LUNG AND COLON CANCERS DETECTION

Dissertation submitted

In partial fulfilment of requirements for the award of the degree of

Master of Computer Applications (M.C.A)

Submitted By

KOLIPAKA HEMA SUNDAR

(Regd. No: 322203320030)

Under the esteemed guidance of

Mr. B. BALA KRISHNA

Assistant Professor

Department of Computer Applications

COLLEGE OF ENGINEERING
(AUTONOMUS)

DEPARTMENT OF COMPUTER APPLICATIONS

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)

(Affiliated to Andhra University, A.P)

2022-2024

LUNG AND COLON CANCERS DETECTION

Dissertation submitted

In partial fulfilment of requirements for the award of the degree of

Master of Computer Applications (M.C.A)

Submitted By

KOLIPAKA HEMA SUNDAR

(Regd. No: 322203320030)

Under the esteemed guidance of

Mr. B. BALA KRISHNA

Assistant Professor

Department of Computer Applications

COLLEGE OF ENGINEERING
(AUTONOMUS)

DEPARTMENT OF COMPUTER APPLICATIONS

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)

(Affiliated to Andhra University, A.P)

2022-2024

CERTIFICATE

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING

(AUTONOMOUS)

This is to certify that, the dissertation titled **"LUNG AND COLON CANCERS DETECTION"** is submitted by **Mr. KOLIPAKA HEMA SUNDAR** with Reg. No **322203320030,** in partial fulfillment of the requirement for award of the Degree of **M.C.A** in **GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMUS)** affiliated to Andhra University, Visakhapatnam is a bona fide record of project carried out by him under my guidance and supervision.

The content of the project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Signature of the Internal Guide **Signature of the Head of the Department**

Signature of the External Examiner

CERTIFICATE OF PLAGIARISM CHECK

COLLEGE OF ENGINEERING
(AUTONOMUS)

This is to certify the Master of Computer Applications (MCA) dissertation submitted by KOLIPAKA HEMA SUNDAR (322203320030) of the Department of Computer Applications under the supervision of Mr. B. BALA KRISHNA, Assistant Professor, has undergone a plagiarism check and found to have similarity index less than ……. The details of the plagiarism check are as under:

File Name :

Dissertation Title :

Date and Time of Submission :

Submission ID :

Similarity Index :

Dean Academic Programs (PG)

DECLARATION

I hereby declare that dissertation titled **"LUNG AND COLON CANCERS DETECTION"** is submitted to the Department of Computer Applications, **Gayatri Vidya Parishad College of Engineering (Autonomous)** affiliated to Andhra University, Visakhapatnam in partial fulfilment of the requirements for the award of the Degree of **Master of Computer Applications (M.C.A).** This work is done by me and authentic to the best of my knowledge under the direction and valuable guidance of **Mr. B. BALA KRISHNA** Assistant Professor, Department of Computer Applications.

KOLIPAKA HEMA SUNDAR

(Regd. No. 322203320030)

ACKNOWLEDGEMENT

I take the opportunity to thank everyone who has contributed to making the project possible. I am thankful to **Gayatri Vidya Parishad College of Engineering (Autonomous)** for giving opportunity me to work on a project as part of the curriculum.

I offer my copious thanks to **Dr. A. BalaKoteswara Rao**, the Principal, **Gayatri Vidya Parishad College of Engineering (Autonomous),** for providing the best faculty and lab facilities throughout the completion of Master of Computer Applications course.

I offer my copious thanks to Prof. **Dr. K. Narasimha Rao** Professor & Dean, Academics (PG), for his valuable suggestions and constant motivation that greatly helped me to complete project successfully.

I offer my copious thanks to **Dr. Y. Anuradha,** Associate Professor & Head, Department of Computer Applications, for her valuable suggestions and constant motivation that greatly helped me to complete the project successfully.

My sincere thanks to **Mr. B. Bala Krishna**, Assistant Professor, Department of Computer Applications, my project guide, for her constant support, encouragement, and guidance. I am very much grateful for her valuable suggestions.

KOLIPAKA HEMA SUNDAR

(Regd. No. 322203320030)

ABSTRACT

Lung and colon cancers are among the most common types of cancer worldwide, and accurate detection and classification of these cancers is crucial for effective treatment and it improves patient outcomes. In this project Median Filtering for Preprocessing and **Convolutional Neural Network** models like **Sequential model and ResNet models** for the detection of cancer through **histopathological images**. Moreover, there are **machine learning models** like **SVM, Decision tree** which can classify the cancers, but those are not much accurate. This project demonstrates that the system outperforms existing methods in terms of accuracy. The proposed algorithm has the potential to significantly enhance the efficiency and accuracy of cancer detection and reduces the **diagnosing time**, ultimately contributing to improved treatment quality and patient outcomes**.**

Keywords: Convolutional Neural Network,Median Filtering,Sequential model, Histopathological Images.

