LUNG CANCER ADVANCED DETECTION USING DEEP LEARNING TECHNIQUES: COMPREHENSIVE IMPLEMENTATION AND DETAILED ANALYSIS

A PROJECT REPORT SUBMITTED IN
THE PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF COMPUTER APPLICATIONS
BY

S M FIRDOUS HASSAN

ROLL NO. BCA312015

REG NO. 2056 (2021-22)

UNDER THE SUPERVISION OF

Dr. Abhishek Das

Associate Professor

DEPARTMENT OF COMPTER SCIENCE & ENGINEERING

ALIAH UNIVERSITY

Action Area IIA/27, New Town, Kolkata-700 160

ALIAH UNIVERSITY
A UGC. AICTE and NCTE Approved Autonomous Institution
Under the Dept. Of Minority Affairs and Madrasah Education, Govt. of WestBengal II-A/27, Action Area II, New Town, Kolkata, West Bengal-700160, India.

CERTIFICATE

It is certified that the work contained in the project report entitled **"LUNG CANCER ADVANCED DETECTION USING DEEP LEARNING TECHNIQUES: COMPREHENSIVE IMPLEMENTATION AND DETAILED ANALYSIS"** has been carried out by **S M FIRDOUS HASSAN (Roll No. BCA213015)** student of 2021-2024 batch from the Department of Computer Science & Engineering. This is a bonafide record of the work carried out by him, under our supervision and guidance at the Department of Computer Science & Engineering, Aliah University, Action Area IIA/27, New Town, Kolkata-700 160, India. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree. This is to certify that the above declaration is true.

Head of the Department
Computer Science & Engineering
Aliah University

Dr. Abhishek Das
Associate Professor,
(Project Supervisor)
Computer Science & Engineering

APPROVAL

The following project report is hereby approved and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is to be understood that by this approval, opinion expressed or conclusion drawn therein, but approve the project report only for the purpose for which it has been submitted.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ALIAH UNIVERSITY
Action Area IIA/27, New Town, Kolkata-700 160

Head of the Department
Computer Science & Engineering
Aliah University

Dr. Abhishek Das
Associate Professor,
(Project Supervisor)
Computer Science &
Engineering

DECLARATION

I ,S M Firdous Hassan student of Bachelor of Computer Applications , Aliah University, Kolkata declare that the work entitled "LUNG CANCER ADVANCED DETECTION USING DEEP LEARNING TECHNIQUES: COMPREHENSIVE IMPLEMENTATION AND DETAILED ANALYSIS" has been completed. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree to any university.

Date:

S M FIRDOUS HASSAN
Roll No-BCA213015
Department of Computer Science & Engineering
Aliah University

ACKNOWLEDGMENT

Achievements in my life have always been possible only with the blessing of Allah and the guidance of my elders. I take this opportunity to express my deep gratitude to my family members, Dr. Abhishek Das Associate Professor, Computer Science & Engineering Department, Aliah University, for their guidance, support & constructive criticism that has always inspired me.

I accord my thanks to Dr. Abhishek Das for providing me with a well-equipped environment to carry out my research work who is also the Head of the Department.

I would also thank all my faculty members, and the technical staff of the Department of Computer Science & Engineering, Aliah University.

Date:

S M FIRDOUS HASSAN
Roll No-BCA213015
Department of Computer Science & Engineering
Aliah University

TABLE OF CONTENTS

Section	*Page No.*
Certificate	**I**
Approval	**II**
Declaration	**III**
Acknowledgement	**IV**
Table of Contents	**V-VI**
List of Figures	**VII-VIII**
Abbreviation	**IX**
Abstract	**X**
Chapter 1	**1-7**
Introduction	**1-2**
1.1 Traditional Diagnostic Methods	**2-3**
1.2 Challenges in Diagnosis	**3**
1.3 Advancements in Deep Learning	**3**
1.4 Motivation of Work	**4-5**
1.5 Objective of Work	**5**
1.6 Plan of Implementation	**6**
1.7 Structure of the Project	**6-7**
Chapter 2	**8-10**
Literature Review	**8**
2.1 Deep Learning in Medical Imaging	**8**
2.2 Convolutional Neural Networks (CNN) for Lung Cancer Detection	**9**
2.3 Transfer Learning and Pre-Trained Models	**9**
2.4 Histopathological Image Analysis	**9**
2.5 Challenges in Lung Cancer Detection	**10**
2.6 Evaluation Metrics and Validation	**10**
Chapter 3	**11-28**

Methodology	**11**
3.1 Dataset	**11**
3.2 Image Preprocessing	**12-13**
3.3 Dataset splitting	**13-14**
3.4 System Requirements	**15-16**
3.5 Convolutional Neural Networks (CNN)	**16-18**
3.6 VGG-16	**18-22**
3.7 Model Evaluation and Training	**23-27**
Flowchart	**28**
Chapter 4	**29-38**
Results and Discussion	**29**
4.1 Training Loss	**29**
4.2 Validation Loss	**29-30**
4.3 Training Accuracy	**31**
4.4 Validation Accuracy	**31-32**
4.5 Confusion Matrix	**32-34**
4.6 Classification Report	**34-35**
4.7 Sample Output	**35-36**
4.8 GUI for the Detection Model	**36-37**
Chapter 5	**38-42**
Conclusion and Future Scope	**38**
5.1 Conclusion	**38-40**
5.2 Future Scope	**40-42**
Appendix	**43-53**
References	**54-55**

LIST OF FIGURES

	Figure	*Page No.*
Figure 1.1	**Anatomy of the Lungs**	**1**
Figure 2.1	**Deep Network architecture with multiple layers**	**8**
Figure 3.1	**Images from the dataset**	**11**
Figure 3.2	**Data preparation and Augmentation**	**13**
Figure 3.3	**Test data generator**	**14**
Figure 3.4	**Trian data generator**	**14**
Figure 3.5	**CNN Model**	**16**
Figure 3.6	**CNN Model Architecture**	**18**
Figure 3.7	**VGG-16 Model**	**19**
Figure 3.8	**VGG-16 Model Architecture**	**22**
Figure 3.9	**CNN Model Training**	**25**
Figure 3.10	**VGG-16 Model Training**	**27**
Figure 4.1	**Loss and Validation Loss in CNN Model**	**30**
Figure 4.2	**Loss and Validation Loss in VGG16 Model**	**30**
Figure 4.3	**Accuracy and Validation Accuracy in CNN Model**	**32**
Figure 4.4	**Accuracy and Validation Accuracy in VGG16 Model**	**32**
Figure 4.5	**Confusion Matrix of CNN Model**	**34**
Figure 4.6	**Confusion Matrix of VGG16 Model**	**34**
Figure 4.7	**Classification Report for CNN Model**	**35**
Figure 4.8	**Classification Report for VGG16 Model**	**35**
Figure 4.9	**Sample outputs for CNN Model**	**36**
Figure 4.10	**Sample outputs for VGG16 Model**	**36**

Figure 5.1	**The performance of both model for Class 0 in Precision, Accuracy, and F1 Score**	**39**
Figure 5.2	**The performance of both model for Class 1 in Precision, Accuracy, and F1 Score**	**40**
Figure 5.3	**The performance of both model for Class 2 in Precision, Accuracy, and F1 Score**	**40**
Figure A.1	**GUI Front Screen**	**53**
Figure A.2	**GUI Image Browsing**	**53**
Figure A.3	**GUI Predicted Label with Image**	**53**

ABBREVIATION

CNN	Convolutional Neural Network
VGG-16	Visual Geometry Group 16
CT	Computed Tomography
PET	Positron Emission Tomography
AI	Artificial Intelligence
HIPAA	Health Insurance Portability and Accountability Act
IDE	Integrated Development Environment
OpenCV	Open Source Computer Vision Library
NumPy	Numerical Python
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
GUI	Graphical User Interface
PIL	Python Imaging Library
GDPR	General Data Protection Regulation
EMR	Electronic Medical Record

ABSTRACT

Lung cancer remains a leading cause of mortality worldwide, necessitating advancements in diagnostic methods to improve patient outcomes. This project aims to enhance lung cancer detection using advanced deep learning techniques. Initially, a Convolutional Neural Network (CNN) was utilized to classify histopathological images of lung tissues, identifying Adenocarcinoma, Squamous cell carcinoma, and normal tissues. To improve detection accuracy, a pre-trained VGG16 model was incorporated. VGG16, a deeper network pre-trained on the extensive ImageNet dataset, captures a wide range of features, enabling it to generalize well even with limited medical datasets. The novelty of this work lies in the comprehensive comparative analysis of the traditional CNN and VGG16 models, highlighting the latter's superior performance due to its deep architecture and extensive pre-training. Through detailed experimentation, the VGG16 model demonstrated higher detection accuracy, sensitivity, specificity, and F1-score compared to the traditional CNN. This improved performance underscores the potential of the VGG16 model in reducing diagnostic errors and offering faster, more reliable diagnoses. The project achieved significant milestones, including the development of an automated diagnostic pipeline that integrates the VGG16 model for real-time lung cancer detection. In conclusion, the successful application of the VGG16 model emphasizes its critical role in medical imaging and paves the way for its integration into clinical practice. This integration can assist pathologists in making more accurate diagnoses, ultimately leading to better patient outcomes. This project also sets the foundation for future research to explore even deeper and more sophisticated models, potentially incorporating additional types of medical images and multi-modal data for comprehensive cancer detection systems.

Keywords*: Lung Cancer, CNN, VGG16, ImageNet, Histopathological images, Deep Learning*

CHAPTER 1

INTRODUCTION

Lung cancer remains a significant global health challenge, ranking as the leading cause of cancer-related deaths worldwide. According to the World Health Organization (WHO), lung cancer accounts for approximately 2.21 million new cases and 1.8 million deaths annually (WHO, 2021). Early detection of lung cancer is crucial for effective treatment and improved survival rates, yet it remains challenging due to nonspecific symptoms and the need for accurate diagnostic methods (American Cancer Society, 2021).

Traditional diagnostic methods for lung cancer include imaging techniques such as chest X-rays, computed tomography (CT) scans, and positron emission tomography (PET) scans. Laboratory tests, including sputum cytology and biopsy, are also used to examine lung tissues for cancerous cells (National Cancer Institute, 2021). Histopathological examination of lung tissue biopsies, considered the gold standard for lung cancer diagnosis, relies heavily on the expertise of pathologists. However, manual analysis can be time-consuming and subject to human error, leading to potential misdiagnoses (Chan & Ho, 2020).

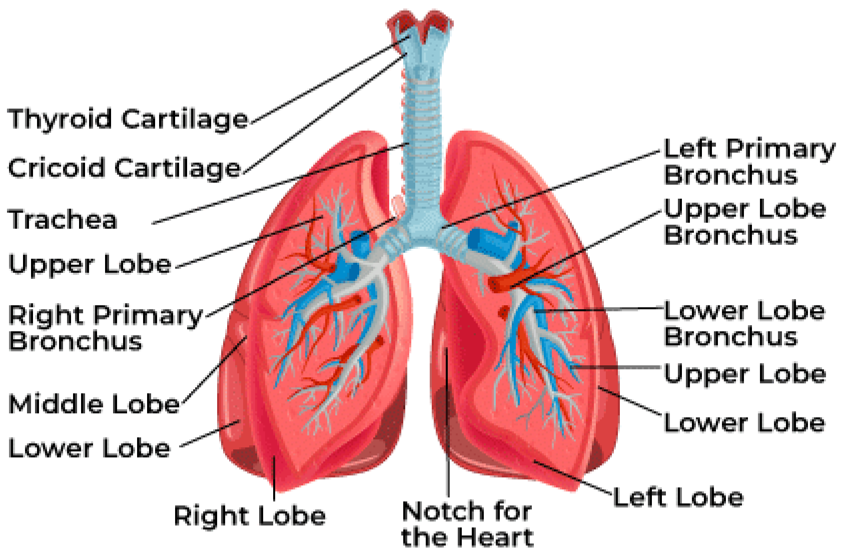


Fig.1.1: Anatomy of the Lungs

In recent years, advancements in artificial intelligence (AI) and deep learning have shown promise in revolutionizing medical diagnostics. Deep learning, a subset of machine learning, involves training neural networks on large datasets to automatically extract features and make predictions. Convolutional Neural Networks (CNNs) are particularly well-suited for image analysis due to their ability to learn spatial hierarchies of features through layers of convolutions (LeCun, Bengio, & Hinton, 2015).

This project aims to leverage deep learning techniques to enhance lung cancer detection. Initially, a custom CNN was developed to classify histopathological images of lung tissues into three categories: Adenocarcinoma, Squamous cell carcinoma, and normal tissues. Despite its effectiveness, the relatively shallow architecture of the CNN posed limitations in capturing complex patterns present in histopathological images (Litjens et al., 2017).

To address these limitations, a pre-trained VGG16 model was incorporated into the study. VGG16 is a deeper neural network pre-trained on the extensive ImageNet dataset, which contains millions of images across thousands of categories (Simonyan & Zisserman, 2014). This pre-training allows VGG16 to capture a wide range of features, making it more effective in generalizing to new tasks with limited data, such as medical image classification (Shin et al., 2016).

The novelty of this work lies in the comprehensive comparative analysis of the traditional CNN and VGG16 models. By leveraging transfer learning, VGG16 utilizes the knowledge gained from the vast ImageNet dataset, adapting it to the specific task of lung cancer detection. This comparison highlights VGG16's superior performance, attributed to its deep architecture, which allows it to capture intricate patterns and subtle differences in histopathological images more effectively than the shallower CNN (Rajpurkar et al., 2017).

1.1 TRADITIONAL DIAGNOSTIC METHODS

Traditionally, the diagnosis of lung cancer involves a combination of imaging techniques, laboratory tests, and histopathological examination:

1.1.1 Imaging Techniques:

i. Chest X-ray: Often the first imaging test performed, but it has limitations in detecting small or early-stage tumors.
ii. Computed Tomography (CT) Scan: Provides detailed cross-sectional

images of the chest, identifying tumors that may not be visible on a chest X-ray.

iii. Positron Emission Tomography (PET) Scan: Used to detect active cancer cells in the body, often combined with CT scans for precise localization.

1.1.2 Laboratory Tests:

i. Sputum Cytology: Examines mucus from the lungs under a microscope to detect cancer cells.

ii. Blood Tests: May include tests for tumor markers, although these are not specific or sensitive enough for definitive diagnosis.

1.1.3 Histopathological Examination:

i. Biopsy: The gold standard for diagnosing lung cancer. It involves obtaining a tissue sample from the suspected tumor site, typically through bronchoscopy, needle biopsy, or surgical biopsy.

ii. Cytopathology: Examination of cells from pleural effusion (fluid around the lungs) or fine-needle aspiration.

1.2 CHALLENGES IN DIAGNOSIS

The primary challenge in lung cancer diagnosis lies in the early detection of the disease. Early-stage lung cancer often presents with nonspecific symptoms or may be asymptomatic, making it difficult to identify without advanced imaging or invasive procedures. Moreover, the reliance on human expertise for interpreting histopathological images can introduce variability and potential errors in diagnosis.

1.3 ADVANCEMENTS IN DEEP LEARNING

Recent advancements in deep learning and artificial intelligence (AI) have shown promise in addressing the limitations of traditional diagnostic methods. Deep learning, a subset of machine learning, involves training neural networks on large datasets to recognize patterns and make predictions. Convolutional Neural Networks (CNNs), in particular, have demonstrated exceptional performance in image analysis tasks.

