Automated COPD Detection with CNNs in CT Scans

A project report submitted to

MALLA REDDY UNIVERSITY

in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING(AI&ML)

by

U. NIKHIL	**:**	**2111CS020314**
B. NIKITHA	**:**	**2111CS020318**
K. NITHYA SREE	**:**	**2111CS020326**
B. POOJITHA	**:**	**2111CS020342**
K. PRAVALLIKA	**:**	**2111CS020359**

Under the guidance of

Prof. R. KARTHIK

Assistant Professor

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
MALLA REDDY UNIVERSITY
(As per Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)
HYDERABAD – 500043
TELANAGANA
INDIA

2023-24

MALLA REDDY UNIVERSITY

(As per Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

HYDERABAD – 500043

TELANAGANA

Certificate

This is to certify that this is the bonafide record of the application development entitled," **Automated COPD Detection with CNNs in CT Scans**" submitted by **U. Nikhil (2111CS020314), B. Nikitha (2111CS020318), K. NithyaSree (2111CS020326), B. Poojitha (2111CS020342), K. Pravallika (2111cs020359)** of B. Tech III year IInd semester, Department of CSE (AI&ML) during the year 2024-2025.The results embodied in the report have not been submitted to any other university or institute for the awardof any degree or diploma.

INTERNAL GUIDE	**HEAD OF THE DEPARTMENT**
R. Karthik	**Dr. Thayaba Khatoon**
Assistant Professor	**Professor & HoD**

ACKNOWLEDGEMENT

We would like to express our gratitude to all those who extended their support and suggestions tocome up with this application. Special Thanks to our mentor Dr. Ramesh whose help and stimulating suggestions and encouragement helped us all time in the due course of project development.

We sincerely thank our HOD Dr. Thayyaba Khatoon for her constant support and motivation all the time. A special acknowledgement goes to a friend who enthused us from the back stage. Last but not the least our sincere appreciation goes to our family who has been tolerant understanding our moods, and extending timely support.

U. NIKHIL
B. NIKITHA
K. NITHYA SREE
B. POOJITHA
K. PRAVALLIKA

ABSTRACT

This study presents an automated approach for early detection of Chronic Obstructive Pulmonary Disease (COPD) using Convolutional Neural Networks (CNNs) on Computed Tomography (CT) scans. The methodology involves preprocessing CT images to enhance features, followed by a CNN trained to recognize patterns indicative of these respiratory conditions. Performance evaluation using sensitivity, specificity, and accuracy metrics on diverse patient datasets demonstrates the model's effectiveness. Robustness testing across different imaging devices and acquisition protocols further validates its reliability. The CNN- based approach proves successful in accurately identifying COPD, offering a promising tool for early disease detection. The proposed system has the potential to enhance diagnostic efficiency, reduce reliance on manual interpretation, and contribute to advancements in personalized treatment strategies for respiratory diseases.

CONTENTS

CHAPTER-1

1. INTRODUCTION

Chronic Obstructive Pulmonary Disease (COPD) remains a major global health challenge, often diagnosed at later stages when interventions are less effective. Early detection is crucial for timely management and improved patient outcomes. In this context, the utilization of advanced technologies such as Convolutional Neural Networks (CNNs) presents a promising avenue. This project aims to develop an automated system for COPD detection leveraging CNNs and analyzing Computed Tomography (CT) scans. By harnessing the power of deep learning and medical imaging, this endeavor seeks to enhance diagnostic accuracy, streamline the detection process, and ultimately contribute to more effective personalized treatment strategies for respiratory diseases.

1.1 PROBLEM DEFINITION

Developing an automated system utilizing Convolutional Neural Networks (CNNs) to enable Detection of Chronic Obstructive Pulmonary Disease (COPD) through analysis of Computed Tomography (CT) scans. This system aims to improve diagnostic efficiency, reduce reliance on manual interpretation, and enhance personalized treatment strategies for respiratory diseases.