INDEX

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

INTRODUCTION

Lung and colon cancers are among the most common types of cancer worldwide, with cancer rates of 11.4% and 10% respectively in 2020. Although these cancers are widespread, there is a possibility of synchronous occurrence of lung and colon cancers (LCC), resulting in high mortality rates of 18% and 9.4% respectively for each cancer. Therefore, precise detection of these cancer sub-types is necessary to improve therapeutic strategies in the early stages of cancer.

Non-invasive diagnostic methods for lung cancer include computed tomography (CT) images and radiography, while CT colonoscopy and flexible sigmoidoscopy are common for colon cancer. However, these non-invasive techniques may not always reliably differentiate specific sub-categories of these cancers, necessitating slightly invasive methods like histopathology for accurate detection and better treatment quality. Manual grading of histopathological images (HI) can be challenging, time-consuming, and prone to error. Thus, automatic image processing approaches for LCC images have proven beneficial for easing the burden on pathologists.

In recent years, advancements in biomedical applications have expanded opportunities for the diagnosis and treatment of various types of cancer. Utilizing artificial intelligence (AI) technology, including deep learning (DL) techniques, offers the potential to accelerate decision-making and enhance diagnostic accuracy. This project aims to analyze data containing the HI of LCCs, employing a Convolutional Neural Network (CNN) approach for precise cancer detection and classification. Additionally, diagnosing cancer can be a lengthy process that relies on the judgments of multiple physicians, especially in the early stages.

AI applications have made significant contributions to the medical field, such as early diagnosis of biomedical images, disease prediction, and medical emergencies. DL techniques can extract latent features in medical images, providing timely cancer identification and differentiating between various stages.

- Development of a new LCCD algorithm consisting of median filtering (MF)-based preprocessing and CNN classification, designed specifically for LCC detection. To our knowledge, this LCCD approach is novel in the literature.

- Implementation of MobileNet for extracting relevant features from HIs, critical for precise cancer detection. Additionally, the CNN Sequential model excels at identifying occurrences of LCC in HIs, improving detection accuracy and contributing to better treatment outcomes.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

[1] Mizuho et al. employed machine learning algorithms for diagnosing three types of lung cancer. They extracted features using homology-based image and texture analysis methods, finding that machine learning with homology-based image features performed better than texture analysis. This demonstrates the value of different feature extraction techniques in improving diagnosis.

[2] Vinod et al. designed a method to identify pulmonary nodules using the watershed algorithm and Gabor filter. Features extracted from these methods were then categorized using SVM. This approach underscores the potential of combining traditional image processing techniques with machine learning for accurate diagnosis.

[3] Mesut et al. developed the DarkNet-19 model for training lung and colon cancer datasets from scratch. They applied the Equilibrium algorithm to select efficient features from the data, which were then classified using Support Vector Machines (SVM). This approach emphasizes the importance of feature selection in achieving better classification results.

[4] Mehedi et al. proposed a deep learning model for diagnosing five classes of lung and colon cancer, incorporating 2D Fourier and 2D wavelet features. This innovative approach achieved an accuracy of 96.33%, demonstrating the effectiveness of these features in cancer diagnosis.

[5] Sanidhya et al. presented a CNN Pre-Trained Diagnostic Network for diagnosing lung and colon cancer. Their approach uses a shallow CNN architecture trained on histological images, yielding impressive accuracy rates of 96% and 97% for diagnosing colon and lung cancer, respectively. The shallow CNN architecture leverages pre-trained models to improve accuracy in diagnosing these types of cancers.

[6] Mumtaz et al. introduced a capsule network with multiple inputs for diagnosing abnormal cell carcinoma of the lung and colon. This approach combines convolutional layers with the capsule

network to improve detection capabilities. The use of capsule networks, which retain spatial relationships between features, adds value in the accurate diagnosis of cancers.

[7] SHAHID et al. optimized AlexNet, a deep learning model, for diagnosing lung and colon cancer cells. By modifying its four essential layers and training it on the dataset, they achieved an accuracy of 89%. This demonstrates the flexibility of AlexNet and its potential in cancer diagnosis.

[8] Dipanjan et al. generated a 1D CNN network to classify small cell lung tumors. By combining hybrid features with clinical features, their approach exceeded traditional machine learning techniques, highlighting the potential for hybrid models in lung cancer diagnosis.