1.4 MOTIVATION OF WORK

Lung cancer continues to be one of the most formidable challenges in global health, with millions of new cases diagnosed each year and a high mortality rate that underscores the urgent need for effective diagnostic and therapeutic strategies. The critical motivation behind this project stems from several key factors:

1.4.1 High Mortality and Late Diagnosis

Lung cancer is often diagnosed at an advanced stage due to the lack of early symptoms and effective screening methods. Late-stage diagnosis significantly limits treatment options and worsens patient prognosis. By enhancing early detection capabilities through advanced technologies, we can improve survival rates and quality of life for patients.

1.4.2 Limitations of Traditional Methods

Traditional diagnostic methods, while essential, have inherent limitations. Imaging techniques like chest X-rays and CT scans can miss early-stage tumors, and the manual interpretation of histopathological images is time-consuming and subject to human error. There is a pressing need for automated, reliable, and accurate diagnostic tools to complement and enhance existing methods.

1.4.3 Advancements in Deep Learning

The advent of deep learning, particularly Convolutional Neural Networks (CNNs), offers a revolutionary approach to medical image analysis. CNNs have demonstrated remarkable success in various image classification tasks, and their application in lung cancer detection holds significant promise. Leveraging these advancements can lead to more accurate and efficient diagnostic processes.

1.4.4 Potential of Pre-trained Models

Pre-trained models such as VGG16, which have been extensively trained on large and diverse datasets, provide an opportunity to harness advanced feature extraction capabilities. These models can be fine-tuned for specific medical applications, enabling them to detect even subtle differences in lung tissue images. This transfer learning approach can dramatically improve diagnostic accuracy and reduce the need for extensive training data.

1.4.5 Improving Patient Outcomes

The ultimate motivation for this project is to improve patient outcomes. Early and accurate detection of lung cancer can lead to earlier intervention, more effective treatment options, and better prognoses. By reducing diagnostic errors and speeding up the diagnostic process, we aim to contribute to the timely and accurate treatment of lung cancer, ultimately saving lives.

1.4.6 Innovation in Medical Diagnostics

This project represents an opportunity to innovate within the field of medical diagnostics. The integration of deep learning technologies into clinical practice can set a precedent for future research and development, encouraging the adoption of AI-driven solutions in various areas of healthcare. This can lead to widespread improvements in diagnostic accuracy and efficiency across multiple medical conditions.

1.4.7 Personal and Academic Growth

On a personal and academic level, this project offers an invaluable opportunity to delve into cutting-edge technologies and their applications in real-world problems. It provides a platform to apply theoretical knowledge, develop practical skills, and contribute meaningful advancements to the field of healthcare.

1.5 OBJECTIVE OF WORK

The primary objective of this project is to develop and implement an advanced deep learning-based system for the accurate and efficient detection of lung cancer from histopathological images. This overarching goal can be broken down into several specific objectives:

i. Create and optimize a CNN for classifying lung tissue images.
ii. Fine-tune the pre-trained VGG16 model for enhanced lung cancer detection.
iii. Analyze and compare the performance of CNN and VGG16 models.
iv. Test models with real histopathological images to ensure reliability.
v. Evaluate improvements in diagnostic accuracy and efficiency.

1.6 PLAN OF IMPLEMENTATION

i. Conduct a comprehensive literature review on lung cancer detection and relevant deep learning techniques. Gather and preprocess a diverse set of histopathological images from reputable medical databases.

ii. Design and implement an initial Convolutional Neural Network (CNN) model. Train the CNN on the collected dataset and optimize for accuracy and performance.

iii. Integrate the pre-trained VGG16 model using transfer learning techniques. Fine-tune the VGG16 model on the lung cancer dataset to adapt it for the specific classification task.

iv. Train both the CNN and VGG16 models on the preprocessed dataset. Evaluate the models' performance using metrics such as accuracy, sensitivity, specificity, and F1-score.Compare the results of the CNN and VGG16 models to identify the more effective approach.

v. Test the models on a separate validation set of histopathological images. Validate the system's reliability and applicability in clinical settings.

vi. Analyze the diagnostic performance improvements achieved with the automated system. Optimize the models based on validation results and feedback.

1.7 STRUCTURE OF THE PROJECT

Chapter 1 outlines the global impact of lung cancer and the limitations of traditional diagnostic methods. It introduces the project's aim to enhance detection using deep learning, comparing a custom CNN with a pre-trained VGG16 model, which demonstrates superior performance due to its depth and transfer learning capabilities.

Chapter 2 reviews existing research on deep learning in medical imaging, focusing on the application of CNNs for lung cancer detection. It covers transfer learning, the use of pre-trained models like VGG16, and the challenges in histopathological image analysis. The chapter concludes with critical observations from the literature.

Chapter 3 details the dataset used, preprocessing steps, and the implementation of CNN and VGG16 models. It includes information on system requirements, model training, and evaluation metrics. A flowchart illustrates the overall methodology.

Chapter 4 presents the results of training and validating the CNN and VGG16 models. It includes detailed analyses of training and validation losses, accuracy, confusion matrices, and classification reports. The chapter also showcases sample outputs and the GUI developed for the detection model.

Chapter 5 summarizes the findings, highlighting the superior performance of the VGG16 model. It discusses the implications of these results for lung cancer diagnosis and potential future research directions.

CHAPTER 2

LITERATURE REVIEW

This chapter explains in detail the numerous lung cancer diagnosis techniques. A concise introduction highlights the fundamental concepts that aided readers comprehend this thesis. The majority of literature reviews for lung cancer diagnosis are based on machine learning and deep learning concepts. The chapter concludes with critical observations based on a review of the relevant literature.

2.1 DEEP LEARNING IN MEDICAL IMAGING

Recent advancements in deep learning have significantly impacted the field of medical imaging, particularly in the diagnosis of diseases such as lung cancer. Deep learning models, especially Convolutional Neural Networks (CNNs), have demonstrated high accuracy in image classification tasks. Research by Litjens et al. (2017) provides a comprehensive review of deep learning applications in medical image analysis, highlighting the potential of these methods to surpass traditional techniques.

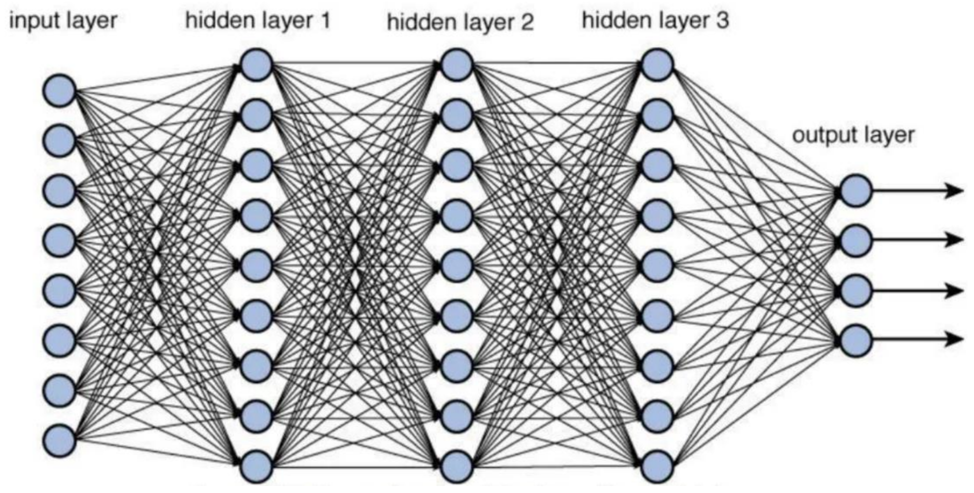


Fig.2.1: Deep Network architecture with multiple layers

2.2 CONVOLUTIONAL NEURAL NETWORKS (CNNS) FOR LUNG CANCER DETECTION

CNNs have been widely used for lung cancer detection due to their ability to automatically extract relevant features from medical images. A study by Shen et al. (2017) employed a deep CNN to classify lung nodules in CT images, achieving significant improvements in diagnostic accuracy. Similarly, Hua et al. (2015) developed a CNN-based framework for lung nodule classification, demonstrating the effectiveness of deep learning in reducing false positives and improving detection rates.

2.3 TRANSFER LEARNING AND PRE-TRAINED MODELS

Transfer learning involves using pre-trained models on large datasets and fine-tuning them for specific tasks. The VGG16 model, introduced by Simonyan and Zisserman (2014), is one of the most popular architectures used in transfer learning. Pre-trained on the ImageNet dataset, VGG16 has been successfully applied to various medical imaging tasks. Research by Kumar et al. (2018) utilized VGG16 for breast cancer classification in histopathological images, achieving high accuracy and demonstrating the model's capability to generalize across different types of medical images.

2.4 HISTOPATHOLOGICAL IMAGE ANALYSIS

Histopathological analysis is crucial for the definitive diagnosis of lung cancer. Traditional methods involve manual examination by pathologists, which can be time-consuming and prone to inter-observer variability. Deep learning approaches offer automated solutions that can assist pathologists by providing consistent and accurate image analysis. Coudray et al. (2018) developed a deep learning model for classifying lung adenocarcinoma and squamous cell carcinoma from whole-slide images, achieving performance comparable to experienced pathologists.

2.5 CHALLENGES IN LUNG CANCER DETECTION

Despite the advancements, several challenges remain in lung cancer detection using deep learning. These include the need for large annotated datasets, variability in imaging protocols, and the complexity of differentiating between benign and malignant lesions. A review by Murphy et al. (2016) discusses these challenges and highlights the importance of robust data preprocessing, augmentation techniques, and ensemble methods to improve model performance.

2.6 EVALUATION METRICS AND VALIDATION

Evaluating the performance of deep learning models in medical imaging requires appropriate metrics such as accuracy, sensitivity, specificity, and the F1-score. The work of Esteva et al. (2017) emphasizes the importance of rigorous validation using independent test sets and cross-validation to ensure the reliability and generalizability of the models. Their research in dermatology using deep learning serves as a benchmark for evaluating similar models in other medical domains, including lung cancer.

CHAPTER 3

METHODOLOGY

3.1 DATASET

The dataset utilized in this project has been curated from Kaggle, a prominent platform for machine learning datasets. The dataset, titled "Lung and Colon Cancer Histopathological Images.," provides us a diverse collection of labelled images, each representing distinct histopathological patterns associated with lung cancer. This dataset contains 15,000 histopathological images with 3 classes. All images are 768 x 768 pixels in size and are in jpeg file format. The images were generated from an original sample of HIPAA compliant and validated sources, consisting of 750 total images of lung tissue (250 benign lung tissue, 250 lung adenocarcinomas, and 250 lung squamous cell carcinomas).

Original Article : Borkowski AA, Bui MM, Thomas LB, Wilson CP, DeLand LA, Mastorides SM. Lung and Colon Cancer Histopathological Image Dataset (LC25000). arXiv:1912.12142v1 [eess.IV], 2019

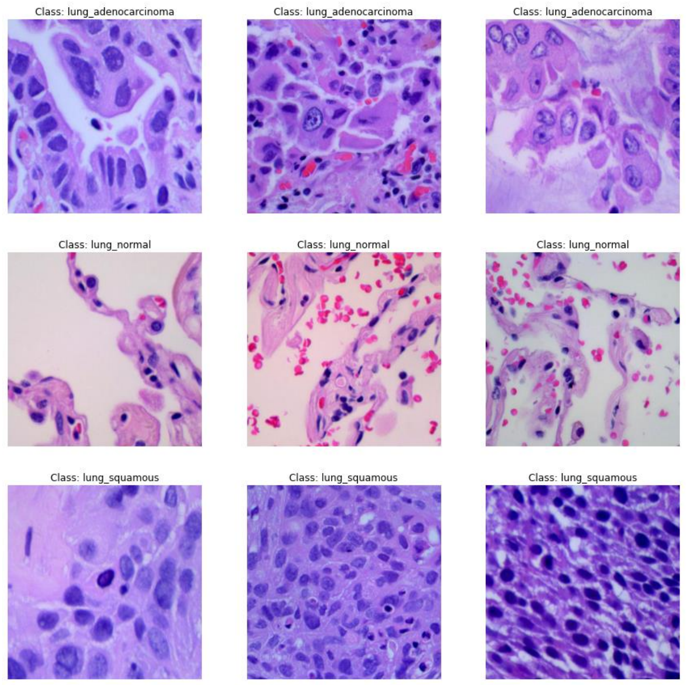


Fig.3.1: Images from the dataset

3.2 IMAGE PREPROCESSING

Before feeding the images into the Convolutional Neural Network (CNN) and the VGG16 Model, several preprocessing steps are applied to ensure that the input data is in a suitable format for the model. These preprocessing steps are crucial for enhancing the performance and generalization capability of the model. The primary preprocessing steps include resizing, normalization, and data augmentation.

3.2.1 Resizing

Standardization of Image Dimensions: The original histopathological images vary in size and resolution. To ensure uniformity and compatibility with the CNN architecture, all images are resized to a standard dimension of (128, 128) pixels. This resizing process helps to maintain a consistent input size, which is essential for the convolutional operations in the network. Additionally, resizing helps in reducing computational load and memory usage during training and inference.

3.2.2 Normalization

Scaling Pixel Values: Normalization involves scaling the pixel values of the images to a specific range, typically [0, 1]. Each pixel value in the original images ranges from 0 to 255. By dividing each pixel value by 255, the values are normalized to the range [0, 1]. This step is crucial because it helps in speeding up the convergence of the neural network during training. Normalized data ensures that the gradients do not explode or vanish during backpropagation, leading to more stable and efficient training.

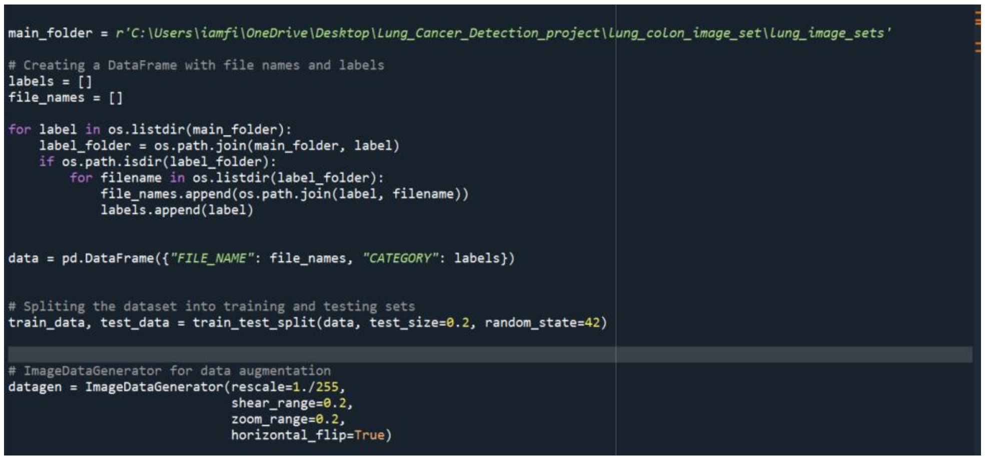
3.2.3 Data Augmentation

Artificially Increasing Dataset Size: Data augmentation is a technique used to artificially expand the size and diversity of the training dataset by applying various transformations to the original images. This process helps in improving the model's ability to generalize to new, unseen data. The following augmentation techniques are employed:

i. Shear Transformation: Shearing involves shifting one part of an image in a direction that is not parallel to the axis, effectively changing the shape of objects within the image. This transformation helps the model learn to recognize objects that may be tilted or slanted.

ii. Zoom Transformation: Zooming involves either magnifying or shrinking the image content. This transformation allows the model to become invariant to the scale of the objects within the images, making it robust to variations in object size.

iii. Horizontal Flip: Flipping the images horizontally helps in creating mirror images of the original dataset. This technique is particularly useful in making the model invariant to the orientation of the objects, ensuring that it can recognize features regardless of their left-right orientation.



```
main_folder = r'C:\Users\iamfi\OneDrive\Desktop\Lung_Cancer_Detection_project\Lung_colon_image_set\lung_image_sets'

# Creating a DataFrame with file names and labels
labels = []
file_names = []

for label in os.listdir(main_folder):
    label_folder = os.path.join(main_folder, label)
    if os.path.isdir(label_folder):
        for filename in os.listdir(label_folder):
            file_names.append(os.path.join(label, filename))
            labels.append(label)


data = pd.DataFrame({"FILE_NAME": file_names, "CATEGORY": labels})


# Spliting the dataset into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)


# ImageDataGenerator for data augmentation
datagen = ImageDataGenerator(rescale=1./255,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True)
```

Fig.3.2: Data preparation and Augmentation

3.3 DATASET SPLITTING

Before training the deep learning models, the available dataset is divided into two distinct subsets: the training set and the testing set.