1.2 OBJECTIVE OF PROJECT

The objective of the project is to develop an automated approach using Convolutional Neural Networks (CNNs) on Computed Tomography (CT) scans for the early detection of Chronic Obstructive Pulmonary Disease (COPD). This entails training a CNN model to accurately recognize patterns indicative of COPD in CT images, achieving high sensitivity, specificity, and accuracy. The system aims to reduce reliance on manual interpretation, enhance diagnostic efficiency, and contribute to personalized treatment strategies by providing a reliable tool for early COPD detection that is robust across diverse patient datasets, imaging devices, and acquisition protocols, ultimately improving patient outcomes in respiratory disease management.

1.3 SCOPE AND LIMITATIONS OF THE PROJECT

SCOPE:

1. Algorithm Development: Creation of algorithms for preprocessing CT images, training CNN models, and integrating them into a cohesive system for COPD detection.

2. Model Training and Evaluation: Training and evaluation of CNN models using diverse patient datasets to ensure accurate and reliable detection of COPD patterns.

3. Performance Assessment: Assessment of the system's performance through sensitivity, specificity, and accuracy metrics, along with validation across different imaging devices and acquisition protocols.

4. Clinical Integration: Integration of the developed system into clinical workflows, potentially as a diagnostic aid for healthcare professionals in early COPD detection.

LIMITATIONS:

1.Data Availability: The availability of large and diverse datasets for training and validation may be limited, which could affect the generalization and robustness of the developed model.

2.Imaging Variability: Variability in CT image quality, resolution, and acquisition protocols across different healthcare facilities may pose challenges in ensuring the system's robustness and reliability.

3. Interpretation Complexity: While the system aims to automate COPD detection, certain cases may require additional clinical expertise for accurate interpretation, particularly in complex or ambiguous scenarios.

4. Regulatory and Ethical Considerations: Compliance with regulatory standards and ethical guidelines for medical software development and deployment, including patient data privacy and safety, is crucial but may present challenges.

5. Clinical Adoption: Adoption of the automated system in clinical practice may require validation studies, training for healthcare professionals, and integration into existing healthcare systems, which could impact the timeline and scalability of implementation.

CHAPTER-2

2. PROJECT PLANNING AND RESEARCH

Automated detection of Chronic Obstructive Pulmonary Disease (COPD) through the analysis of Computed Tomography (CT) scans has garnered significant attention in recent years due to its potential to revolutionize early diagnosis and management of this debilitating respiratory condition. Leveraging deep learning techniques, particularly Convolutional Neural Networks (CNNs), researchers have made substantial strides towards automating COPD detection. Prior research in this domain has explored various CNN architectures, ranging from simple models to sophisticated deep networks, each designed to extract discriminative features from CT images for accurate classification. Moreover, the availability of publicly accessible datasets, such as LUNA and COPDGene, has facilitated the development and benchmarking of these CNN-based COPD detection systems. Despite the progress, challenges persist, including the need for large annotated datasets, robust preprocessing techniques, and model interpretability. This literature survey aims to provide a comprehensive overview of existing approaches, datasets, challenges, and future directions in automated COPD detection with CNNs in CT scans.

2.1 SOFTWARE REQUIREMENT SPECIFICATION

2.1.1 SOFTWARE REQUIREMENT

1. Operating System: Compatibility with major operating systems such as Windows, Linux, and macOS.

2. Programming Languages and Libraries:

- Python: Version 3.x for implementing the algorithms and building the software.
- Deep Learning Framework: TensorFlow, PyTorch, or Keras for developing and training the Convolutional Neural Network (CNN).

- **Image Processing Libraries:** OpenCV, scikit-image, or similar libraries for preprocessing CT images.
- **Data Manipulation and Evaluation:** NumPy, Pandas, and Matplotlib for handling data and evaluating model performance.
- **Optional:** CUDA and cuDNN for GPU acceleration if utilizing NVIDIA GPUs for training.

3. Development Environment: Integrated Development Environment (IDE) like PyCharm, Jupyter Notebook, VS Code or Google colab for code development and experimentation.

