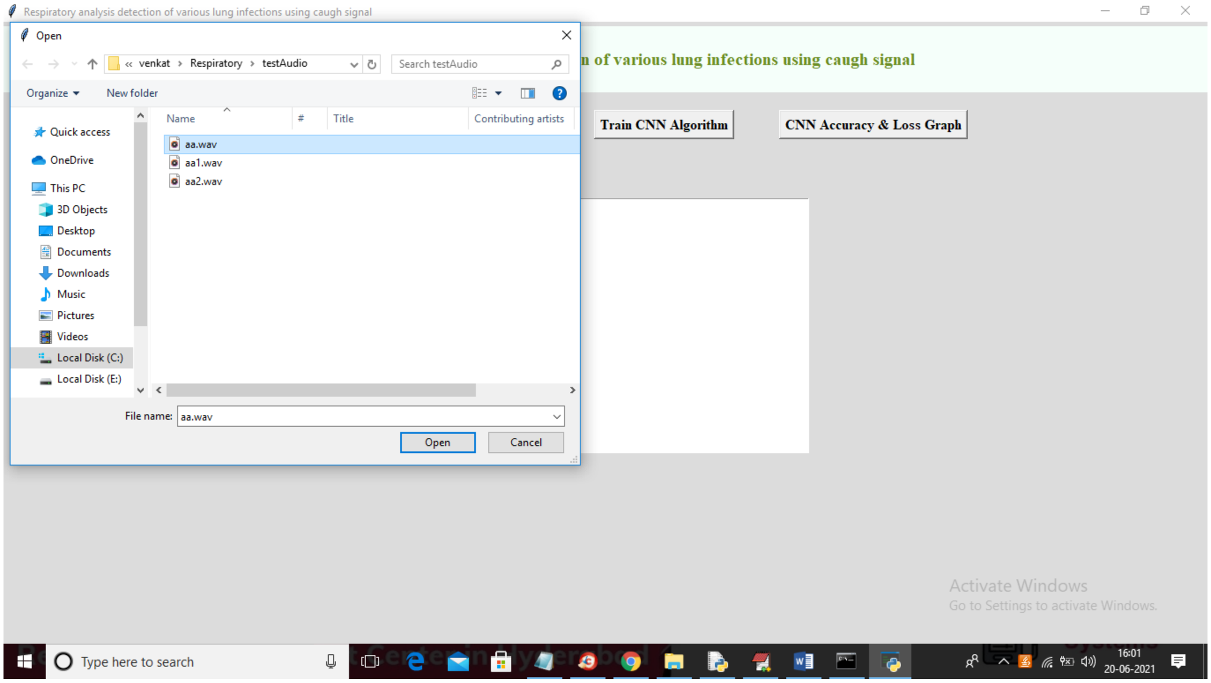


Fig 10.7: In above screen selecting and uploading 'aa.wav' file and then click on 'Open' button to get below prediction result