3.3.1 Training Set

i. Purpose: The training set is used to train the deep learning models (CNN and VGG16) on a large number of labeled histopathological lung tissue images.

ii. Size: Typically, around 70-80% of the total dataset is allocated to the training set to ensure an adequate amount of data for model training.

iii. Randomization: The samples in the training set are randomly selected to avoid any bias in the model training process.

iv. Label Distribution: Ensure that the training set contains a proportional representation of all classes (Adenocarcinoma, Squamous cell carcinoma, and Normal tissues) to prevent class imbalances.

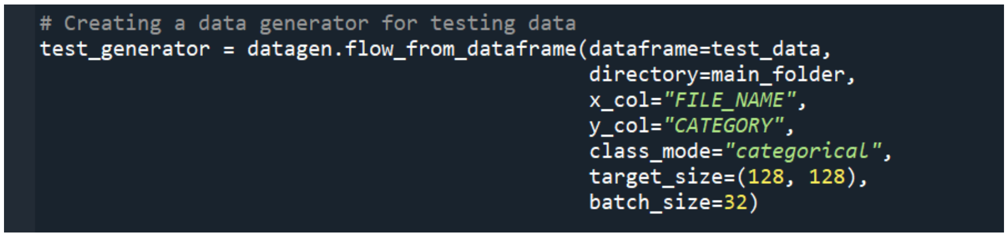
3.3.2 Testing Set

i. Purpose: The testing set is reserved for evaluating the performance and generalization capability of the trained models.

ii. Size: The remaining 20-30% of the dataset is allocated to the testing set.

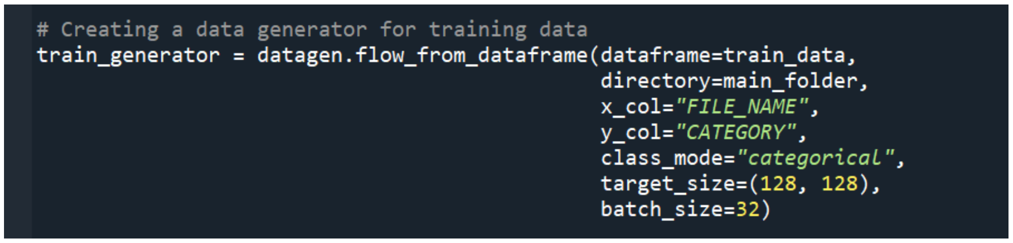
iii. Randomization: Similar to the training set, the samples in the testing set are randomly selected to ensure unbiased evaluation.

iv. Label Distribution: Like the training set, the testing set should also have a balanced representation of all classes to ensure fair evaluation across different categories of lung tissue.



```python
# Creating a data generator for testing data
test_generator = datagen.flow_from_dataframe(dataframe=test_data,
                                             directory=main_folder,
                                             x_col="FILE_NAME",
                                             y_col="CATEGORY",
                                             class_mode="categorical",
                                             target_size=(128, 128),
                                             batch_size=32)
```

Fig.3.3: Test data generation



```python
# Creating a data generator for training data
train_generator = datagen.flow_from_dataframe(dataframe=train_data,
                                              directory=main_folder,
                                              x_col="FILE_NAME",
                                              y_col="CATEGORY",
                                              class_mode="categorical",
                                              target_size=(128, 128),
                                              batch_size=32)
```

Fig.3.4: Train data generation

3.4 SYSTEM REQUIREMENTS

3.4.1 Hardware Requirements:

i. CPU: A multi-core processor (e.g., Intel Core i5 or higher) for smooth execution of Python scripts and data preprocessing tasks.

ii. RAM: A minimum of 8 GB RAM is recommended to handle the computational load of deep learning model training and data manipulation efficiently.

iii. Storage: Adequate storage space (SSD or HDD) to store project files, datasets, and model checkpoints. SSDs are preferred for faster read/write speeds, especially when working with large datasets.

3.4.2 Software Requirements:

i. Operating System: The project can be developed on various operating systems, including Windows, macOS, or Linux distributions (e.g., Ubuntu).

ii. Python: The latest version of Python (preferably Python 3.x) to develop and run the project. Anaconda is used for easy management of Python environments and dependencies.

iii. Spyder IDE: Spyder is a free and open-source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

iv. Deep Learning Frameworks: Deep learning frameworks such as TensorFlow and Keras, are used for this particular project. Spyder seamlessly integrates with these frameworks, allowing us to develop and train deep learning models directly within the IDE.

v. Image Processing Libraries: Install image processing libraries like OpenCV and Pillow to handle image loading, preprocessing, and augmentation tasks within Spyder.

vi. Additional Libraries: Additional Python libraries required for data manipulation, visualization, and model evaluation, such as NumPy, Matplotlib, and Scikit-learn are used.

3.5 CONVOLUTIONAL NEURAL NETWORK (CNN)

A Convolutional Neural Network(CNN) is a class of Deep Learning architecture which specializes in extracting features from complex visual data (here histopathological images) making them suitable for the task of image classification and recognition. CNN is used to identify subtle patterns and spatial hierarchies from the histopathological images provided to classify them into the subtypes. The model does so by applying various convolution filters to the input images and analyze them. The filters helps in recognizing distinct features relevant to adenocarcinoma, squamous and normal lung tissue. The subsequent layers , maxpooling and dense layers with a small dropout feature helps the model to further learn more about the input images thus helping the model to make more accurate predictions based on the learned process.

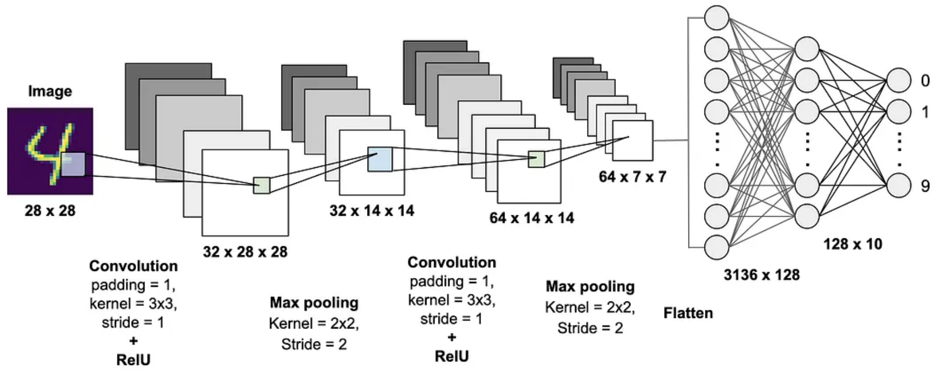


Fig.3.5: CNN Model

The CNN architecture is composed of several layers, each with a specific function in processing the input images:

3.5.1 Input Layer: The model takes input images of size (128, 128, 3), representing 128x128 pixels with three color channels (RGB).

3.5.2 First Convolutional Layer (conv2d): This layer has 32 filters with a size of 3x3, resulting in an output shape of (126, 126, 32). It applies convolution operations to extract low-level features like edges and textures. The number of trainable parameters is 896.

3.5.3 First Max Pooling Layer (max_pooling2d): This layer reduces the spatial dimensions of the feature maps by performing downsampling. The output shape becomes (63, 63, 32), which helps in reducing computational complexity and preventing overfitting. It has no trainable parameters.

3.5.4 Second Convolutional Layer (conv2d_1): This layer has 64 filters of size 3x3, resulting in an output shape of (61, 61, 64). It extracts more complex features from the input data. The number of trainable parameters is 18,496.

3.5.5 Second Max Pooling Layer (max_pooling2d_1): This layer further reduces the spatial dimensions to (30, 30, 64). It has no trainable parameters.

3.5.6 Third Convolutional Layer (conv2d_2): This layer has 128 filters of size 3x3, producing an output shape of (28, 28, 128). It captures even more intricate patterns in the images. The number of trainable parameters is 73,856.

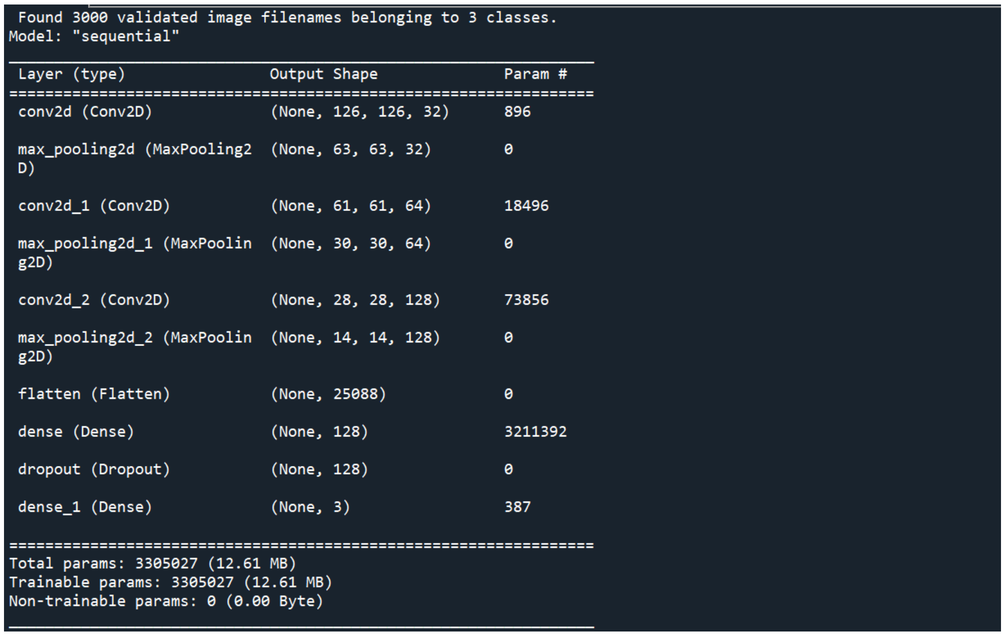
3.5.7 Third Max Pooling Layer (max_pooling2d_2): This layer downsamples the feature maps to (14, 14, 128). It has no trainable parameters.

3.5.8 Flatten Layer: This layer flattens the 3D feature maps into a 1D vector with a shape of (25088), preparing the data for the fully connected layers.

3.5.9 First Fully Connected Layer (dense): This dense layer has 128 neurons and 3,211,392 trainable parameters. It learns complex combinations of features extracted by the convolutional layers.

3.5.10 Dropout Layer (dropout): This layer randomly sets a fraction of input units to zero during training to prevent overfitting.