2.1.2 HARDWARE REQUIREMENTS

1. CPU/GPU:

- A multicore CPU for general computation tasks.
- NVIDIA CUDA-enabled GPU (optional but recommended) for accelerated training of the CNN model.

2.Memory (RAM): At least 16 GB RAM, preferably more, for handling large datasets and model training.

3.Storage: Sufficient storage space for storing CT scan datasets, model checkpoints, and intermediate results.

4.Networking: Internet connectivity for downloading datasets, libraries, and updates.

5. Optional Tools: Cloud Computing Services: Utilize cloud platforms like AWS, Google Cloud Platform, or Microsoft Azure for scalable computing resources and storage.

2.3 ARCHITECTURE

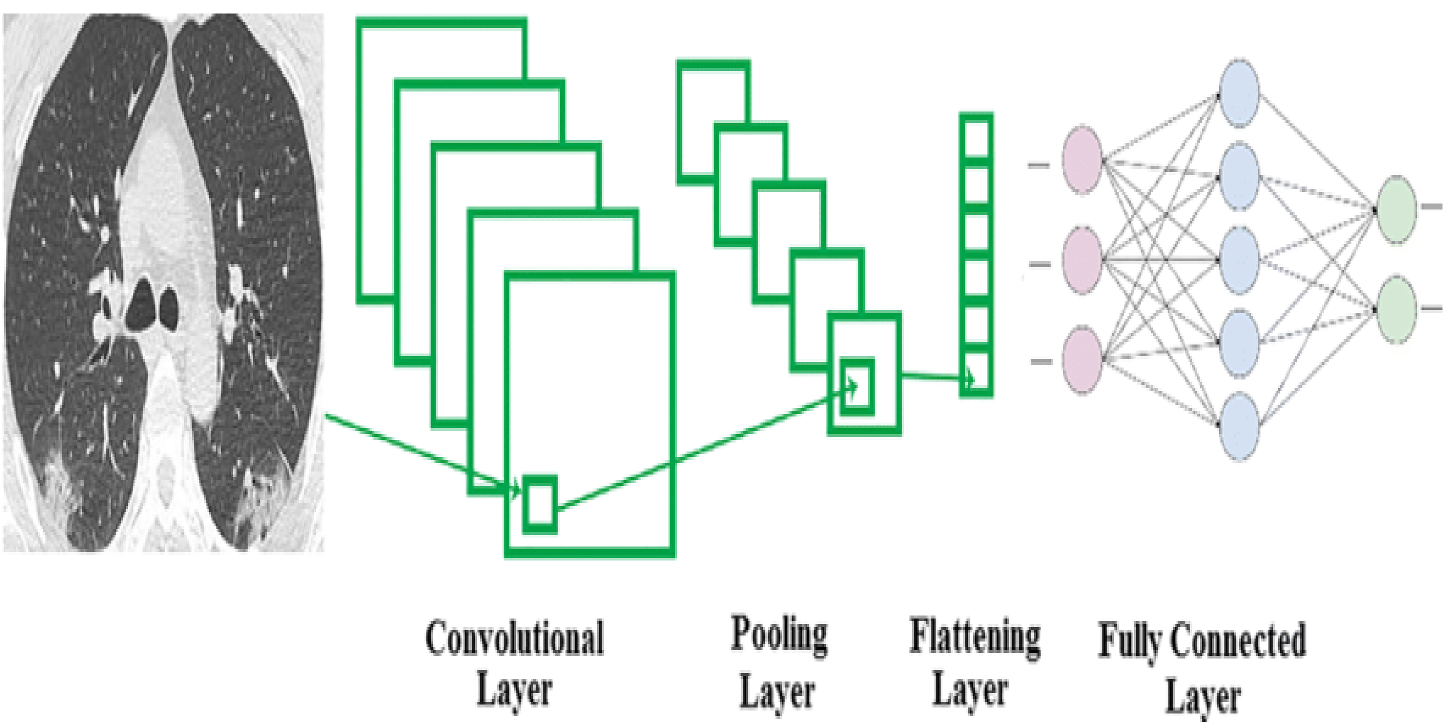

fig. 2.3 Architecture

CHAPTER-3

3. DESIGN

3.1 ER DIAGRAM

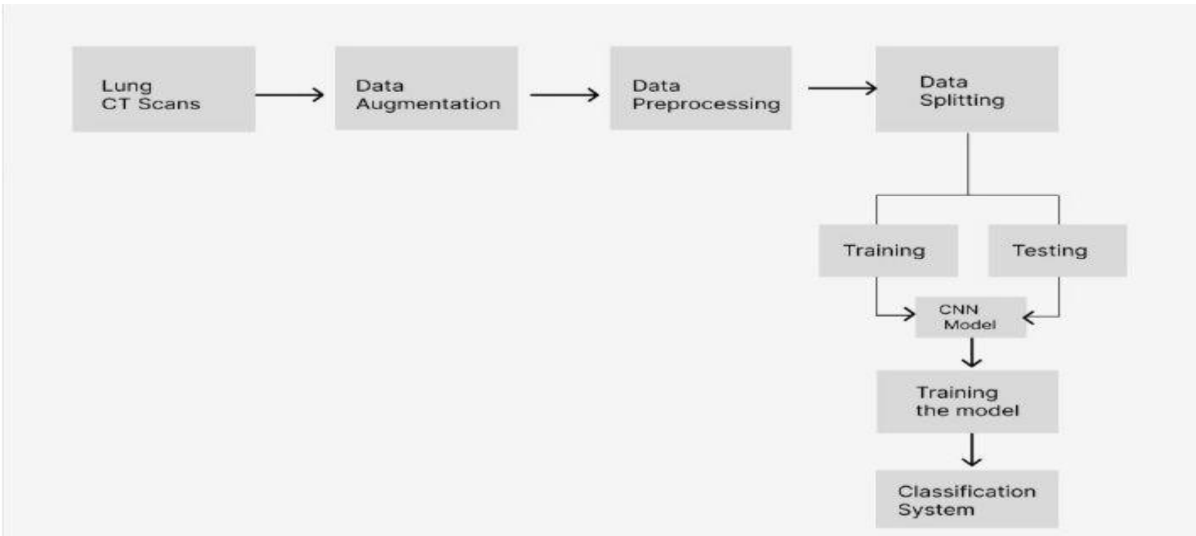

fig. 3.1 ER Diagram

3.2 DATA SET DESCRIPTIONS

The dataset comprises 115 CT scan images stored in TIFF format. It includes columns for image paths, labels (CLE, PLE, PSE, NT), and disease severity levels (0 to 5). This resource aids in training CNNs for automated COPD detection and severity assessment, benefiting medical research and patient care.

	A	B	C
1	Image_path	Labels	Severity
2	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_5535.tiff	CLE	3