Overall, these studies showcase the breadth and depth of approaches being explored in the diagnosis of lung and colon cancers. By combining different deep learning architectures, feature extraction methods, and machine learning algorithms, researchers are achieving high accuracy rates and advancing the field of medical diagnostics.

CHAPTER 3

SYSTEM ANALYSIS

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system likely includes **traditional methods of cancer detection**, such as **manual grading of histopathological images by pathologists**, as well as **non-invasive techniques** like computed **tomography (CT) and radiography for lung cancer, and CT colonoscopy** for colon cancer. However, the limitations of these existing approaches, including the **time-consuming nature** of manual grading, the potential for error, and the inability of non-invasive techniques to accurately identify sub-categories of these cancers. Moreover, there are **machine learning models** which can classify the cancers, but those are not much accurate. So, there is a need for enhanced algorithms and techniques to address the challenges associated with the existing system, particularly in the context of accurately detecting and classifying lung and colon cancers using of complex histopathological images.

DISADVANTAGES:

- The early detection and classification of lung and colon cancers using histopathological images (HI) and deep learning techniques hold great promise; however, there are several challenges and disadvantages to consider:
- Manual Grading of HI: The manual grading of histopathological images can be a time-consuming and labor-intensive process for pathologists. This method is also subject to human error, leading to inconsistencies in diagnosis.
- Non-Invasive Techniques: Although non-invasive diagnostic techniques, such as computed tomography (CT) and radiography, can provide valuable information, they may not always accurately identify specific sub-categories of lung and colon cancers.
- Limited Machine Learning Accuracy: While machine learning models have been employed to classify cancers, their performance may vary and may not always achieve the level of accuracy required for reliable diagnosis. These models may struggle with complex histopathological images.
- Lack of Precision in Sub-Category Detection: Traditional methods and existing machine learning models may lack the precision needed to accurately classify the different sub-categories of lung and colon cancers, which is crucial for targeted treatment plans.

- Dependency on High-Quality Data: The effectiveness of machine learning models depends heavily on the quality of available data. Inconsistencies in data quality or lack of sufficient annotated datasets can hinder model performance.

3.2 PROPOSED SYSTEM

In the proposed system, the Convolutional Neural Network for Lung and Colon Cancer Detection method, aims to enhance the accuracy and efficiency of cancer detection and classification in histopathological images. This new approach incorporates advanced components such as the median filtering (MF) approach for noise removal, a deep convolutional neural network (DCNN) based models like Sequential and ResNet model for cancer detection, the proposed system seeks the grading of histopathological images and accurately identify sub-categories of lung and colon cancers. This system represents a significant advancement in the field of medical imaging and demonstrates superior performance compared to existing approaches.

ADVANTAGES:

- Enhanced Noise Removal: The median filtering (MF) approach effectively removes noise from histopathological images, leading to clearer and more accurate image analysis.
- Sequential CNN Model: Employing a CNN Sequential model for cancer detection allows the system to accurately classify and grade histopathological images. This approach provides reliable identification of sub-categories of lung and colon cancers.
- ResNet CNN Model: Employing a CNN ResNet model for cancer detection allows the system to accurately classify and grade histopathological images. This approach provides reliable identification of sub-categories of lung and colon cancers
- Improved Accuracy: The combination of advanced feature extraction and noise reduction techniques enhances the system's overall accuracy in cancer detection, outperforming existing methods.
- Efficiency in Diagnosis: By automating the grading and classification of histopathological images, the proposed system significantly reduces the time and effort required by pathologists, leading to faster diagnosis and improved patient outcomes.
- Superior Performance: This system demonstrates superior performance compared to traditional approaches, as it leverages advanced deep learning components to analyse complex images and detect cancer more effectively.
- Potential for Early Detection: The proposed system's ability to accurately grade and classify histopathological images can facilitate earlier detection of lung and colon cancers, enabling timely intervention and treatment.

CHAPTER 4

SOFTWARE REQUIREMENTS AND SPECIFICATIONS

SOFTWARE REQUIREMENTS AND SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS:

- Processor : Intel Core i5 or higher or AMD Ryzen 5 or higher.
- RAM : minimum of 8 GB

4.2 SOFTWARE REQUIREMENTS:

- Operating System : Windows 7/8/10/11, MacOS
- IDE : VS Code, Google Colab
- Front-End : HTML, CSS, JAVASCRIPT
- Framework : Flask

4.3 LIBRARIES USED:

- **OpenCV (Open-Source Computer Vision Library):**

  OpenCV is a popular open-source library for computer vision and image processing tasks. It provides a wide range of functions and algorithms for tasks such as image manipulation, feature detection, object tracking, and facial recognition. With its extensive documentation and cross-platform support, OpenCV is widely used in various applications, including robotics, augmented reality, and medical imaging.