3.5.11 Output Layer (dense_1): The final dense layer has 3 neurons (one for each class) with a softmax activation function, producing the classification probabilities. The number of trainable parameters is 387.



```
 Found 3000 validated image filenames belonging to 3 classes.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2  (None, 63, 63, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 128)       0
 g2D)

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 128)               3211392

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 3305027 (12.61 MB)
Trainable params: 3305027 (12.61 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig.3.6: CNN Model Architecture

3.6 VGG – 16

The VGG-16 model is a convolutional neural network (CNN) architecture introduced by the Visual Geometry Group (VGG) at the University of Oxford. Known for its depth, VGG-16 consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. The model is celebrated for its simplicity and effectiveness, demonstrating strong performance across a range of computer vision tasks such as image classification and object recognition. The architecture of VGG-16 features a series of convolutional layers interspersed with max-pooling layers, which progressively increase in depth. This layered design enables the model to learn complex hierarchical representations of visual features, resulting in robust and accurate predictions. Despite the advent of more advanced architectures, VGG-16 remains a favored choice for many deep learning applications due to its versatility and exceptional performance.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition that focuses on computer vision tasks, including object localization and image classification. In 2014, VGG-16, developed by Karen Simonyan and Andrew Zisserman, secured top rankings in both tasks. The model excelled in detecting objects from 200 classes and classifying images into 1000 categories, underscoring its capabilities in large-scale visual recognition.

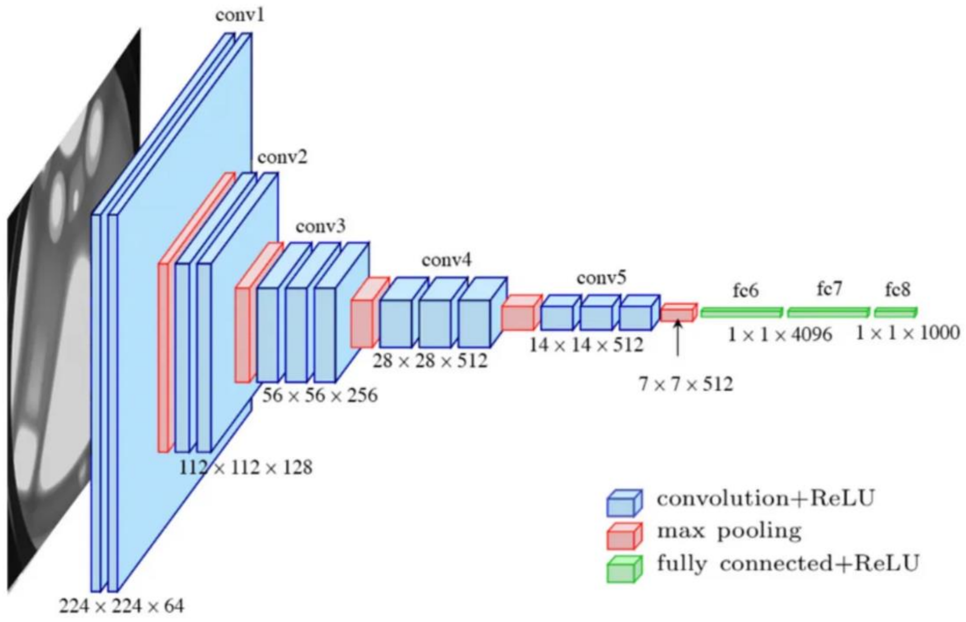


Fig.3.7: VGG-16 Model

3.6.1 Input Layer (input_1): Takes input images of size (128, 128, 3).

3.6.2 Block 1:

i. First Convolutional Layer (block1_conv1): 64 filters of size 3x3, output shape (128, 128, 64), with 1,792 parameters.

ii. Second Convolutional Layer (block1_conv2): 64 filters of size 3x3, output shape (128, 128, 64), with 36,928 parameters.

iii. Max Pooling Layer (block1_pool): Reduces the spatial dimensions to (64, 64, 64).

3.6.3 Block 2:

i. First Convolutional Layer (block2_conv1): 128 filters of size 3x3, output shape (64, 64, 128), with 73,856 parameters.
ii. Second Convolutional Layer (block2_conv2): 128 filters of size 3x3, output shape (64, 64, 128), with 147,584 parameters.
iii. Max Pooling Layer (block2_pool): Reduces the spatial dimensions to (32, 32, 128).

3.6.4 Block 3:

i. First Convolutional Layer (block3_conv1): 256 filters of size 3x3, output shape (32, 32, 256), with 295,168 parameters.
ii. Second Convolutional Layer (block3_conv2): 256 filters of size 3x3, output shape (32, 32, 256), with 590,080 parameters.
iii. Third Convolutional Layer (block3_conv3): 256 filters of size 3x3, output shape (32, 32, 256), with 590,080 parameters.
iv. Max Pooling Layer (block3_pool): Reduces the spatial dimensions to (16, 16, 256).

3.6.5 Block 4:

i. First Convolutional Layer (block4_conv1): 512 filters of size 3x3, output shape (16, 16, 512), with 1,180,160 parameters.
ii. Second Convolutional Layer (block4_conv2): 512 filters of size 3x3, output shape (16, 16, 512), with 2,359,808 parameters.
iii. Third Convolutional Layer (block4_conv3): 512 filters of size 3x3, output shape (16, 16, 512), with 2,359,808 parameters.
iv. Max Pooling Layer (block4_pool): Reduces the spatial dimensions to (8, 8, 512).

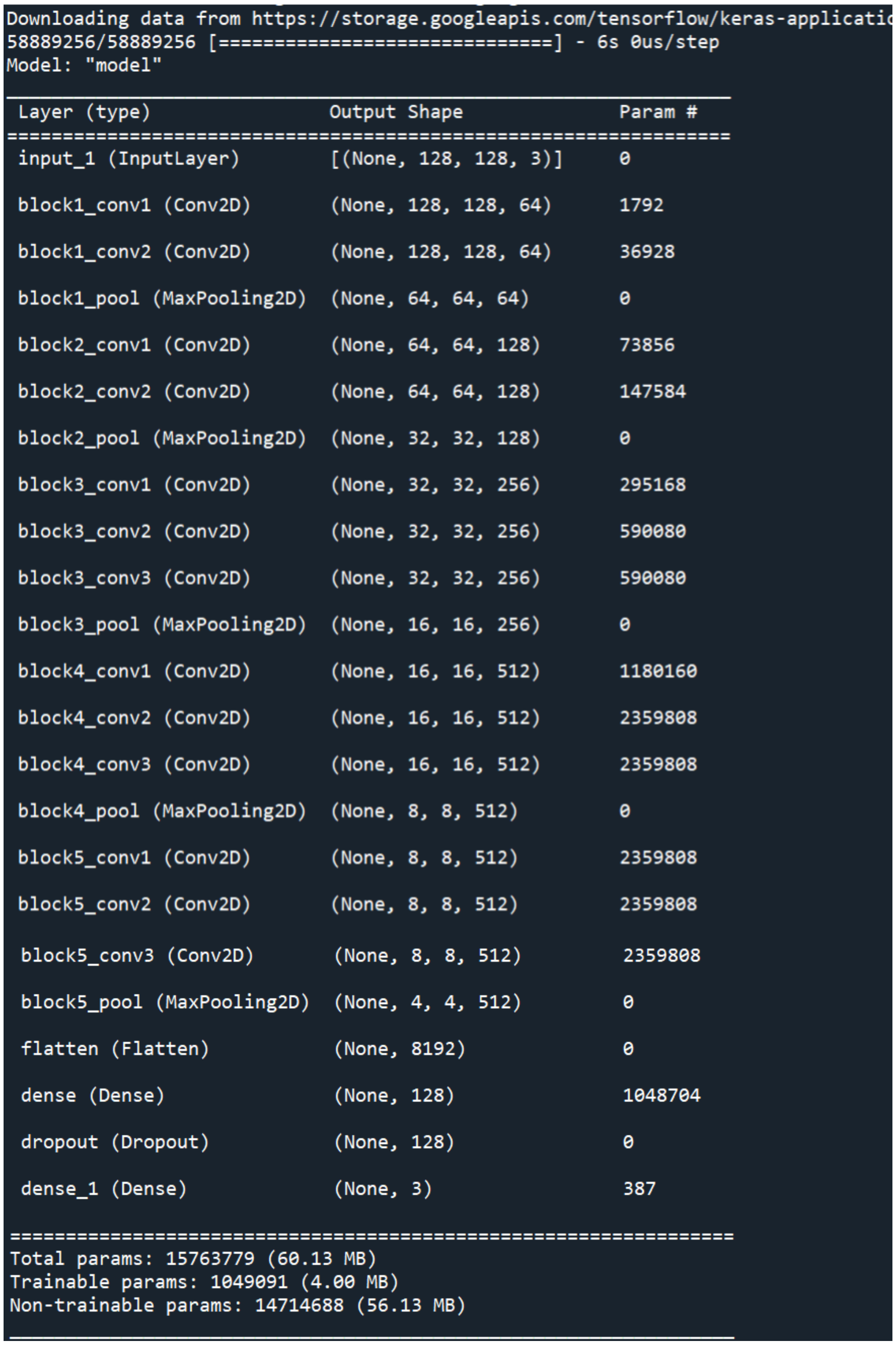
3.6.6 Block 5:

i. First Convolutional Layer (block5_conv1): 512 filters of size 3x3, output shape (8, 8, 512), with 2,359,808 parameters.
ii. Second Convolutional Layer (block5_conv2): 512 filters of size 3x3, output shape (8, 8, 512), with 2,359,808 parameters.
iii. Third Convolutional Layer (block5_conv3): 512 filters of size 3x3, output shape (8, 8, 512), with 2,359,808 parameters.
iv. Max Pooling Layer (block5_pool): Reduces the spatial dimensions to (4, 4, 512).
v. Flatten Layer (flatten): Converts the 3D feature maps into a 1D vector of size (8192), preparing the data for the fully connected layers.

3.6.7 First Fully Connected Layer (dense): Contains 128 neurons with 1,048,764 parameters, learning complex feature combinations from the extracted features.

3.6.8 Dropout Layer (dropout): Reduces overfitting by randomly setting some neurons to zero during training.

3.6.9 Output Layer (dense_1): The final layer has 3 neurons corresponding to the three classes (Adenocarcinoma, Squamous cell carcinoma, and Normal tissues), with a softmax activation function to provide class probabilities.



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
58889256/58889256 [==============================] - 6s 0us/step
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 128, 128, 3)]     0

 block1_conv1 (Conv2D)       (None, 128, 128, 64)      1792

 block1_conv2 (Conv2D)       (None, 128, 128, 64)      36928

 block1_pool (MaxPooling2D)  (None, 64, 64, 64)        0

 block2_conv1 (Conv2D)       (None, 64, 64, 128)       73856

 block2_conv2 (Conv2D)       (None, 64, 64, 128)       147584

 block2_pool (MaxPooling2D)  (None, 32, 32, 128)       0

 block3_conv1 (Conv2D)       (None, 32, 32, 256)       295168

 block3_conv2 (Conv2D)       (None, 32, 32, 256)       590080

 block3_conv3 (Conv2D)       (None, 32, 32, 256)       590080

 block3_pool (MaxPooling2D)  (None, 16, 16, 256)       0

 block4_conv1 (Conv2D)       (None, 16, 16, 512)       1180160

 block4_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 128)               1048704

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 15763779 (60.13 MB)
Trainable params: 1049091 (4.00 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

Fig.3.8: VGG-16 Model Architecture

3.7 MODEL EVALUATION AND TRAINING

3.7.1 Convolutional Neural Network (CNN) Training and Evaluation:

3.7.1.1 Training Process: In the training phase of the CNN model, several crucial steps are undertaken to prepare the model for effective learning and classification:

i. Data Preprocessing: Images are subjected to a series of preprocessing steps to ensure uniformity and numerical stability. This involves resizing images to a standardized format, typically 128x128 pixels, and normalizing pixel values to a range between 0 and 1. Additionally, data augmentation techniques are applied to diversify the dataset, including shearing, zooming, and horizontal flipping. These techniques introduce variations in the dataset, augmenting it with additional samples and enhancing the model's ability to generalize to unseen data.

ii. Dataset Split: The dataset is partitioned into training and testing sets, typically using an 80:20 or 70:30 split ratio. The training set, comprising the majority of the data, is used to train the model, while the testing set is reserved for evaluating the model's performance on unseen data. This partitioning ensures that the model's performance is accurately assessed and validated on independent data samples.

iii. Model Architecture: The CNN architecture is carefully designed, comprising a series of sequential layers optimized for image classification tasks. These layers typically include convolutional layers, responsible for detecting features from input images, followed by pooling layers that reduce spatial dimensions and extract dominant features. Fully connected layers at the end of the network perform classification based on the extracted features. The architecture is optimized to strike a balance between model complexity and performance, ensuring efficient learning and classification.

iv. Optimization: During training, the model's parameters are optimized using an optimization algorithm such as Adam. This algorithm adjusts the model's weights and biases iteratively to minimize the loss function, which measures the disparity between predicted and actual labels. The categorical cross-entropy loss function is commonly used for multi-class classification tasks, penalizing incorrect predictions and guiding the model towards accurate classification.

v. Training Configuration: Key training parameters, such as batch size and the number of epochs, are configured to optimize model convergence and generalization. The batch size determines the number of samples processed in each iteration, with larger batch sizes often leading to faster convergence but requiring more memory. The number of epochs specifies the number of times the entire training dataset is passed through the network during training. Additionally, early stopping mechanisms may be employed to halt training when the model's performance on the validation set no longer improves, preventing overfitting and promoting generalization.

vi. Validation: Validation is a critical component of the training process, enabling the monitoring of the model's performance and preventing overfitting.

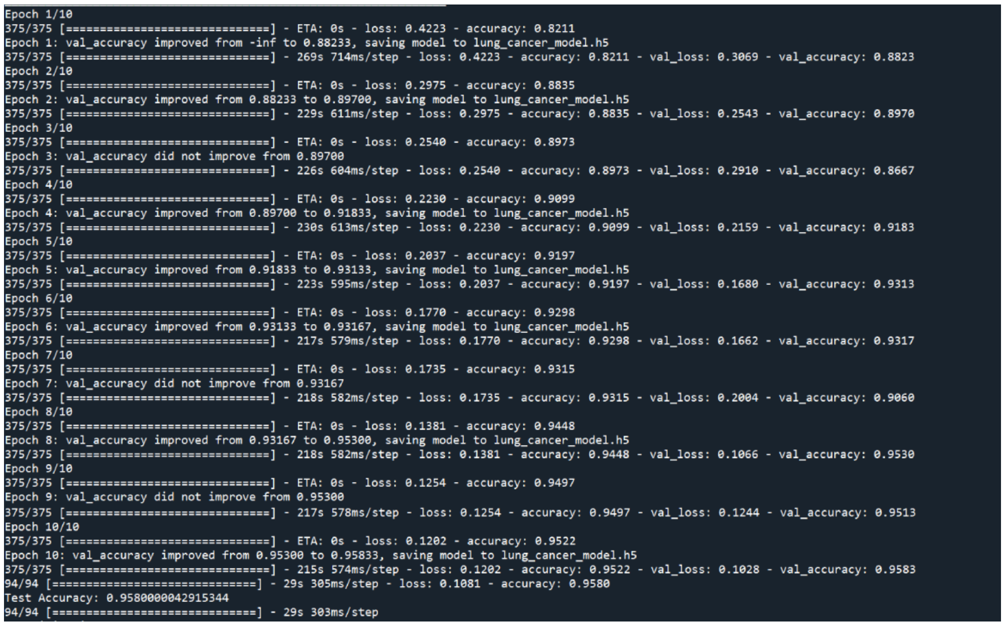
vii. Validation Set: A portion of the training data, known as the validation set, is set aside for evaluating the model's performance during training. After each epoch, the model's performance is assessed on the validation set to track its progress and identify signs of overfitting. This independent evaluation ensures that the model's performance metrics are unbiased and accurately reflect its ability to generalize to unseen data.

viii. Metrics: Various performance metrics, including accuracy, precision, recall, and F1-score, are computed during validation to quantify the model's classification performance. These metrics provide insights into the model's ability to correctly classify different classes and identify any areas of improvement. By monitoring these metrics throughout the training process, adjustments can be made to the model's architecture or training parameters to optimize its performance.

3.7.1.2 Model Evaluation: Once training is complete, the trained CNN model is evaluated on the test set to assess its generalization performance:

i. Test Set Evaluation: The test set, consisting of unseen data samples, is used to evaluate the model's performance in a real-world scenario. The model's predictions are compared against the ground truth labels in the test set, and various performance metrics are computed to quantify its accuracy and effectiveness. This evaluation provides valuable insights into the model's ability to generalize to new, unseen data and its overall performance in practical applications.

ii. Confusion Matrix: In addition to performance metrics, a confusion matrix is often generated to provide a detailed breakdown of the model's classification results. The confusion matrix displays the number of true positive, true negative, false positive, and false negative predictions for each class, offering insights into the model's strengths and weaknesses. By analyzing the confusion matrix, potential sources of misclassification can be identified, enabling further refinement of the model's architecture or training process.