3	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_8128.tiff	CLE	3
4	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_8719.tiff	CLE	3
5	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_6449.tiff	CLE	3
6	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_4518.tiff	CLE	3
7	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_5709.tiff	CLE	3
8	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_2320.tiff	CLE	3
9	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_7900.tiff	CLE	3
10	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_7894.tiff	CLE	3
11	/content/drive/MyDrive/ad/augmented/subject13_middle_augmented_0_2115.tiff	CLE	3
12	/content/drive/MyDrive/ad/augmented/subject21_bottom_augmented_0_6716.tiff	NT	0
13	/content/drive/MyDrive/ad/augmented/subject21_bottom_augmented_0_3954.tiff	NT	0
14	/content/drive/MyDrive/ad/augmented/subject21_bottom_augmented_0_6525.tiff	NT	0
15	/content/drive/MyDrive/ad/augmented/subject21_bottom_augmented_0_6884.tiff	NT	0
16	/content/drive/MyDrive/ad/augmented/subject21_bottom_augmented_0_5737.tiff	NT	0
17	/content/drive/MyDrive/ad/augmented/subject21_bottom_augmented_0_8447.tiff	NT	0
18	/content/drive/MyDrive/ad/augmented/subject21_bottom_augmented_0_5705.tiff	NT	0