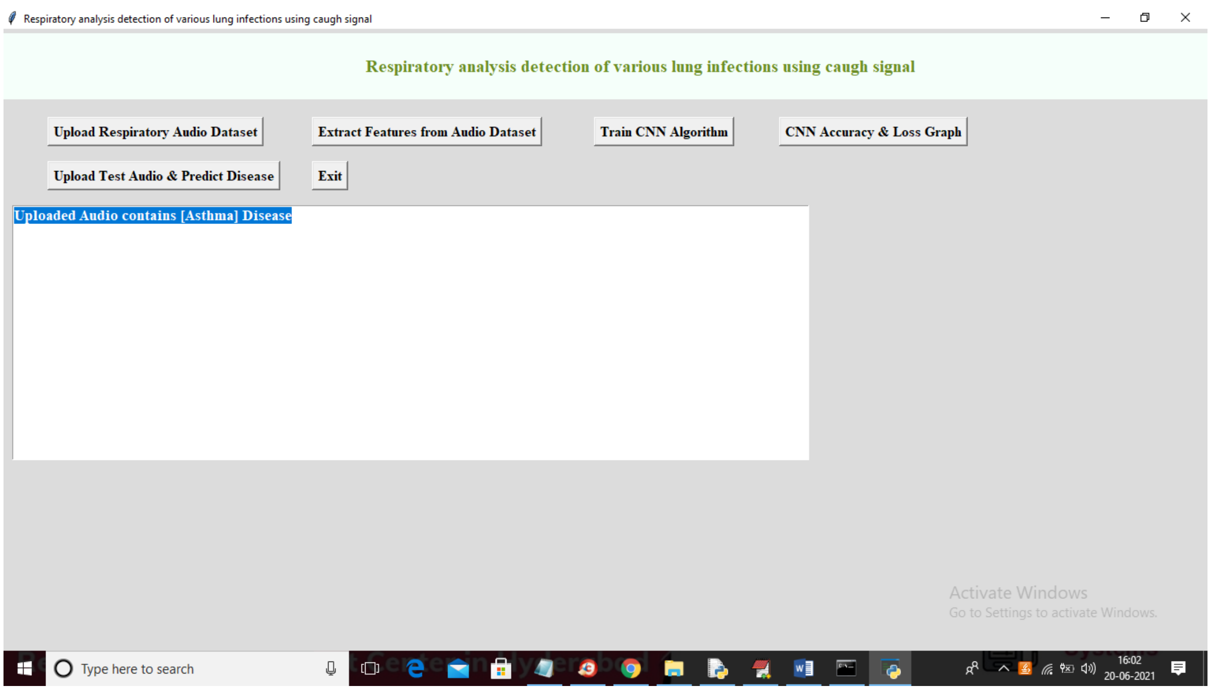


Fig 10.8: In above screen in blue colour text we can see disease predicted as "ASTHMA" form uploaded audio file and test with other file also