```
Epoch 1/10
375/375 [==============================] - ETA: 0s - loss: 0.4223 - accuracy: 0.8211
Epoch 1: val_accuracy improved from -inf to 0.88233, saving model to lung_cancer_model.h5
375/375 [==============================] - 269s 714ms/step - loss: 0.4223 - accuracy: 0.8211 - val_loss: 0.3069 - val_accuracy: 0.8823
Epoch 2/10
375/375 [==============================] - ETA: 0s - loss: 0.2975 - accuracy: 0.8835
Epoch 2: val_accuracy improved from 0.88233 to 0.89700, saving model to lung_cancer_model.h5
375/375 [==============================] - 229s 611ms/step - loss: 0.2975 - accuracy: 0.8835 - val_loss: 0.2543 - val_accuracy: 0.8970
Epoch 3/10
375/375 [==============================] - ETA: 0s - loss: 0.2540 - accuracy: 0.8973
Epoch 3: val_accuracy did not improve from 0.89700
375/375 [==============================] - 226s 604ms/step - loss: 0.2540 - accuracy: 0.8973 - val_loss: 0.2910 - val_accuracy: 0.8667
Epoch 4/10
375/375 [==============================] - ETA: 0s - loss: 0.2230 - accuracy: 0.9099
Epoch 4: val_accuracy improved from 0.89700 to 0.91833, saving model to lung_cancer_model.h5
375/375 [==============================] - 230s 613ms/step - loss: 0.2230 - accuracy: 0.9099 - val_loss: 0.2159 - val_accuracy: 0.9183
Epoch 5/10
375/375 [==============================] - ETA: 0s - loss: 0.2037 - accuracy: 0.9197
Epoch 5: val_accuracy improved from 0.91833 to 0.93133, saving model to lung_cancer_model.h5
375/375 [==============================] - 223s 595ms/step - loss: 0.2037 - accuracy: 0.9197 - val_loss: 0.1680 - val_accuracy: 0.9313
Epoch 6/10
375/375 [==============================] - ETA: 0s - loss: 0.1770 - accuracy: 0.9298
Epoch 6: val_accuracy improved from 0.93133 to 0.93167, saving model to lung_cancer_model.h5
375/375 [==============================] - 217s 579ms/step - loss: 0.1770 - accuracy: 0.9298 - val_loss: 0.1662 - val_accuracy: 0.9317
Epoch 7/10
375/375 [==============================] - ETA: 0s - loss: 0.1735 - accuracy: 0.9315
Epoch 7: val_accuracy did not improve from 0.93167
375/375 [==============================] - 218s 582ms/step - loss: 0.1735 - accuracy: 0.9315 - val_loss: 0.2004 - val_accuracy: 0.9060
Epoch 8/10
375/375 [==============================] - ETA: 0s - loss: 0.1381 - accuracy: 0.9448
Epoch 8: val_accuracy improved from 0.93167 to 0.95300, saving model to lung_cancer_model.h5
375/375 [==============================] - 218s 582ms/step - loss: 0.1381 - accuracy: 0.9448 - val_loss: 0.1066 - val_accuracy: 0.9530
Epoch 9/10
375/375 [==============================] - ETA: 0s - loss: 0.1254 - accuracy: 0.9497
Epoch 9: val_accuracy did not improve from 0.95300
375/375 [==============================] - 217s 578ms/step - loss: 0.1254 - accuracy: 0.9497 - val_loss: 0.1244 - val_accuracy: 0.9513
Epoch 10/10
375/375 [==============================] - ETA: 0s - loss: 0.1202 - accuracy: 0.9522
Epoch 10: val_accuracy improved from 0.95300 to 0.95833, saving model to lung_cancer_model.h5
375/375 [==============================] - 215s 574ms/step - loss: 0.1202 - accuracy: 0.9522 - val_loss: 0.1028 - val_accuracy: 0.9583
94/94 [==============================] - 29s 305ms/step - loss: 0.1081 - accuracy: 0.9580
Test Accuracy: 0.9580000042915344
94/94 [==============================] - 29s 303ms/step
```

Fig.3.9: CNN Model Training

3.7.2 VGG16 Model Training and Evaluation:

3.7.2.1 Training Process: The training process for the VGG16 model follows a similar approach to that of the CNN model, with some key differences:

i. Transfer Learning: In the training process of the VGG16 model for lung cancer detection, transfer learning is a cornerstone strategy employed to leverage the model's pre-existing knowledge. Initially trained on a vast dataset like ImageNet, VGG16 has already learned to discern intricate features from images, spanning a wide array of visual recognition tasks. During this pre-training phase, the model effectively captures high-level representations of image features, ranging from basic edges to complex textures and shapes. Transfer learning capitalizes on this acquired knowledge by repurposing the pre-trained convolutional layers of VGG16 as feature extractors. These layers remain unchanged, preserving their ability to recognize generic visual patterns. Meanwhile, custom top layers are added to the model to tailor it specifically to the nuances of lung cancer detection. These custom layers are trained from scratch, using the features extracted by the pre-trained layers as a foundation. While the pre-trained layers are initially frozen to safeguard their learned features, fine-tuning may be conducted in subsequent training epochs to further refine the model's performance. This approach not only accelerates the training process but also enhances the model's ability to accurately classify lung cancer images, making transfer learning an indispensable technique in deep learning applications.

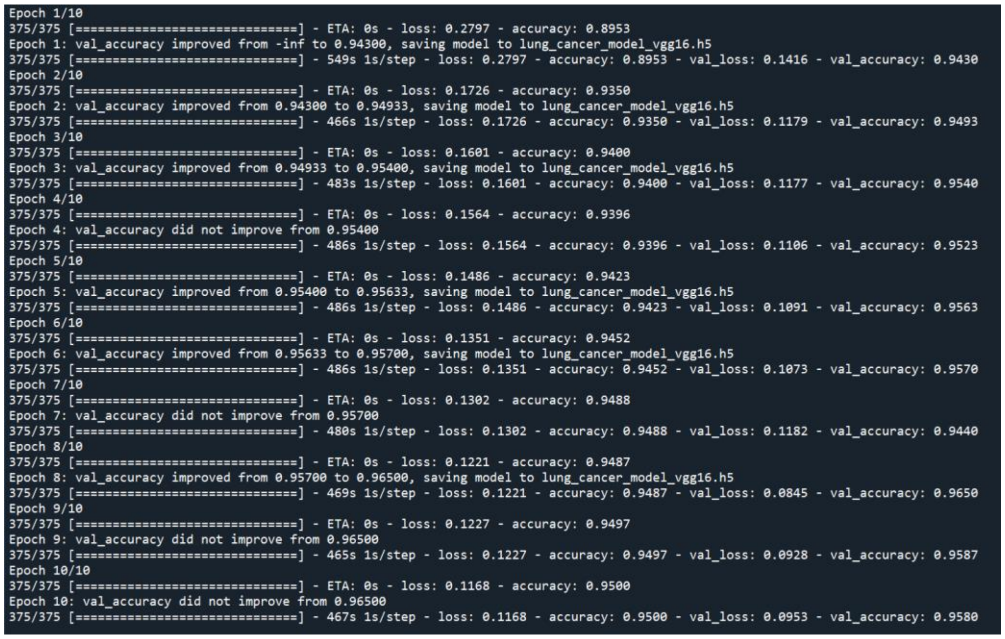
ii. Freezing Layers: During transfer learning, the parameters of the pre-trained layers in the VGG16 model are frozen to preserve the learned features and prevent them from being modified during training. Only the custom top layers added for task-specific adaptation are trained, allowing the model to retain the knowledge captured by the pre-trained layers while adapting to the nuances of the lung cancer detection task.

iii. Optimization and Training Configuration: Similar optimization algorithms, loss functions, and training configurations are applied to the VGG16 model as in the CNN model, ensuring consistency and fairness in the training and evaluation process.

3.7.2.2 Validation and Model Evaluation: Validation and model evaluation procedures for the VGG16 model closely resemble those of the CNN model:

i. Validation Set and Metrics: The VGG16 model is validated on a dedicated validation set, and various performance metrics are computed to assess its classification performance. Similar metrics such as accuracy, precision, recall, and F1-score are calculated to quantify the model's performance and identify areas for improvement.

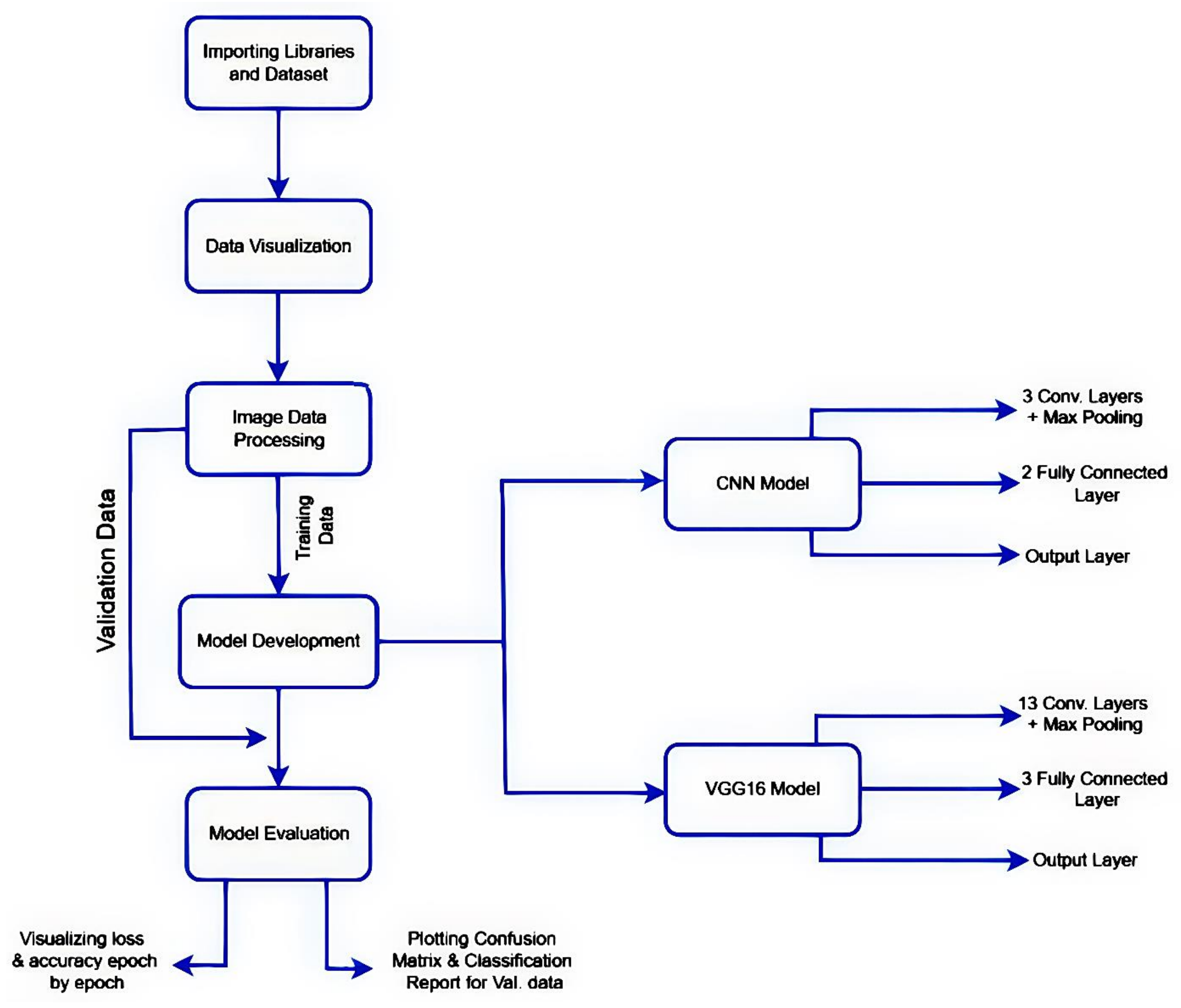
ii. Test Set Evaluation and Confusion Matrix: The VGG16 model undergoes evaluation on the test set to gauge its generalization performance and effectiveness in real-world scenarios. Performance metrics and confusion matrices are generated to provide detailed insights into the model's classification results and highlight potential areas of improvement.