- **TensorFlow (2.15.0):**

  TensorFlow is a popular open-source deep learning framework developed by Google. It allows developers to create, train, and deploy machine learning models for a variety of tasks, such as image and speech recognition, natural language processing, and more. TensorFlow 2.15.0 includes enhancements for ease of use and performance improvements.

- **scikit-learn (1.2.2):**

  Scikit-learn is a popular open-source machine learning library that provides tools for data preprocessing, model training, and evaluation. It supports a variety of algorithms such as classification, regression, clustering, and dimensionality reduction. Version 1.2.2 includes improvements and bug fixes.

- **Pandas (2.0.3):**

  Pandas is a powerful open-source data manipulation and analysis library for Python. It provides data structures such as DataFrames and Series for handling tabular data efficiently. Version 2.0.3 includes new features and improvements in performance and memory usage.

- **Gradio (4.26.0):**

  Gradio is an open-source library for creating interactive web interfaces for machine learning models. It allows developers to quickly build user-friendly applications for model inference and testing. Version 4.26.0 includes enhancements and new features.

- **Keras (2.15.0):**

  Keras is a high-level neural network library that provides an easy-to-use interface for building, training, and deploying deep learning models. Keras can be used as a frontend to TensorFlow. Version 2.15.0 includes compatibility with the latest TensorFlow release and performance improvements.

- **Matplotlib (3.7.1):**

  Matplotlib is a widely used open-source plotting library for Python. It allows users to create a variety of static, animated, and interactive plots and visualizations. Version 3.7.1 offers various enhancements and new features for plotting data.

- **Flask (2.2.5):**

  Flask is a lightweight web framework for Python that allows developers to create web applications quickly and efficiently. Version 2.2.5 includes performance improvements and bug fixes.

- **NumPy (1.25.2):**

  NumPy is an open-source numerical computing library for Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these data structures. Version 1.25.2 includes performance optimizations and new features.