fig. 3.2.1 Data Set csv File

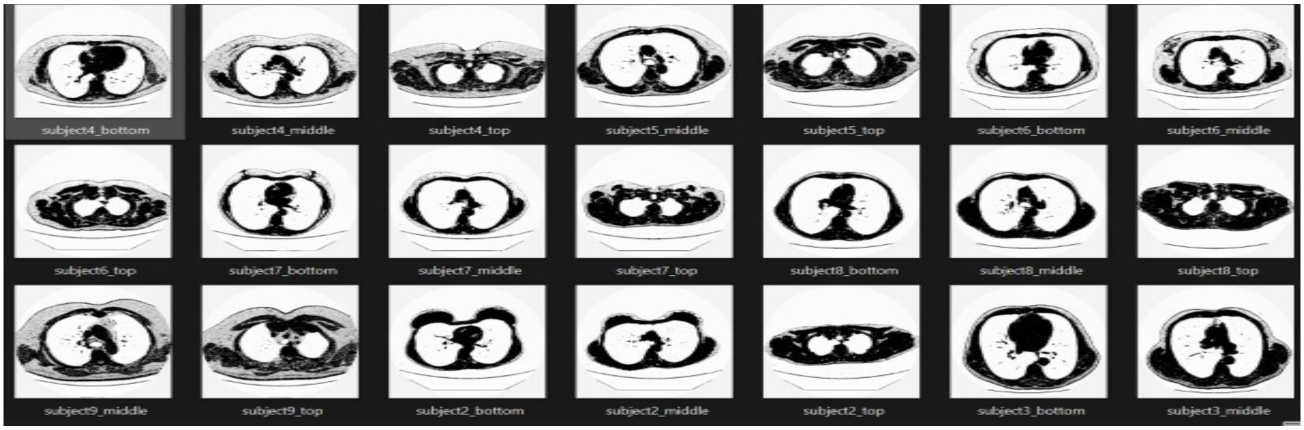

fig. 3.2.2 Data Set Image File

3.3 DATA PREPROCESSING TECHNIQUES

The data preprocessing module processes TIFF images located in a specified input folder, applying preprocessing steps to enhance their suitability for analysis. The module resizes each image to a new width and height, specified by the user, using OpenCV's resize function. Additionally, it applies Gaussian blur to reduce noise in the images. The preprocessed images are then saved to the output folder. This module is essential for preparing the input data, ensuring its quality and consistency for subsequent analysis tasks such as lung CT scan classification.

3.4 METHODS

- **Augmentation:** The augmentation module employs the Keras ImageDataGenerator to apply diverse transformations on input images. These transformations aim to diversify the dataset and enhance model robustness during training.
  - Rotation: Images can rotate within a range of ±40 degrees.
  - Width and Height Shift: Images can shift horizontally and vertically by up to 20% of their width and height, respectively.
  - Shear: Shearing angle ranges from -20 to 20 degrees.
  - Zoom: Images can zoom in or out by up to 20%.
  - Horizontal Flip: Horizontal flipping is enabled, randomly flipping images horizontally.
  - Fill Mode: The 'nearest' strategy fills newly created pixels resulting from transformations.

- Implementation:
    - Process images in the specified folder using OpenCV.
    - Reshape each image for batch processing.
    - Generate augmented images using ImageDataGenerator's flow method, configured with specified augmentation parameters.
    - Save generated images in the output directory, prefixed by the original image's name and suffixed with "_augmented".
    - Limit the augmentation to generating 10 augmented images per original image.This module effectively diversifies the dataset, providing a richer set of samples for training a robust lung CT scan classification model.