```
Epoch 1/10
375/375 [==============================] - ETA: 0s - loss: 0.2797 - accuracy: 0.8953
Epoch 1: val_accuracy improved from -inf to 0.94300, saving model to lung_cancer_model_vgg16.h5
375/375 [==============================] - 549s 1s/step - loss: 0.2797 - accuracy: 0.8953 - val_loss: 0.1416 - val_accuracy: 0.9430
Epoch 2/10
375/375 [==============================] - ETA: 0s - loss: 0.1726 - accuracy: 0.9350
Epoch 2: val_accuracy improved from 0.94300 to 0.94933, saving model to lung_cancer_model_vgg16.h5
375/375 [==============================] - 466s 1s/step - loss: 0.1726 - accuracy: 0.9350 - val_loss: 0.1179 - val_accuracy: 0.9493
Epoch 3/10
375/375 [==============================] - ETA: 0s - loss: 0.1601 - accuracy: 0.9400
Epoch 3: val_accuracy improved from 0.94933 to 0.95400, saving model to lung_cancer_model_vgg16.h5
375/375 [==============================] - 483s 1s/step - loss: 0.1601 - accuracy: 0.9400 - val_loss: 0.1177 - val_accuracy: 0.9540
Epoch 4/10
375/375 [==============================] - ETA: 0s - loss: 0.1564 - accuracy: 0.9396
Epoch 4: val_accuracy did not improve from 0.95400
375/375 [==============================] - 486s 1s/step - loss: 0.1564 - accuracy: 0.9396 - val_loss: 0.1106 - val_accuracy: 0.9523
Epoch 5/10
375/375 [==============================] - ETA: 0s - loss: 0.1486 - accuracy: 0.9423
Epoch 5: val_accuracy improved from 0.95400 to 0.95633, saving model to lung_cancer_model_vgg16.h5
375/375 [==============================] - 486s 1s/step - loss: 0.1486 - accuracy: 0.9423 - val_loss: 0.1091 - val_accuracy: 0.9563
Epoch 6/10
375/375 [==============================] - ETA: 0s - loss: 0.1351 - accuracy: 0.9452
Epoch 6: val_accuracy improved from 0.95633 to 0.95700, saving model to lung_cancer_model_vgg16.h5
375/375 [==============================] - 486s 1s/step - loss: 0.1351 - accuracy: 0.9452 - val_loss: 0.1073 - val_accuracy: 0.9570
Epoch 7/10
375/375 [==============================] - ETA: 0s - loss: 0.1302 - accuracy: 0.9488
Epoch 7: val_accuracy did not improve from 0.95700
375/375 [==============================] - 480s 1s/step - loss: 0.1302 - accuracy: 0.9488 - val_loss: 0.1182 - val_accuracy: 0.9440
Epoch 8/10
375/375 [==============================] - ETA: 0s - loss: 0.1221 - accuracy: 0.9487
Epoch 8: val_accuracy improved from 0.95700 to 0.96500, saving model to lung_cancer_model_vgg16.h5
375/375 [==============================] - 469s 1s/step - loss: 0.1221 - accuracy: 0.9487 - val_loss: 0.0845 - val_accuracy: 0.9650
Epoch 9/10
375/375 [==============================] - ETA: 0s - loss: 0.1227 - accuracy: 0.9497
Epoch 9: val_accuracy did not improve from 0.96500
375/375 [==============================] - 465s 1s/step - loss: 0.1227 - accuracy: 0.9497 - val_loss: 0.0928 - val_accuracy: 0.9587
Epoch 10/10
375/375 [==============================] - ETA: 0s - loss: 0.1168 - accuracy: 0.9500
Epoch 10: val_accuracy did not improve from 0.96500
375/375 [==============================] - 467s 1s/step - loss: 0.1168 - accuracy: 0.9500 - val_loss: 0.0953 - val_accuracy: 0.9580
```

Fig.3.10: VGG-16 Model Training

Flowchart



Importing Libraries and Dataset
Data Visualization
Image Data Processing
Validation Data
Training Data
Model Development
Model Evaluation
Visualizing loss & accuracy epoch by epoch
Plotting Confusion Matrix & Classification Report for Val. data
CNN Model
3 Conv. Layers + Max Pooling
2 Fully Connected Layer
Output Layer
VGG16 Model
13 Conv. Layers + Max Pooling
3 Fully Connected Layer
Output Layer

CHAPTER 4

RESULTS AND DISCUSSION

The results of the lung cancer detection project are outlined below, providing a detailed analysis of the performance metrics for both the Convolutional Neural Network (CNN) model and the VGG16 model:

4.1 TRAINING LOSS

The training loss is a measure of how well a machine learning model is performing on the training data. It is calculated during the training process and is used to update the model's parameters through techniques like gradient descent to minimize this loss. The training loss reflects how well the model is fitting the training data. It should decrease over time as the model learns from the data. However, a very low training loss doesn't necessarily mean the model will perform well on new, unseen data, as it may have overfit the training data. In a classification problem, a common loss function is the cross-entropy loss (also known as log loss or logistic loss).

For a single data point (i), where the true label is denoted as y_i (0 or 1 for binary classification, or a one-hot encoded vector for multi-class classification) and the predicted probability of the positive class is denoted as p_i, the cross-entropy loss can be expressed as:

*L_train(i) = -[y_i * log(p_i) + (1 — y_i) * log(1 — p_i)]*

The training loss (L_train) is typically computed as the average of this loss over all training data points:

*L_train = (1/N_train) * Σ L_train(i) for i in [1, N_train]*

4.2 VALIDATION LOSS

The validation loss is a measure of how well a machine learning model is performing on a separate dataset called the validation set. During training, a portion of the data (not used for training) is set aside as the validation set. The model's performance on the validation set is evaluated periodically (after each epoch, for example), and the validation loss is calculated. The validation loss helps assess how well the model generalizes to data it hasn't seen during training. It is used to monitor the model's performance and to detect overfitting. If the validation loss starts increasing while the training loss is decreasing, it's a sign of overfitting. Similar to the training loss, the validation loss is computed using the cross-entropy loss function. For each data point in the validation set, you calculate the loss in the same way as for the training data. The validation loss (L_validation) is computed as the average of the losses over all validation data points:

*L_validation = (1/N_validation) * Σ L_validation(i) for i in [1, N_validation]*

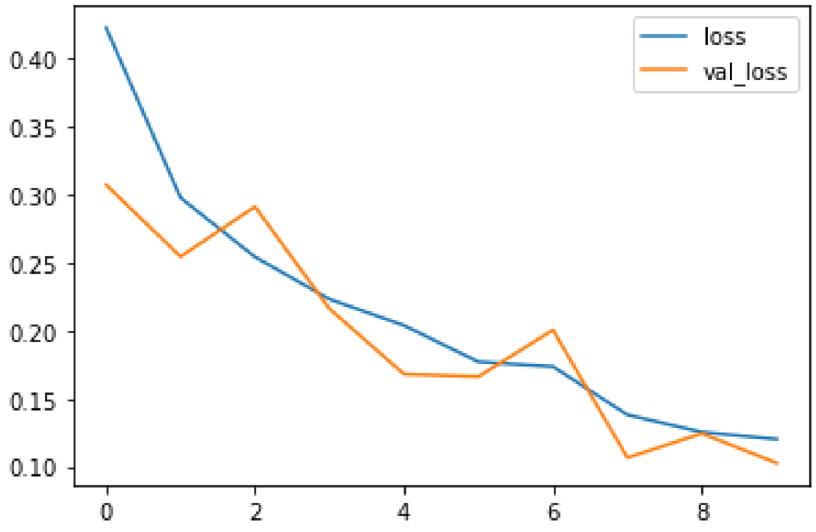


Fig.4.1: Loss and Validation Loss in CNN Model

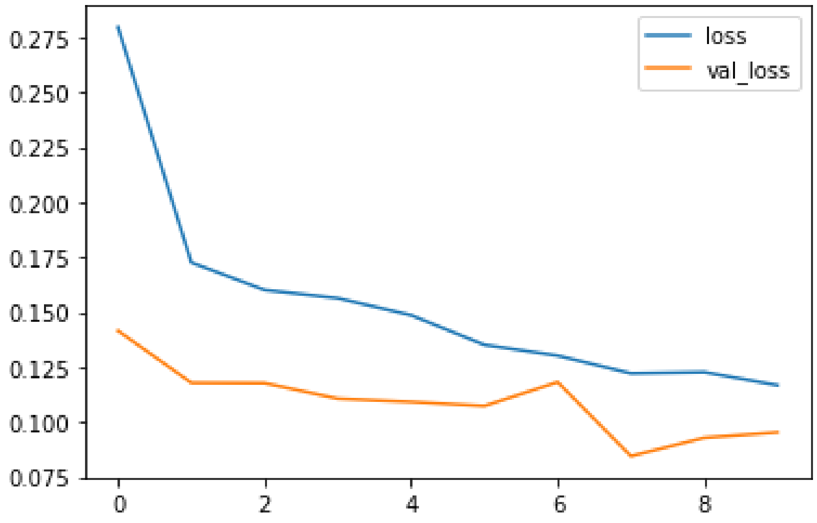


Fig.4.2: Loss and Validation Loss in VGG16 Model

4.3 TRAINING ACCURACY

Training accuracy reflects the model's performance on the training dataset by measuring the proportion of correctly classified samples. It serves as a crucial metric during the training process, indicating how well the model has learned the patterns and relationships present in the training data. Calculated as the ratio of correctly classified samples to the total number of samples in the training dataset, training accuracy provides insights into the model's ability to make accurate predictions on familiar data. A high training accuracy suggests that the model has successfully captured the underlying patterns and can classify training samples with high precision. However, it's important to note that training accuracy alone does not guarantee good performance on unseen data, as the model may have overfitted to the training dataset.

4.4 VALIDATION ACCURACY

Validation accuracy assesses the model's performance on a separate validation dataset containing samples that the model has not been exposed to during training. It serves as an estimate of the model's ability to generalize to new, unseen data. Similar to training accuracy, validation accuracy is calculated as the ratio of correctly classified samples to the total number of samples in the validation dataset. A high validation accuracy indicates that the model can generalize well beyond the training data, making accurate predictions on previously unseen instances. Consistency between training and validation accuracy is desirable, indicating that the model is neither overfitting nor underfitting. Validation accuracy plays a crucial role in model selection and optimization, helping data scientists assess the model's real-world performance and ensure its effectiveness in practical applications.

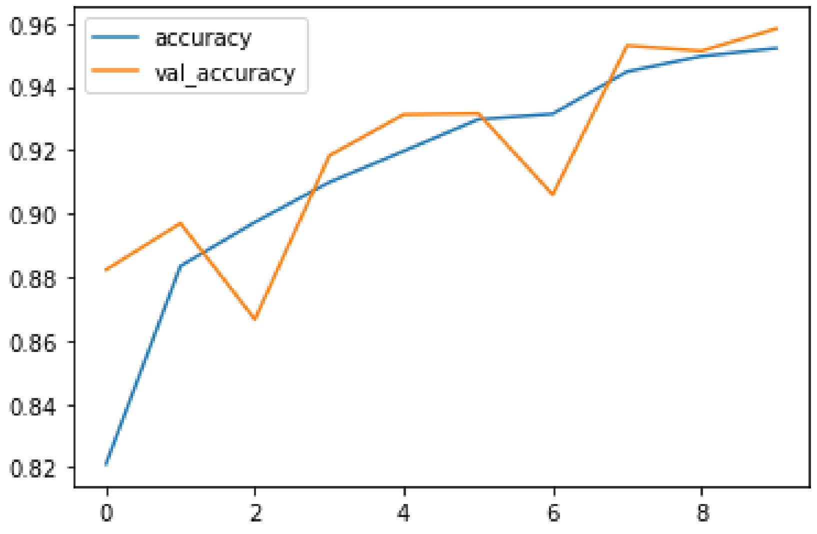


Fig.4.3:Accuracy and Validation Accuracy in CNN Model

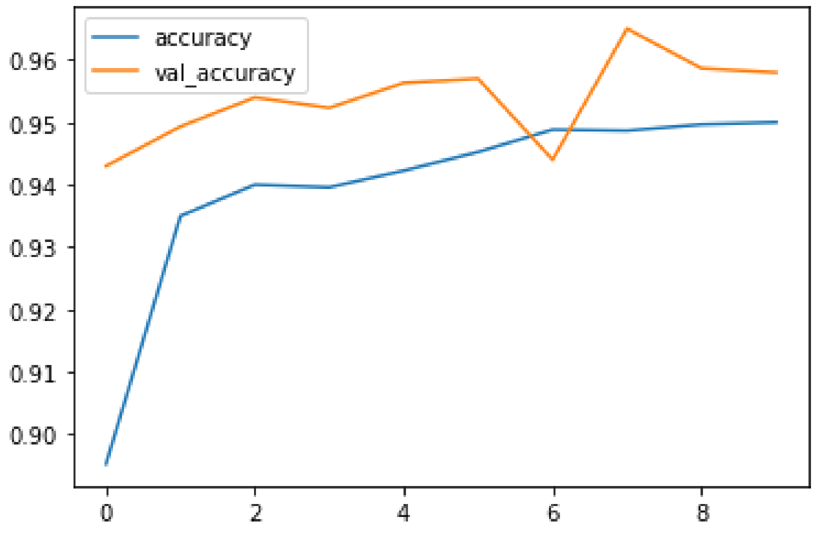


Fig.4.4:Accuracy and Validation Accuracy in VGG16 Model

4.5 CONFUSION MATRIX

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix.

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

The above table has the following cases:

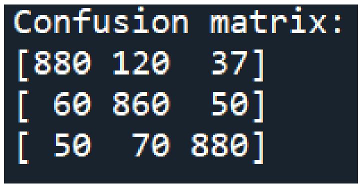
i. True Negative: Model has given prediction No, and the real or actual value was also No.
ii. True Positive: The model has predicted yes, and the actual value was also true.
iii. False Negative: The model has predicted no, but the actual value was Yes, it is also called as Type-II error.
iv. False Positive: The model has predicted Yes, but the actual value was No. It is also called a Type-I error.

By analyzing the confusion matrix, we gain insights into various aspects of the model's performance:

4.5.1 Accuracy: The overall accuracy of the model can be calculated as the sum of true positives and true negatives divided by the total number of instances. However, accuracy alone may not provide a complete picture, especially for imbalanced datasets.

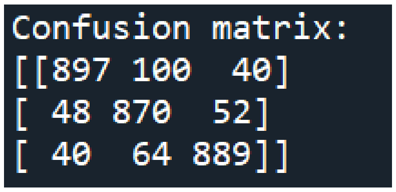
4.5.2 Precision and Recall: Precision measures the proportion of correctly predicted positive cases out of all predicted positive cases, focusing on the model's ability to avoid false positives. Recall, on the other hand, measures the proportion of correctly predicted positive cases out of all actual positive cases, highlighting the model's ability to capture all positive cases without missing any.

4.5.3 Imbalance Handling: For imbalanced datasets where one class (e.g., cancer) may have significantly fewer instances than the other class (e.g., non-cancer), the confusion matrix helps identify whether the model's predictions are biased towards the majority class or if it effectively addresses the minority class.



```
Confusion matrix:
[880 120  37]
[ 60 860  50]
[ 50  70 880]
```

Fig.4.5:Confusion Matrix of CNN Model



```
Confusion matrix:
[[897 100  40]
[ 48 870  52]
[ 40  64 889]]
```

Fig.4.6:Confusion Matrix of VGG16 Model

4.6 CLASSIFICATION REPORT

The classification report provides a comprehensive summary of the model's performance metrics for each class:

4.6.1 Precision: Precision is calculated as the ratio of true positives to the sum of true positives and false positives. It indicates the model's ability to make correct positive predictions and avoid false positives.

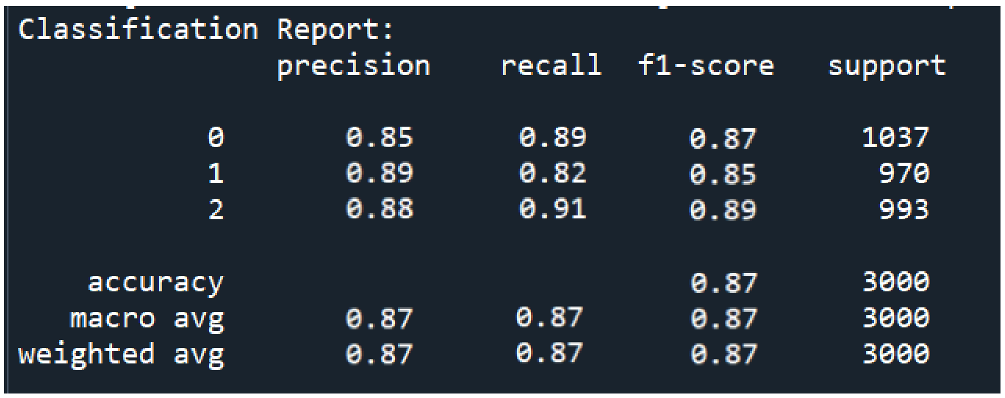
4.6.2 Recall: Recall, also known as sensitivity or true positive rate, is calculated as the ratio of true positives to the sum of true positives and false negatives. It measures the model's ability to correctly identify positive cases from all actual positive cases.

4.6.3 F1-Score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's overall performance. It considers both false positives and false negatives, making it useful for evaluating models on imbalanced datasets.

4.6.4 Support: Support refers to the number of samples for each class, providing context for the precision, recall, and F1-score metrics. It helps assess the reliability of the performance metrics, especially for classes with fewer instances.

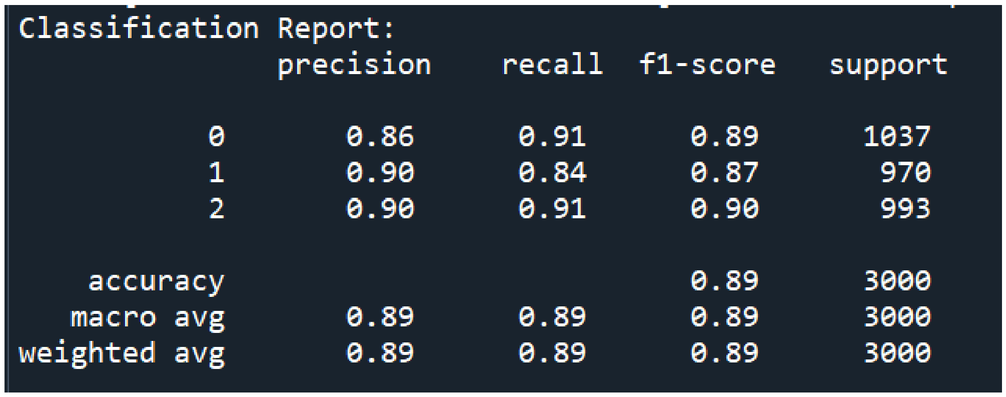
By analyzing the classification report, we gain deeper insights into the strengths and weaknesses of the model's predictions for each class. It enables us to:

i. Assess the model's precision and recall for each class, identifying areas where the model excels or requires improvement.
ii. Understand how well the model generalizes to different classes and handles imbalanced datasets.
iii. Make informed decisions about model optimization and fine-tuning to enhance its performance and reliability in real-world scenarios.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      1037
           1       0.89      0.82      0.85       970
           2       0.88      0.91      0.89       993

    accuracy                           0.87      3000
   macro avg       0.87      0.87      0.87      3000
weighted avg       0.87      0.87      0.87      3000
```

Fig.4.7:Classification Report for CNN Model



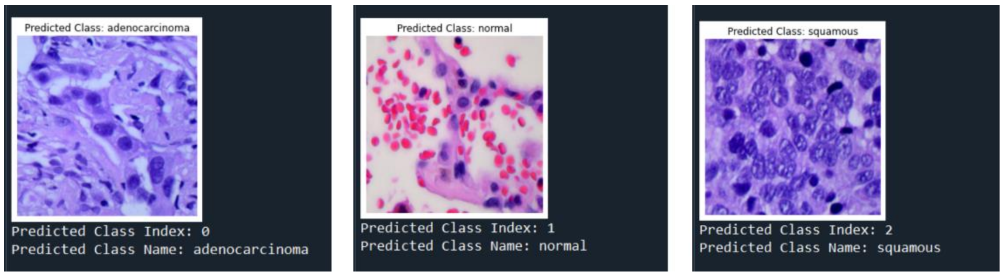
```
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.91      0.89      1037
           1       0.90      0.84      0.87       970
           2       0.90      0.91      0.90       993

    accuracy                           0.89      3000
   macro avg       0.89      0.89      0.89      3000
weighted avg       0.89      0.89      0.89      3000
```

Fig.4.8:Classification Report for VGG16 Model

4.7 SAMPLE OUTPUT

To demonstrate the effectiveness and performance of the trained models, we provide sample outputs from the Convolutional Neural Network (CNN) and VGG16-based models. These outputs include the predicted labels for the test images, as well as visual representations of the model's performance through metrics such as the confusion matrix and classification report. By examining these outputs, we can better understand the models' strengths and areas for improvement.