CHAPTER 5

SYSTEM DESIGN

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

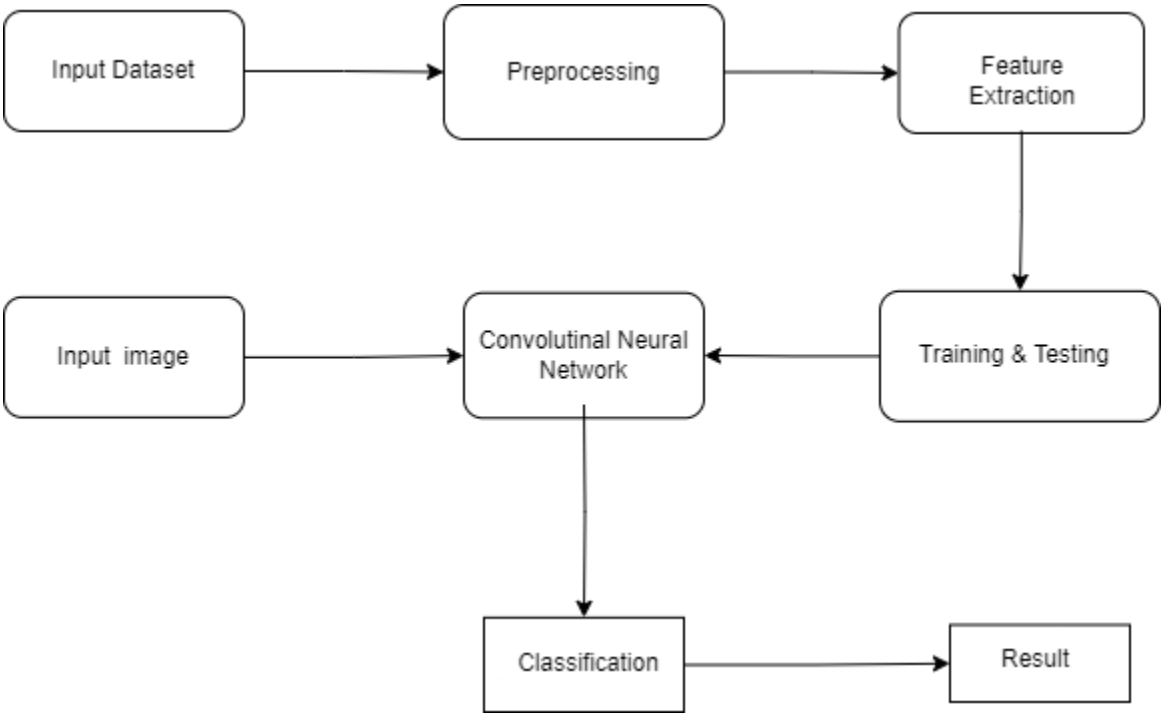

Fig 5.1: System Architecture

5.2 SYSTEM FLOWCHART:

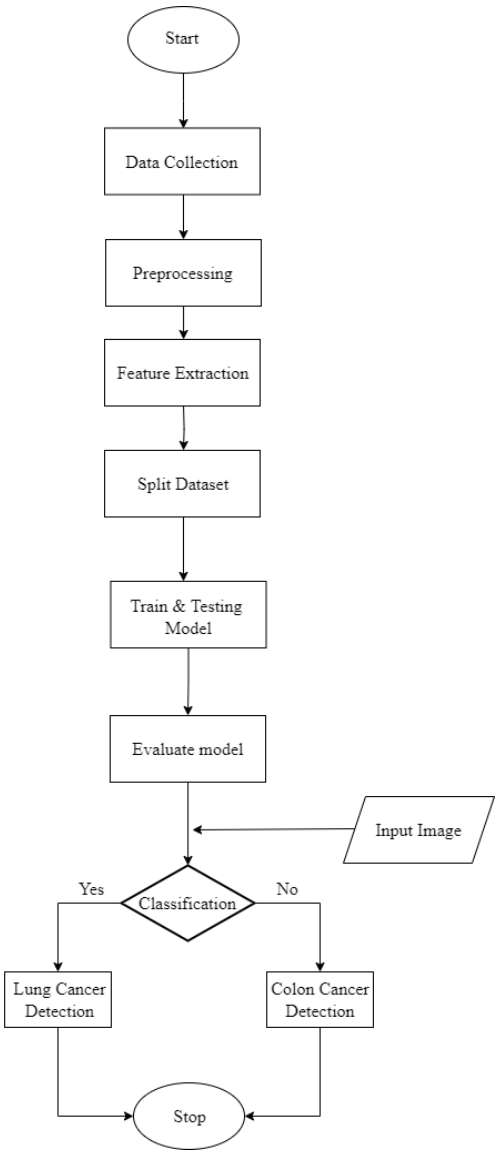

Fig 5.2: System Flow Chart

5.3 SYSTEM UML SEQUENCE DIAGRAM:

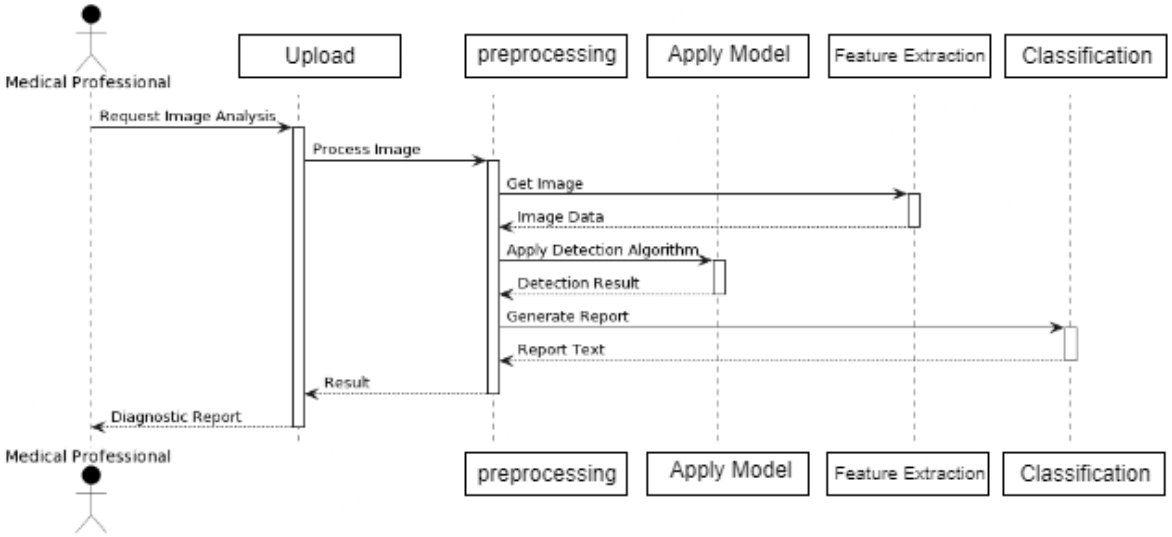

Fig 5.3: UML Sequence Diagram

CHAPTER 6

IMPLEMENTATION

IMPLEMENTATION

6.1 SEQUENTIAL ALGORITHM:

The Sequential model is a straightforward way to build and train neural networks for image classification using Keras, a high-level neural network API in TensorFlow. The model takes extracted features from images as input and predicts probabilities for each class as output. It involves defining the architecture by creating a Sequential model, then adding layers such as a dense layer with 512 units and ReLU activation, a dropout layer with a rate of 0.5 to prevent overfitting, and an output dense layer with softmax activation for classification. Once the model's architecture is set, it is trained using the input features and labels, specifying the number of epochs and batch size. After training, the model is evaluated on a test dataset to assess its performance and obtain metrics like accuracy. Once the model is trained and evaluated, it can be deployed for inference on new unseen data, providing predicted probabilities for each class based on the input features.

6.1.1 STEP BY STEP PROCESS OF SEQUENTIAL:

Input:

- Batch of images from the Dataset.

Output:

- Predicted probabilities of each class.

Steps:

1. Define the architecture of the CNN model:

- Create a Sequential model.

- Add layers sequentially:

- Add a Dense layer with 512 units and ReLU activation.

- Add a Dropout layer with a dropout rate of 0.5 to prevent overfitting.

- Add a Dense layer with the number of output classes and softmax activation for classification.

2. Train the model:

- Train the model using the extracted features as input and labels as output.

- Set the number of epochs and batch size for training.

3. Evaluate the model:

- Evaluate the trained model on the test dataset to assess its performance.

- Obtain metrics such as accuracy to measure the model's effectiveness.

4. Use the model for inference:

- Once trained and evaluated, deploy the model for making predictions on new unseen data.

- Provide input features to the model and obtain predicted probabilities for each class.

6.2 RESNET50 ALGORITHM:

ResNet50, or Residual Network with 50 layers, is a deep convolutional neural network (CNN) architecture designed to efficiently train very deep neural networks by introducing residual blocks with skip connections. These connections allow the network to bypass some layers, alleviating the vanishing gradient problem and improving training stability. ResNet50 consists of convolutional, pooling, and fully connected layers and begins with simple