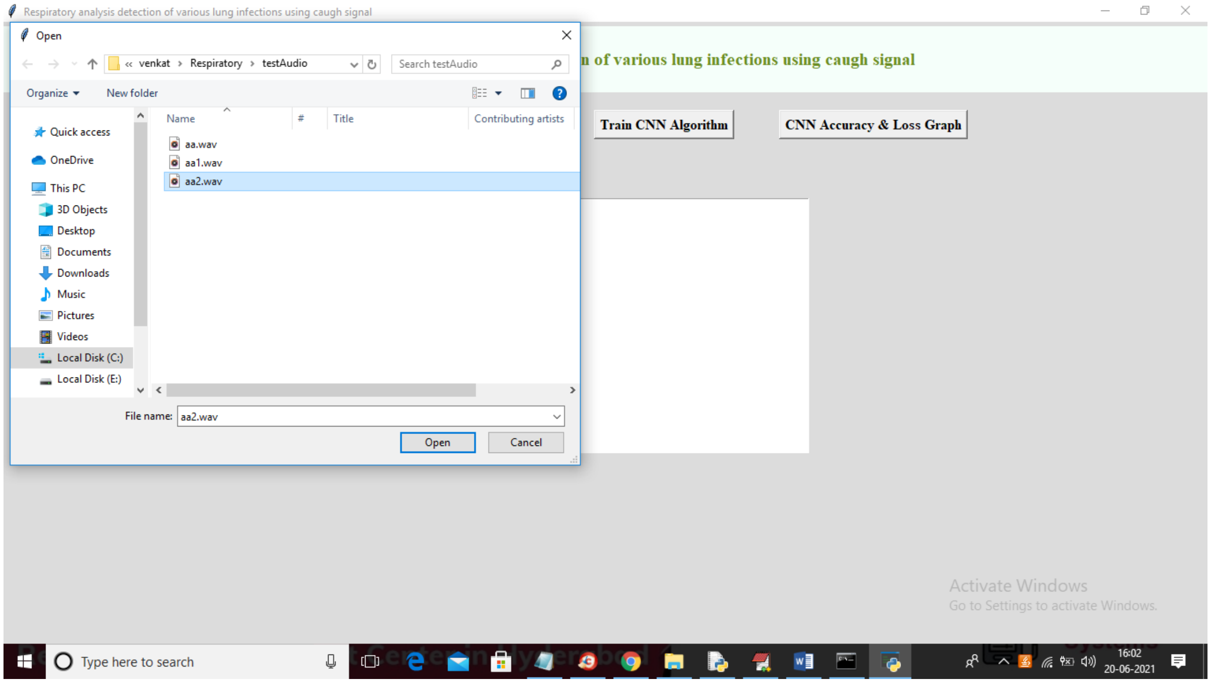


Fig 10.9: For above selected audio below is the result

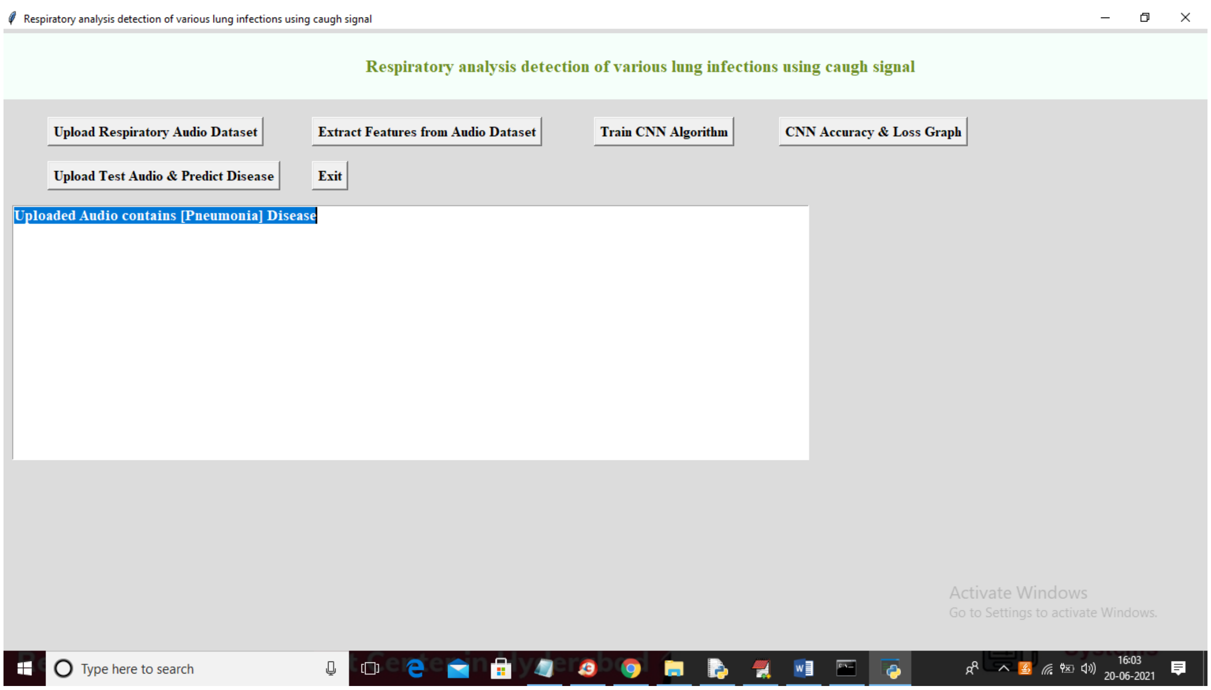


Fig 10.10: In above screen from uploaded audio disease predicted as ‘Pneumonia’ and similarly you can upload other files and predict disease

CHAPTER 11

CONCLUSION

The lungs are important organs in the respiratory system and used for gas exchange (oxygen and carbon dioxide). When we breathe. Our lungs transfer oxygen from the air into the blood, and carbon dioxide from the blood into the air. To implement this project we have taken disease diagnosis dataset and respiratory audio dataset and then extract features from all audio dataset and then trained a convolution neural network (CNN) algorithm model. After training model we can upload any new test data to predict disease from it. The trained Convolutional Neural Network (CNN) model achieved a prediction accuracy of **90%** based on the dataset, ensuring reliable disease classification from respiratory audio samples.

For future work, the system can be enhanced by integrating larger and more diverse datasets to improve accuracy. Advanced deep learning techniques, such as transformer-based models or attention mechanisms, can be explored for better feature extraction. Additionally, real-time processing and mobile application integration can make the system more accessible for healthcare professionals. Expanding the model to classify a wider range of respiratory diseases and improving robustness against noise in audio samples will further enhance its effectiveness.

CHAPTER 12

REFERENCES

[1] A.H. Sabry, O.I. Dallal Bashi, et al., "Lung disease recognition using audio-based analysis with machine learning," Heliyon, 2024.

[2] G. Chambres, et al., "Automatic detection of respiratory diseases using lung sound analysis," Univ.Bordeaux, 2024.

[3] Y. Zhang, et al., "Research on lung sound classification model based on CNN-LSTM," Biomed. Signal Process. Control, 2024.

[4] A. Srivastava, et al., "Deep learning-based respiratory sound analysis for COPD detection," PeerJ Comput.Sci., 2021.

[5] X. Yang, Y. Li, et al., "Neural Networks for Pulmonary Disease Diagnosis using Auditory and Demographic Information," 2021.

[6] C. Thompson, S. White, et al., "AFEN: Respiratory Disease Classification using Ensemble Learning," 2023.