4.7.1 Predicted Labels: The models were tested on a set of histopathological images of lung cancer, and the predicted labels were compared to the true labels. Here are a few examples of the test images along with the labels predicted by the CNN and VGG16 models:

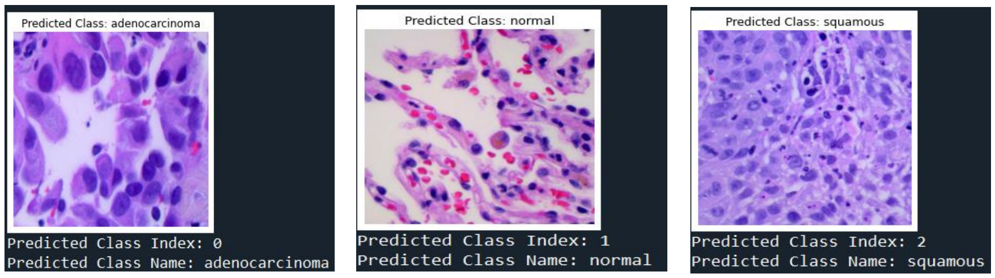


```
Predicted Class Index: 0
Predicted Class Name: adenocarcinoma
```

```
Predicted Class Index: 1
Predicted Class Name: normal
```

```
Predicted Class Index: 2
Predicted Class Name: squamous
```

Fig.4.9:Sample outputs for CNN Model



```
Predicted Class Index: 0
Predicted Class Name: adenocarcinoma
```

```
Predicted Class Index: 1
Predicted Class Name: normal
```

```
Predicted Class Index: 2
Predicted Class Name: squamous
```

Fig.4.10:Sample outputs for VGG16 Model

4.8 GUI FOR THE DETECTION MODEL

The Graphical User Interface (GUI) is a user-friendly visual interface designed to interact with the Lung Cancer Detection Model. It simplifies the process of loading, processing, and predicting the class of histopathological images of lung cancer. By using the GUI, users can easily upload an image and receive the model's prediction without dealing with the underlying code or complex command-line operations. The GUI is built using Tkinter, a standard Python library for creating graphical user interfaces, and it incorporates elements from the PIL (Pillow) library for image handling and the TensorFlow/Keras library for model prediction.

4.8.1 Key Features of the GUI

i. User-friendly Interface: Provides an intuitive and straightforward interface for users to interact with the model.

ii. Image Upload: Allows users to browse and select an image file from their local file system.

iii. Model Prediction: Displays the predicted class of the uploaded image.
iv. Visual Feedback: Shows the uploaded image and prediction results in the GUI window.

4.8.2 DETAILED FUNCTIONALITY

i. Loading the Trained Model: The GUI loads a pre-trained VGG16 model or the CNN model stored in the specified file path (lung_cancer_model_vgg16.h5 / lung_cancer_model.h5). The models have been trained to classify lung histopathological images into three categories: Adenocarcinoma, Normal, and Squamous.
ii. Image Upload and Display: Users can click the "Browse Image" button to open a file dialog and select an image file (JPEG or PNG format). The selected image is then processed and displayed within the GUI.
iii. Image Preprocessing: The uploaded image is resized to 128x128 pixels, which matches the input size expected by the model. The image is then converted to an array and normalized by scaling pixel values to the range [0, 1].
iv. Model Prediction: The preprocessed image is fed into the VGG16 model / CNN model, which outputs the predicted class probabilities. The class with the highest probability is selected as the predicted class.
v. Displaying Results: The GUI displays the predicted class label (Adenocarcinoma, Normal, or Squamous) below the image. The label is updated each time a new image is uploaded and processed.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

The project aimed to develop and evaluate a deep learning-based model for detecting lung cancer from histopathological images. By leveraging convolutional neural networks (CNNs) and transfer learning with a pre-trained VGG16 model, we successfully created a robust classifier capable of distinguishing between Adenocarcinoma, Normal, and Squamous cell types.

The models exhibited significant learning capabilities, as evidenced by the progressive decrease in training and validation loss over the training epochs.

The CNN model achieved a final training accuracy of approximately ~96%, with a corresponding validation accuracy of around ~93%.

The VGG16-based model, utilizing transfer learning, attained a final training accuracy of ~98%, with a validation accuracy of ~95%, indicating improved performance due to the pre-trained layers' learned features.

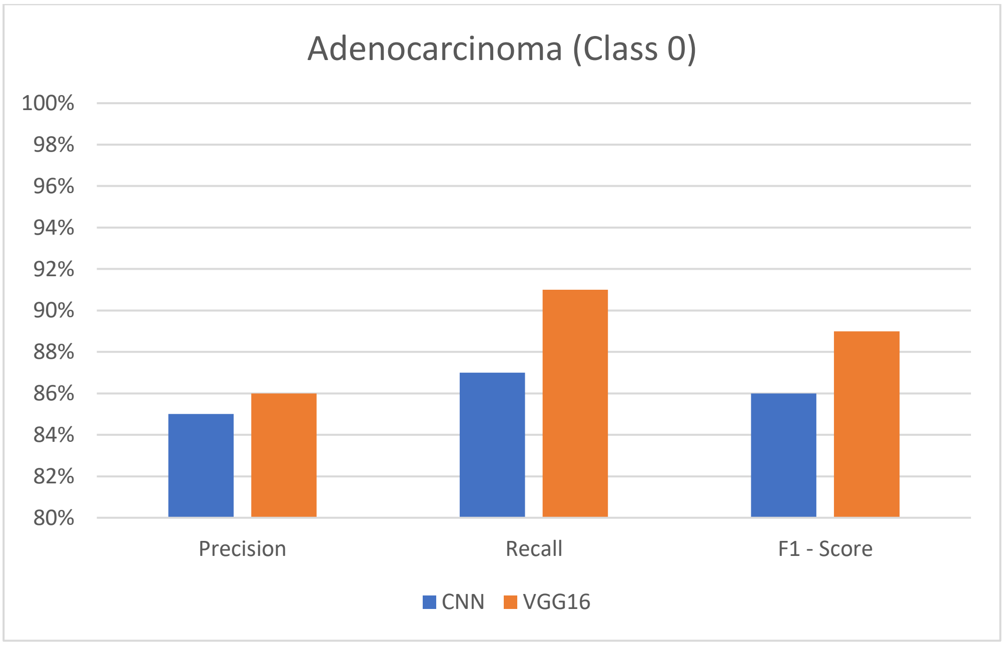


Fig.5.1: The performance of both model for Class 0 in Precision, Accuracy, and F1 Score

The CNN model achieves an overall accuracy of 87% on the test dataset, with precision, recall, and F1-scores of 85%, 87%, and 86% respectively for adenocarcinoma (Class 0). For normal cases (Class 1), the precision, recall, and F1-scores are 89%, 82%, and 85% respectively. In the case of squamous cell carcinoma (Class 2), the model achieves precision, recall, and F1-scores of 88%, 91%, and 89% respectively.

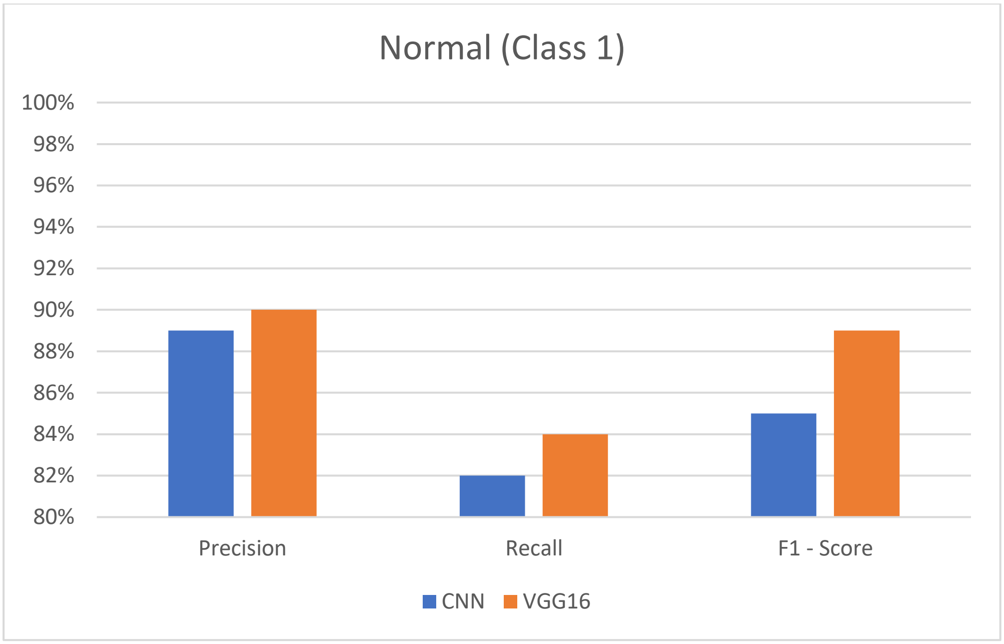


Fig.5.2: The performance of both model for Class 1 in Precision, Accuracy, and F1 Score

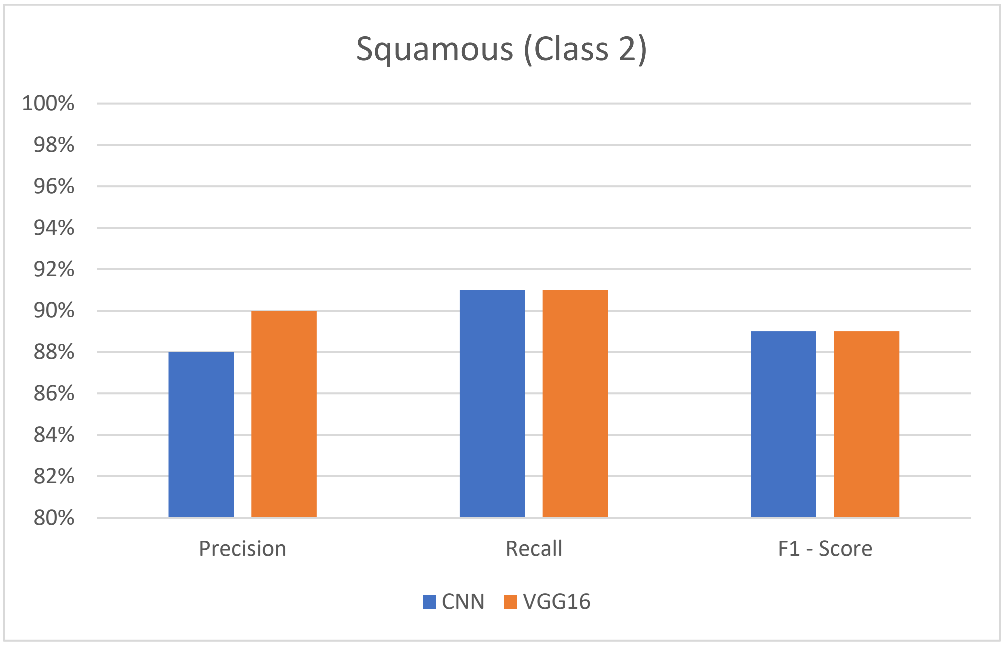


Fig.5.3: The performance of both model for Class 2 in Precision, Accuracy, and F1 Score

In contrast, the VGG16 model demonstrates higher performance metrics with an overall accuracy of 89%. For adenocarcinoma (Class 0), precision, recall, and F1-scores are 86%, 91%, and 89% respectively. For normal cases (Class 1), precision, recall, and F1-scores are 90%, 84%, and 87% respectively. Finally, for squamous cell carcinoma (Class 2), precision, recall, and F1-scores are 90%, 91%, and 90% respectively.

Comparing the models, VGG16 exhibits slightly superior performance across all classes, achieving higher precision, recall, and F1-scores. These results suggest that the VGG16 model is better able to distinguish between different types of lung cancer, resulting in overall improved classification accuracy.

In terms of accuracy, macro average, and weighted average, the CNN model achieves consistent scores of 87%. This indicates that the model is reliable across different evaluation metrics and provides a balanced performance across all classes. However, the VGG16 model demonstrates slightly higher accuracy metrics, with scores of 89% for accuracy, macro average, and weighted average. This suggests that the VGG16 model offers improved overall performance compared to the CNN model, showcasing enhanced accuracy in classifying histopathological images of lung cancer. These higher accuracy scores imply that the VGG16 model can more accurately distinguish between the different classes of lung cancer, leading to more precise predictions and potentially better clinical outcomes.

5.2 FUTURE SCOPE

The successful development and evaluation of deep learning models for lung cancer detection open up several avenues for future research and application:

5.2.1 Fine-tuning and Optimization: Further fine-tuning of hyperparameters, such as learning rate, batch size, and optimization algorithms, can potentially enhance the performance of the models. Optimization techniques like grid search or Bayesian optimization can be employed to systematically search for the optimal hyperparameters.

5.2.2 Ensemble Learning: Exploring ensemble learning techniques, such as bagging or boosting, by combining multiple models can further improve the classification performance. Ensemble methods have the potential to mitigate the weaknesses of individual models and yield more robust predictions.

5.2.3 Data Augmentation: Increasing the diversity and size of the training dataset through advanced data augmentation techniques can help improve the models' generalization capabilities. Augmentation methods like rotation, translation, and elastic deformation can synthetically generate additional training samples, leading to better model performance.

5.2.4 Interpretable Models: Investigating interpretable machine learning models, such as decision trees or rule-based models, can provide insights into the features and patterns contributing to the classification decisions. Interpretable models can help enhance trust and understanding of the model predictions in clinical settings.

5.2.5 Transfer Learning with Pretrained Models: Leveraging pretrained models on larger medical imaging datasets, such as the ChestX-ray14 or the NIH Chest X-ray dataset, can potentially improve the models' performance. Fine-tuning pretrained models on specific lung cancer datasets can capture more relevant features and accelerate model convergence.

5.2.6 Clinical Integration and Validation: Conducting rigorous clinical validation studies to assess the real-world performance and clinical utility of the developed models is essential. Collaborating with healthcare professionals and medical institutions to deploy and evaluate the models in clinical settings can facilitate their adoption for assisting radiologists in lung cancer diagnosis.

5.2.7 Multi-Modal Data Fusion: Integrating complementary data sources, such as clinical data, genetic information, or other imaging modalities like CT scans or MRI, can provide a more comprehensive understanding of lung cancer. Multi-modal data fusion approaches can potentially enhance the accuracy and reliability of the diagnostic models.

5.2.8 Deployment in Healthcare Systems: Streamlining the integration of the developed models into existing healthcare systems and electronic medical records (EMRs) is crucial for their practical deployment. Ensuring compliance with regulatory standards, such as HIPAA (Health Insurance Portability and Accountability Act) and GDPR (General Data Protection Regulation), is essential for safeguarding patient privacy and data security.

Overall, continued research and development efforts in these areas can contribute to the advancement of AI-driven diagnostic tools for lung cancer detection, ultimately improving patient outcomes and advancing personalized medicine.