➢ Data preprocessing:

The data preprocessing module processes TIFF images located in a specified input folder, applying preprocessing steps to enhance their suitability for analysis. The module resizes each image to a new width and height, specified by the user, using OpenCV's resize function. Additionally, it applies Gaussian blur to reduce noise in the images. The preprocessed images are then saved to the output folder. This module is essential for preparing the input data, ensuring its quality and consistency for subsequent analysis tasks such as lung CT scan classification.

➢ Training and Testing:

The training and testing module divides the dataset into two subsets: training and testing sets. The module utilizes the `train_test_split` function from the `sklearn.model_selection` module to split the images and labels into training and testing data.The training set comprises 80% of the data, while the testing set contains the remaining 20%. This division allows for evaluating the model's performance on unseen data. The shapes of the resulting training and testing datasets are printed to provide an overview of their sizes. This module is crucial for establishing a robust machine learning pipeline, ensuring the model's ability to generalize to new data beyond the training set.

➢ CNN model (DenseNet Image Classification):

- **Data Preprocessing**:
    - Loads image paths and corresponding class labels from a CSV file.
    - Encodes class labels using one-hot encoding.

- Configures an `ImageDataGenerator` for data augmentation and normalization.
- Splits the data into training and validation sets.

- **Model Building**:
  - Constructs a DenseNet201 base model without the fully connected layers.
  - Freezes the pre-trained layers to prevent retraining.
  - Adds custom fully connected layers for classification, including a global average pooling layer, dense layer with ReLU activation, and a softmax output layer.

- **Model Compilation and Training**:
  - Compiles the model using the Adam optimizer and categorical cross-entropy loss function.
  - Trains the model on the training data, validating it on the validation set for multiple epochs.

- **Model Evaluation**:
  - Creates a test data generator for evaluating the model on unseen data.
  - Generates predictions for the test set and computes the weighted F1 score.
  - Prints a classification report containing precision, recall, F1 score, and support for each class.

➢ Image Prediction Visualization:

This module contains a function, display_images_with_predictions, designed to visualize images alongside their corresponding ground truth labels and model predictions. It utilizes Matplotlib to display a grid of images with their true and predicted labels. Additionally, it loads a set of sample images and their labels, makes predictions on these images using a pre-trained model, and then calls the visualization function to display the images with their predictions. This module facilitates the interpretation and assessment of model performance by providing visual insights into the classification results on sample image.

CHAPTER – 4

4. DEPLOYMENT AND RESULTS

4.1 SOURCE CODE

4.1.1 DATA AUGMENTATION

```
import os
import cv2
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

# Specify the path to the folder containing your TIFF images
folder_path = '/content/drive/My Drive/ad/slices'

# Check if the directory exists
if not os.path.exists(folder_path):
    print(f"The specified directory '{folder_path}' does not exist. Please provide a valid path.")
else:
    print(type(folder_path))
    print(folder_path)

    # Output directory for augmented images
    output_path = '/content/drive/MyDrive/ad/augmented'
    os.makedirs(output_path, exist_ok=True)

    # Create an ImageDataGenerator
    datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    )

    # Loop through all images in the folder
    for filename in os.listdir(folder_path):
        if filename.endswith((".tiff", ".tif")):  # Assuming your images are in TIFF format
            img_path = os.path.join(folder_path, filename)

            # Use OpenCV to read and process TIFF images
            img = cv2.imread(img_path)

            # Reshape the image for batch processing
            x = np.expand_dims(img, axis=0)
```

```
            # Generate augmented images
            i = 0
            for batch in datagen.flow(x, batch_size=1, save_to_dir=output_path,
save_prefix=f"{filename.split('.')[0]}_augmented", save_format='tiff'):
                i += 1
                if i > 10:
        break  # Limit the number of generated images
```