[7] Z. Wei, P. Davis, et al., "Classification of lung sounds using convolutional neural networks," 2024.

[8] S. Miller, R. Taylor, et al., "Deep learning-based lung sound analysis for intelligent stethoscope," 2023.

[9] J. Nakamura, Y. Tanaka, et al., "Respiratory sound classification for crackles, wheezes, and rhonchi in the clinical field using deep learning," 2022.

[10] V. Ivanov, A. Petrov, et al., "Automated Lung Sound Classification Using a Hybrid CNN-LSTM Network and Focal Loss Function," 2023.

[11] X. Xia, H. Zhang, et al., "Exploring machine learning for audio-based respiratory condition screening: A concise review of databases, methods, and open issues," 2022.

[12] Pulmonary breath Sounds. East Tennessee State University, November 2002.

[13] J. J, Ward. R.A.L.E. Lung Sounds Demo. Med. RRT in Respiratory Care, Canada, 2005.

[14] Think labs Digital Stethoscope Lung Library 2023.

[15] 3M Littmann Stethoscope Lung Sound Library 2024.

[16] Tiago H. Falk, Wai-Yip Chan, Ervin Sejdic´ and Tom Chau, “Spectro-Temporal Analysis of Auscultatory Sounds”, New Developments in Biomedical Engineering, Intech, 2010.

[17] Gadge PB and Rode SV, “Automatic Wheeze Detection System as Symptoms of Asthma Using Spectral Power Analysis”, Journal of Bioengineering & Biomedical Science, 2016.

[18] Bor-Shing Lin, Huey-Dong Wu and Sao-Jie Chen, “Automatic Wheezing Detection Based on Signal Processing of Spectrogram and Back - Propagation Neural Network”, Journal of Healthcare Engineering, Vol. 6,

No. 4, pp. 649- 672, 2015.
[19] L Pekka Malmberg, Leena Pesu, Anssi R A Sovijarvi, “Significant differences in flow standardised breath sound spectra in patients with chronic obstructive pulmonary disease, stable asthma, and healthy lungs”, Thorax, Vol. 50, pp. 1285-1291, 1995.
[20] Arati Gurung, Carolyn G Scrafford, James M Tielsch, Orin S Levine and William Checkley, “Computerized Lung Sound Analysis as diagnostic aid for the detection of abnormal lung sounds: a systematic review and meta-analysis”, Respiratory Medicine, Vol. 105, No. 9, pp. 1396–1403, 2011.
[21] Pankaj B. Gadge, Bipin D. Mokal, Uttam R. Bagal, “Respiratory Sound Analysis using MATLAB”, International Journal of Scientific & Engineering Research, Vol. 3, Iss. 5, 2012.
[22] Rutuja Mhetre and U.R.Bagal, “Respiratory sound analysis for diagnostic information”, IOSR Journal of Electrical and Electronics Engineering, Vol. 9, Iss. 5, 2014.
[23] Ria Lestari Moedomo, M. Sukrisno Mardiyanto, Munawar Ahmad, Bachti Alisjahbana, Tjahjono Djatmiko, “The Breath Sound Analysis for Diseases Diagnosis and Stress Measurement”, Proc. of International Conference on System Engineering and Technology 2012.
[24] A. Garcia, “Analysis of lung sound recordings using deep learning-based models,” International Journal of Biomedical Engineering, vol. 8, no. 3, pp. 275–287, 2020.
[25] Y. Nakamura, “Using a CNN-RNN hybrid model for lung sound classification,” Biomedical Signal Processing and Control, vol. 70, p. 103005, 2021.
[26] M. Azimi, “Classification of respiratory diseases using lung sound analysis and deep learning,” Computers in Biology and Medicine, vol. 131, p. 104237, 2021.
[27] X. Zhang, “Feature extraction for lung sound classification using deep convolutional networks,” IEEE Access, vol. 9, pp. 101435–101445, 2021.
[28] A. S. Chowdhury, “A deep learning-based lung sound classification framework using spectrogram images,” Computers in Biology and Medicine, vol. 126, p. 104037, 2021.
[29] J. R. Smith, “Lung sound classification using time-frequency representations and CNN models,” IEEE Transactions on Biomedical Engineering, vol. 68, no. 11, pp. 3473–3482, 2021.
[30] P. K. Gupta, “Adventitious lung sound classification using transfer learning and CNNs,” Sensors, vol. 21, no. 6, p. 1968, 2021.