feature detection in the earlier layers before progressing to complex pattern recognition in deeper layers. The model can be used as a powerful feature extractor, and its pre-trained weights on large datasets like ImageNet provide a strong starting point for various tasks. ResNet50 is widely used in computer vision applications, such as image classification, object detection, and segmentation, due to its ability to balance depth with computational efficiency and state-of-the-art performance.

6.2.1 STEP BY STEP PROCESS OF RESNET50:

Input:

- Training data (train_gen)

- Image shape: (224, 224, 3)

- Number of epochs: 6

- Batch size: 36

Output:

- Trained CNN model

- Model performance metrics (e.g., accuracy)

Steps:

1. Define the input layer:

- Create an input layer with the specified image shape.

2. Define convolutional and pooling layers:

- Add Conv2D layers with varying numbers of filters, kernel sizes, and ReLU activation functions.

- Add MaxPooling2D layers for down-sampling after each set of Conv2D layers.

3. Define fully connected layers:

- Flatten the output from the last pooling layer.

- Add Dense layers with ReLU activation functions.

- Make Dropout layers between Dense layers to prevent overfitting.

4. Define the output layer:

- Add a Dense output layer with softmax activation for multi-class classification.

5. Create the model:

- Combine the defined layers to create the custom CNN model.

6. Compile the model:

- Use an optimizer such as Adamax with a specified learning rate.

- Specify the loss function as categorical cross-entropy.

- Specify metrics such as accuracy for evaluation.

7. Train the model:

- Train the model using the training data, specifying the number of epochs and batch size.

- Validate the model using validation data.

8. Evaluate the model:

- Monitor the training and validation performance metrics, such as accuracy, over the epochs.

6.3 VGG ALGORITHM:

VGG, or Visual Geometry Group, is a deep convolutional neural network (CNN) architecture that was introduced by researchers from the University of Oxford in 2014. It is notable for its simplicity and effectiveness in image classification tasks. The model architecture consists of a series of convolutional layers with small kernel sizes (3x3) and rectified linear unit (ReLU) activation functions, followed by max-pooling layers for down-sampling. This approach enables the network to capture intricate details and patterns in images. After several convolutional layers, the model includes fully connected layers for classification. The VGG model has multiple configurations, such as VGG16 and VGG19, which denote the number of weight layers in each configuration. These models have performed exceptionally well in image recognition benchmarks like ImageNet, providing a solid foundation for further advancements in deep learning for computer vision tasks.