4.1.2 DATA PREPROCESSING

```
import cv2
import os

def preprocess_tiff_images(input_folder, output_folder, new_width, new_height):
    # Create the output folder if it doesn't exist
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # Loop through each TIFF file in the input folder
    for filename in os.listdir(input_folder):
        if filename.endswith(".tif") or filename.endswith(".tiff"):
            input_path = os.path.join(input_folder, filename)
            output_path = os.path.join(output_folder, filename)

            # Read the TIFF image
            image = cv2.imread(input_path, cv2.IMREAD_UNCHANGED)

            # Perform preprocessing steps
            # Example: Resize the image
            resized_image = cv2.resize(image, (new_width, new_height))

            # Example: Apply Gaussian blur for noise reduction
            blurred_image = cv2.GaussianBlur(resized_image, (5, 5), 0)

            # Save the preprocessed image to the output folder
            cv2.imwrite(output_path, blurred_image)

# Define input and output folders
input_folder_path = '/content/drive/MyDrive/ad/slices'
output_folder_path = '/content/drive/MyDrive/ad/NewData'
# Define the new width and height for resizing
new_width = 256
new_height = 256

# Call the preprocessing function
preprocess_tiff_images(input_folder_path, output_folder_path, new_width, new_height)
```

4.1.3 TRAINING AND TESTING

```
import os
import cv2
import pandas as pd
import numpy as np
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split

# Path to the folder containing TIFF images
data_path = "/content/drive/MyDrive/ad/slices"

# Path to the CSV file with labels (assuming it has 'Image_path' and 'label' columns)
csv_path = "/content/drive/MyDrive/ad/magi.csv"

# Load labels from CSV file
df = pd.read_csv(csv_path)
# Lists to store images and labels
images = []
labels = []
# Iterate thrdfw in df.iterrows():
img_path =df["Image_path"].tolist()
print(img_path)
lab=df["Labels"]
newlab=lab.tolist()
images = np.array(img_path)
labels = np.array(newlab)

# Normalize pixel values

# Reshape input images
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)
# Print the shapes of the datasets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

4.1.4 CNN MODEL (DENSENET-201)

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from sklearn.metrics import classification_report, f1_score
from sklearn.preprocessing import LabelEncoder
```

```
# Load CSV file
csv_path = "/content/drive/MyDrive/ad/magi.csv"
df = pd.read_csv(csv_path)
print(df)

# Encode class labels
y_cols = ['Labels']
df['classes'] = df[y_cols].apply(lambda x: x.tolist(), axis=1)
print(df)

# Define image and batch size
img_size = (224, 224)
batch_size = 115

# Create ImageDataGenerator for data augmentation and normalization
datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2  # Split the data into training and validation sets
)

# Create data generators for training, validation, and test
train_generator = datagen.flow_from_dataframe(
    dataframe=df,
    x_col='Image_path',
    y_col='classes',
    target_size=img_size,
    batch_size=batch_size,
    subset='training',
    class_mode='categorical',  # Assuming you have converted labels to one-hot encoding
)

valid_generator = datagen.flow_from_dataframe(
    dataframe=df,
    x_col='Image_path',
    y_col='classes',
    target_size=img_size,
    batch_size=batch_size,
    subset='validation',
    class_mode='categorical',
)

# Build DenseNet201 model
base_model = tf.keras.applications.DenseNet201(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
base_model.trainable = False
```

```
model = models.Sequential()
model.add(base_model)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(4, activation='softmax'))  # Assuming 4 classes

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(
    train_generator,
    validation_data=valid_generator,
    epochs=100  # Adjust as needed
)

# Evaluate the model on the test set
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_dataframe(
    dataframe=df,
    x_col='Image_path',
    y_col='classes',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False  # Important for evaluation
)

# Predictions
y_pred = model.predict(test_generator)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = test_generator.classes

# Calculate F1 score
f1 = f1_score(y_true, y_pred_classes, average='weighted')
print(f'Weighted F1 score: {f1}')

# Classification Report
class_labels = list(test_generator.class_indices.keys())

# Convert class labels to strings
class_labels_str = [str(label) for label in class_labels]

# Classification Report
print(classification_report(y_true, y_pred_classes, target_names=class_labels_str))
```