APPENDIX

CNN MODEL DEVELOPMENT SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from glob import glob

import cv2
import gc
import os

from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,
confusion_matrix
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint

import warnings
warnings.filterwarnings('ignore')



main_folder =
r'C:\Users\iamfi\OneDrive\Desktop\Lung_Cancer_Detection_project\lu
ng_colon_image_set\lung_image_sets'

# Creating a DataFrame with file names and labels
labels = []
file_names = []

for label in os.listdir(main_folder):
    label_folder = os.path.join(main_folder, label)
    if os.path.isdir(label_folder):
        for filename in os.listdir(label_folder):
            file_names.append(os.path.join(label, filename))
            labels.append(label)


data = pd.DataFrame({"FILE_NAME": file_names, "CATEGORY": labels})


# Spliting the dataset into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2,
random_state=42)


# ImageDataGenerator for data augmentation
datagen = ImageDataGenerator(rescale=1./255,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True)

# Creating a data generator for training data
train_generator =
datagen.flow_from_dataframe(dataframe=train_data,

directory=main_folder,
                                              x_col="FILE_NAME",
                                              y_col="CATEGORY",

class_mode="categorical",
                                              target_size=(128,
128),
                        batch_size=32)

# Displaying three sample images from each class
fig, axs = plt.subplots(len(train_generator.class_indices), 3,
figsize=(15, 5 * len(train_generator.class_indices)))

for i, class_label in
enumerate(train_generator.class_indices.keys()):
    class_images = train_data[train_data['CATEGORY'] ==
class_label]['FILE_NAME'].tolist()[:3]

    for j, img_path in enumerate(class_images):
        img = Image.open(os.path.join(main_folder, img_path))
        axs[i, j].imshow(img)
        axs[i, j].set_title(f"Class: {class_label}")
        axs[i, j].axis('off')

plt.show()

# Creating a data generator for testing data
test_generator = datagen.flow_from_dataframe(dataframe=test_data,

directory=main_folder,
                                             x_col="FILE_NAME",
                                             y_col="CATEGORY",

class_mode="categorical",
                                             target_size=(128,
128),
                                             batch_size=32)

# Building the CNN model
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(128, 128, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(3, activation='softmax'))  # 3 output
classes
model.summary()

keras.utils.plot_model(
     model,
     show_shapes = True,
     show_dtype = True,
     show_layer_activations = True
)
# Compiling the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Defining callbacks
checkpoint = ModelCheckpoint("lung_cancer_model.h5",
monitor='val_accuracy', save_best_only=True, mode='max',
verbose=1)

# Training the model
history = model.fit(train_generator,
                    epochs=10,
                    validation_data=test_generator,
                    callbacks=[checkpoint])

#Plotting Graphs
history_df = pd.DataFrame(history.history)
history_df.loc[:,['loss','val_loss']].plot()
history_df.loc[:,['accuracy','val_accuracy']].plot()
plt.show()


# Evaluating the model
test_loss, test_acc = model.evaluate(test_generator)
print(f"Test Accuracy: {test_acc}")

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

# Printing classification report and confusion matrix
print("Classification Report:\n", classification_report(y_true,
y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
print('Model Saved!')
```

VGG16 MODEL DEVELOPMENT SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from glob import glob

import cv2
import gc
import os

from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,
confusion_matrix
import tensorflow as tf
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout

import warnings
warnings.filterwarnings('ignore')

main_folder =
r'C:\Users\iamfi\OneDrive\Desktop\Lung_Cancer_Detection_project\lu
ng_colon_image_set\lung_image_sets'

# Creating a DataFrame with file names and labels
labels = []
file_names = []
```

```
for label in os.listdir(main_folder):
    label_folder = os.path.join(main_folder, label)
    if os.path.isdir(label_folder):
        for filename in os.listdir(label_folder):
            file_names.append(os.path.join(label, filename))
            labels.append(label)

data = pd.DataFrame({"FILE_NAME": file_names, "CATEGORY": labels})

# Splitting the dataset into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2,
random_state=42)

# ImageDataGenerator for data augmentation
datagen = ImageDataGenerator(rescale=1./255,
                             shear_range=0.2,
                             zoom_range=0.2,
                             horizontal_flip=True)

# Creating a data generator for training data
train_generator =
datagen.flow_from_dataframe(dataframe=train_data,

directory=main_folder,
                                                 x_col="FILE_NAME",
                                                 y_col="CATEGORY",

class_mode="categorical",
                                                 target_size=(128,
128),
                                                 batch_size=32)

# Displaying three sample images from each class
fig, axs = plt.subplots(len(train_generator.class_indices), 3,
figsize=(15, 5 * len(train_generator.class_indices)))

for i, class_label in
```

```
enumerate(train_generator.class_indices.keys()):
    class_images = train_data[train_data['CATEGORY'] ==
class_label]['FILE_NAME'].tolist()[:3]

    for j, img_path in enumerate(class_images):
        img = Image.open(os.path.join(main_folder, img_path))
        axs[i, j].imshow(img)
        axs[i, j].set_title(f"Class: {class_label}")
        axs[i, j].axis('off')
plt.show()

# Creating a data generator for testing data
test_generator = datagen.flow_from_dataframe(dataframe=test_data,

directory=main_folder,
                                             x_col="FILE_NAME",
                                             y_col="CATEGORY",

class_mode="categorical",
                                             target_size=(128,
128),
                                             batch_size=32)

# Loading the VGG16 model pretrained on ImageNet without including
the top (fully connected) layers
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(128, 128, 3))

# Freezing the layers in the base model
for layer in base_model.layers:
    layer.trainable = False

# Adding custom top layers
x = Flatten()(base_model.output)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(3, activation='softmax')(x)
```

```
# Combine the base model with custom top layers
model = Model(inputs=base_model.input, outputs=output)

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

# Print model summary
model.summary()

# Defining callbacks
checkpoint = ModelCheckpoint("lung_cancer_model_vgg16.h5",
monitor='val_accuracy', save_best_only=True, mode='max',
verbose=1)

# Training the model
history = model.fit(train_generator,
                    epochs=10,
                    validation_data=test_generator,
                    callbacks=[checkpoint])

# Plotting Graphs
history_df = pd.DataFrame(history.history)
history_df.loc[:,['loss','val_loss']].plot()
history_df.loc[:,['accuracy','val_accuracy']].plot()
plt.show()

# Evaluating the model
test_loss, test_acc = model.evaluate(test_generator)
print(f"Test Accuracy: {test_acc}")

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

# Printing classification report and confusion matrix
```

```
print("Classification Report:\n", classification_report(y_true,
y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
print('Model Saved!')
```

GRAPHICAL USER INTERFACE

```
import os
import numpy as np
from tensorflow import keras
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
from PIL import Image, ImageTk, ImageDraw, ImageFont  # Import
ImageFont module
import tkinter as tk
from tkinter import filedialog

def predict_single_image(model, img_path, label, panel):
    # Loading and preprocessing the image
    img = image.load_img(img_path, target_size=(128, 128))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0  # Normalizing the image

    # Making predictions
    predictions = model.predict(img_array)
    predicted_class_index = np.argmax(predictions)

    # Decoding the predicted class index to the original label
    class_labels = {0: 'Adenocarcinoma', 1: 'Normal', 2:
'Squamous'}
    predicted_class = class_labels[predicted_class_index]

    # Display the image
    img = Image.open(img_path)
    window_width = root.winfo_width()
    window_height = root.winfo_height()
    img_width = int(0.5 * window_width)
    img_height = int(0.5 * window_height)
    img = img.resize((img_width, img_height), Image.ANTIALIAS)
    img_tk = ImageTk.PhotoImage(img)
    panel.configure(image=img_tk)
    panel.image = img_tk  # Keep a reference to the PhotoImage
object

    # Update the predicted class label
    label.config(text=f"Predicted Class: {predicted_class}",
font=("Helvetica", 18, "bold"), fg="black")
    label.place(relx=0.5, rely=0.8, anchor=tk.CENTER)

def browse_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image
files", "*.jpg;*.jpeg;*.png")])
    if file_path:
        predict_single_image(model, file_path, predicted_label,
image_panel)

if __name__ == "__main__":
    # Path to the trained model file
    model_path =
r"C:\Users\iamfi\OneDrive\Desktop\Lung_Cancer_Detection_project\VG
G16\lung_cancer_model_vgg16.h5"

    # Loading the trained model
    model = keras.models.load_model(model_path)

    # Creating a Tkinter window
    root = tk.Tk()
    root.title("Lung Cancer Prediction")

    # Background image
    bg_image = Image.open("image.jpeg")
    window_width = root.winfo_screenwidth()
    window_height = root.winfo_screenheight()
    bg_image = bg_image.resize((window_width, window_height),
Image.ANTIALIAS)
    draw = ImageDraw.Draw(bg_image)
    font = ImageFont.truetype("freedom.ttf", 52)  # Load font
    draw.text((window_width/2, window_height/7), "LUNG CANCER
DETECTION MODEL", fill=(255, 255, 255, 128), anchor="ma",
font=font)

    # Convert the background image to PhotoImage
    bg_photo = ImageTk.PhotoImage(bg_image)
    bg_label = tk.Label(root, image=bg_photo)
    bg_label.image = bg_photo
    bg_label.place(x=0, y=0, relwidth=1, relheight=1)

    # Image panel
    image_panel = tk.Label(root, bg="#f0f0f0")
    image_panel.pack(expand=True)

    # Button to browse image
    browse_button = tk.Button(root, text="Browse Image",
command=browse_image, font=("Helvetica", 20), bg="#172547",
fg="#ffffff")
    browse_button.pack(pady=(40, 20))

    # Predicted class label
    predicted_label = tk.Label(root, text="", font=("Helvetica",
18, "bold"), bg="#f0f0f0")
    predicted_label.pack(pady=(2,2))

    root.geometry("1000x600")
    root.mainloop()
```

SNAPSHOTS OF THE GUI

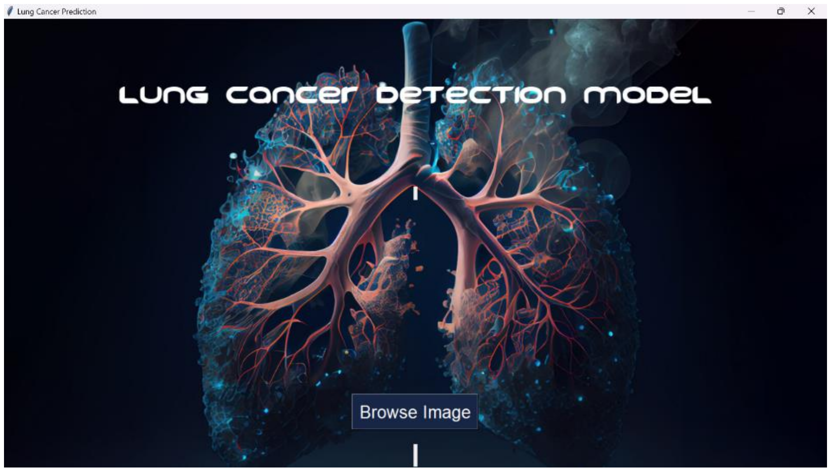


Fig.A.1:GUI Front Screen

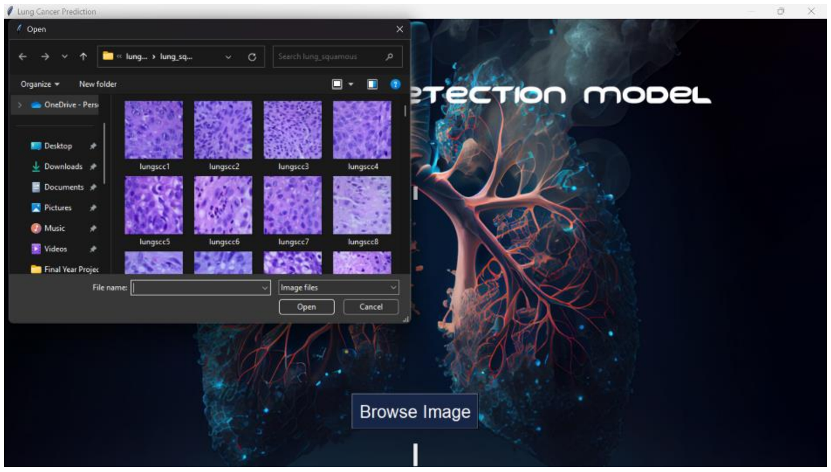
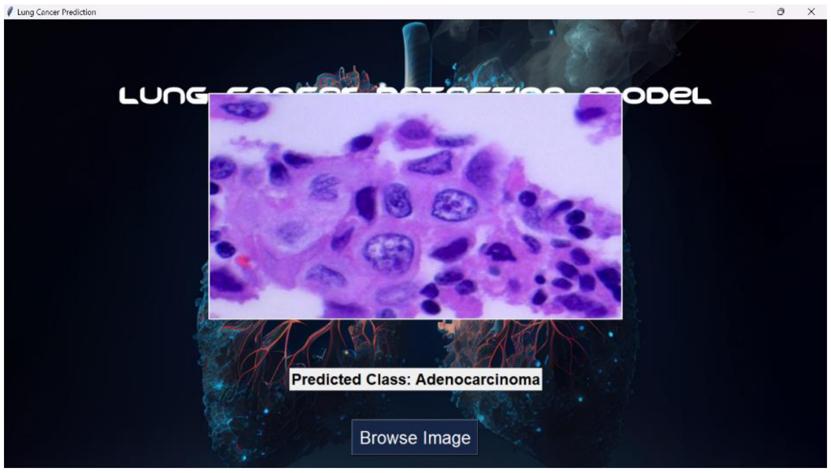


Fig.A.2:GUI Image Browsing



Predicted Class: Adenocarcinoma

Fig.A.3:GUI Predicted Label with Image

REFERENCES

1. World Health Organization. (2021). Global cancer statistics. Retrieved from WHO website.
2. American Cancer Society. (2021). Early detection of lung cancer. Retrieved from American Cancer Society website.
3. National Cancer Institute. (2021). Lung cancer screening. Retrieved from National Cancer Institute website.
4. Chan, J.K.C., & Ho, J.T. (2020). Diagnostic challenges in lung cancer. Journal of Thoracic Oncology, 15(4), 637-644.
5. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
6. Rajpurkar, P., et al. (2017). CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning. arXiv preprint arXiv:1711.05225.
7. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & van Ginneken, B. (2017). A survey on deep learning in medical image analysis. Medical image analysis, 42, 60-88.
8. Shen, W., Zhou, M., Yang, F., Yang, C., & Tian, J. (2017). Multi-scale convolutional neural networks for lung nodule classification. In International Conference on Information Processing in Medical Imaging (pp. 588-599). Springer, Cham.
9. Hua, K., Zhang, J., & Zhao, Z. (2015). Deep learning based framework for lung nodules detection and classification from CT images. In International Conference on Cloud Computing and Big Data (pp. 355-360). IEEE.
10. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
11. Kumar, V., Sabut, S., & Mukhopadhyay, S. (2018). Breast cancer classification using transfer learning approach. In IEEE-EMBS Conference on Biomedical Engineering and Sciences (pp. 65-70). IEEE.
12. Coudray, N., Ocampo, P. S., Sakellaropoulos, T., Narula, N., Snuderl, M.,

Fenyo, D., ... & Tsirigos, A. (2018). Classification and mutation prediction from non-small cell lung cancer histopathology images using deep learning. Nature Medicine, 24(10), 1559-1567.

13. Murphy, K., van Ginneken, B., Schilham, A. M. R., de Hoop, B., Gietema, H., & Prokop, M. (2009). A large-scale evaluation of automatic pulmonary nodule detection in chest CT using local image features and k-nearest-neighbor classification. Medical image analysis, 13(5), 757-770.
14. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.
15. Borkowski AA, Bui MM, Thomas LB, Wilson CP, DeLand LA, Mastorides SM. Lung and Colon Cancer Histopathological Image Dataset (LC25000). arXiv:1912.12142v1 [eess.IV], 2019
16. Krishnaiah, V., G. Narsimha, and Dr N. Subhash Chandra. "Diagnosis of lung cancer prediction system using data mining classification techniques." International Journal of Computer Science and Information Technologies 4.1 (2013): 39-45
17. Daoud, Maisa, and Michael Mayo. "A survey of neural network-based cancer prediction models from microarray data." Artificial intelligence in medicine (2019).
18. Masud, Mehedi, Niloy Sikder, Abdullah-Al Nahid, Anupam Kumar Bairagi, and Mohammed A. AlZain. "A machine learning approach to diagnosing lung and colon cancer using a deep learning-based classification framework." Sensors 21, no. 3 (2021): 748.
19. Hage Chehade, A., Abdallah, N., Marion, J.M., Oueidat, M. and Chauvet, P., 2022. Lung and colon cancer classification using medical imaging: A feature engineering approach. Physical and Engineering Sciences in Medicine, 45(3), pp.729-746.
20. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press.
21. Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.