6.3.1 STEP BY STEP PROCESS OF VGG:

Input:

- Training data (train_gen)

- Validation data (valid_gen)

- Image shape: (224, 224, 3)

- Number of epochs: 6

Output:

- Trained VGG model

- Model performance metrics (e.g., accuracy)

Steps:

1. Define the input layer:

- Create an input layer with the specified image shape of `(224, 224, 3)`.

2. Define convolutional and pooling layers:

- Add a series of Conv2D layers with a kernel size of `3x3` and ReLU activation, increasing the number of filters in each layer.

- Use a padding of "same" in the Conv2D layers to maintain the input shape.

- After each block of Conv2D layers, add a MaxPooling2D layer with a pooling size of `(2, 2)` for down-sampling.

3. Define fully connected layers:

- Flatten the output from the last pooling layer using `Flatten()`.

- Add multiple Dense layers with ReLU activation functions for further processing.

4. Define the output layer:

- Add a Dense output layer with softmax activation and a number of units equal to the number of classes for multi-class classification.

5. Create the model:

- Combine the defined layers to create the VGG model.

6. Compile the model:

- Use an optimizer such as Adamax with a specified learning rate (e.g., `0.0001`).

- Specify the loss function as categorical cross-entropy.

- Specify metrics such as accuracy for model evaluation.

7. Train the model:

- Train the model using the training data, specifying the number of epochs (e.g., `5`) and optionally a batch size.

- Validate the model using validation data.

8. Evaluate the model:

- Monitor the training and validation performance metrics, such as accuracy, over the epochs.

6.4 MAIN CODE

6.4.1 Import Libraries:

```
import os
import time
import shutil
import pathlib
import itertools
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style("darkgrid")
import matplotlib.pyplot  as plt
from sklearn.model_selection  import   train_test_split
from sklearn.metrics import confusion_matrix , classification_report
import tensorflow  as tf
from tensorflow import  keras
from tensorflow.keras.models import  Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam ,Adamax ,AdamW
```

```
from tensorflow.keras.layers import Conv2D ,MaxPooling2D,Flatten,Dense ,Activation, Dropout,BatchNormalization

from tensorflow.keras import regularizers
```

6.4.2 DATA DIRECTORY:

```
data_dir=(r"/content/drive/MyDrive/Colab Notebooks/Lung and Colon cancer /lc25k/lung_colon_image_set")

filepaths = []

labels = []

folds=os.listdir(data_dir)

for fold in folds:

    foldpath= os.path.join(data_dir, fold)

    flist=os.listdir(foldpath)

    for f in flist:

        f_path=os.path.join(foldpath, f)

        filelist=os.listdir(f_path)

        for file in filelist:

            fpath = os.path.join(f_path, file)

            filepaths.append(fpath)

            if f=="colon_aca" :

                labels.append("Colon Adenocarcinoma")

            elif f == "colon_n":

                labels.append("Colon Benign Tissue")

            elif f=="lung_aca":
```

```
            labels.append("Lung Adenocarcinoma")
        elif f == "lung_n":
            labels.append("Lung Benign Tissue")
        elif f=="lung_scc":
            labels.append("Lung Squamous Cell Carcinoma")
Fseries= pd.Series(filepaths ,name= "file_paths")
Lseries =pd.Series(labels ,name ="Labels")
df= pd.concat ([Fseries ,Lseries],axis =1)
```

6.4.3 Split Data Into Train, Valid, Test:

```
strat =df["Labels"]
train_df ,dummy_df = train_test_split(df ,train_size= 0.8,shuffle=True , random_state =123 ,stratify =strat )
strat=dummy_df["Labels"]
valid_df,test_df = train_test_split(dummy_df ,train_size=0.5,shuffle =True , random_state =123,stratify =strat )
```

6.4.4 Create Data Image Generator:

```
batch_size =64
img_size=(224,224)
channels=3
img_shape=(224 ,224 ,3)
tre_gen =ImageDataGenerator()
ts_gen =ImageDataGenerator()
train_gen = tre_gen.flow_from_dataframe(train_df , x_col = 'file_paths' , y_col = 'Labels' , target_size = img_size ,class_mode = 'categorical' , color_mode = 'rgb' , shuffle = True , batch_size = 64)
```

```
valid_gen = tre_gen.flow_from_dataframe(valid_df , x_col = 'file_paths' , y_col = 'Labels' , target_size = img_size ,class_mode = 'categorical' , color_mode = 'rgb' , shuffle = True , batch_size = 64)

test_gen= tre_gen.flow_from_dataframe(test_df , x_col = 'file_paths' , y_col = 'Labels' , target_size = img_size ,class_mode = 'categorical' , color_mode = 'rgb' , shuffle = False , batch_size = 64)
```