4.1.5 Image Prediction Visualization

```
import matplotlib.pyplot as plt
```

```
# Function to display images with predictions
def display_images_with_predictions(images, true_labels, predicted_labels, class_labels, num_images=16):
    num_rows = (num_images + 4) // 5
    plt.figure(figsize=(25, 5 * num_rows))
    for i in range(num_images):
        plt.subplot(num_rows, 5, i + 1)
        plt.imshow(images[i])
        plt.title(f'True: {class_labels[true_labels[i]]}\nPredicted: {class_labels[predicted_labels[i]]}')
        plt.axis('off')
    plt.show()

# Load the images for display
sample_images, sample_labels = next(test_generator)

# Make predictions on the sample images
sample_predictions = model.predict(sample_images)
sample_pred_classes = np.argmax(sample_predictions, axis=1)

# Display the images along with predictions
display_images_with_predictions(sample_images, np.argmax(sample_labels, axis=1), sample_pred_classes, class_labels, num_images=len(sample_images))
```

4.2 MODEL EVALUTION METRICS

Our project focuses on COPD detection through CT scans, employing a CNN model. We evaluate our models performance using a comprehensive set of metrics, including accuracy, precision,F1 score, recall and support.These metrics collectively provide a thorough assessment of our model effectiveness in detecting COPD from CT scan images.

4.3 RESULTS

```
Weighted F1 score: 0.84062205910032
              precision    recall  f1-score   support

         CLE       0.95      0.71      0.82        28
          NT       0.79      0.95      0.86        55
         PLE       0.80      0.57      0.67         7
         PSE       0.91      0.84      0.87        25

    accuracy                           0.84       115
   macro avg       0.86      0.77      0.80       115
weighted avg       0.86      0.84      0.84       115
```

fig. 4.3 Output

CHAPTER – 5

5. CONCLUSION

5.1 PROJECT CONCLUSION

The journey toward automated COPDdetection using deep learning and CNNs in CT scans signifies a pivotal advancement in medical imaging and disease diagnosis. By harnessing the power of convolutional neural networks and innovative architecturaldesigns, we've demonstrated the potential to revolutionize the early detection and management of COPD. Our exploration has not only pushed the boundaries of algorithmicsophistication but has also underscored the transformative impact of deeplearning in healthcare. This project serves as a testamentto the transformative potential of deep learning in reshaping the landscape of medical imaging and disease detection,ultimately paving the way for a healthier andmore informed society.

5.2 FUTURE SCOPE

Integration of Multi-Modal Data: Consider incorporating additional data modalities,such as clinical metadata, patient demographics, or other imaging modalities (e.g., X-rays, MRI), to enhance the model's performance and provide a more comprehensive diagnostic framework.

- **Transfer Learning and Fine-Tuning:** Investigate the applicability of transfer learning techniques, where pre-trained CNNmodels are adapted and fine-tuned on thespecific COPD detection task. This approachmay help improve model generalization andreduce the need for large annotated datasets.
- **Exploration of Explainable AI Techniques:** Explore the integration of explainable AI techniques to enhance model interpretability and provide insights into the features and patterns driving the COPD diagnosis. This could facilitate betterunderstanding and trust in the automated diagnostic system by healthcareprofessionals

REFERENCES

[1] Sørensen, S. B. Shaker, and M. deBruijne, Quantitative Analysis of Pulmonary Emphysema using Local BinaryPatterns, *IEEE*

[2] Quantitative Analysis of Pulmonary Emphysema Using Local Binary Patterns byLauge Sørensen*, Saher B. Shaker, and Marleen de Bruijne.

[3] Quantitative Analysis of Pulmonary Emphysema by Congregating Statistical Features.

[4] Classification and Quantification of Emphysema Using a Multi-Scale Residual Network.

[5] A CNN-Based Approach for Lung 3D-CT Registration.
[6] An Approach to Detect Chronic Obstructive Pulmonary Disease Using UWBRadar-Based Temporal and Spectral Features.
[7] Learning to quantify emphysema extent: what labels do we need? – Silas Nyboe Qrting, Jens Petersen, Laura H. Thomsen, Mathilde M, W. Wille