6.4.5 Show Sample From Train Data:

```
g_dict =train_gen.class_indices

classes= list(g_dict)

images, labels = next(train_gen)

plt.figure(figsize=(20,20))

for i in range (16) :

    plt.subplot(4,4,i+1)

    image = images[i] /255

    plt.imshow(image)

    index = np.argmax(labels[i])

    class_name = classes[index]

    plt.title(class_name , color = "blue", fontsize =12 )

    plt.axis("off")

plt.show()
```

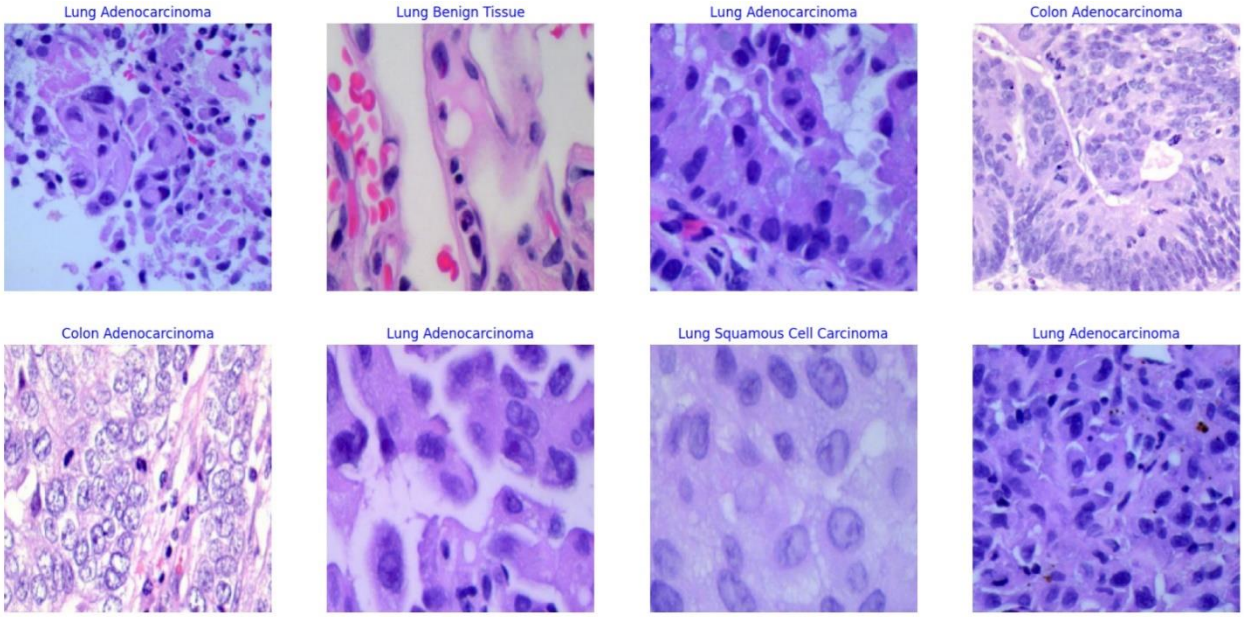

Fig 6.4.5: Sample Trained Images

6.5 Sequential Model Building:

img_size = (224,224)

channels = 3

img_shape = (224,224,3)

class_count = 5

model = tf.keras.Sequential([Conv2D(filters =64 , kernel_size=(3,3) , padding ="same" ,activation="relu" ,input_shape= img_shape),

Conv2D(filters =64 , kernel_size=(3,3) , padding = "same" , activation ="relu"),

MaxPooling2D((2,2)),

Conv2D(filters =128 , kernel_size=(3,3) , padding ="same" ,activation="relu"),

Conv2D(filters =128, kernel_size=(3,3) , padding = "same" , activation ="relu"),

Conv2D(filters =128, kernel_size=(3,3) , padding = "same" , activation ="relu"),

MaxPooling2D((2,2)),

Conv2D(filters =256, kernel_size=(3,3) , padding ="same" ,activation="relu"),

```
Conv2D(filters =256, kernel_size=(3,3) , padding = "same" , activation ="relu"),

Conv2D(filters =256, kernel_size=(3,3) , padding = "same" , activation ="relu"),

MaxPooling2D((2,2)),

Conv2D(filters =512 , kernel_size=(3,3) , padding ="same" ,activation="relu" ),

Conv2D(filters =512, kernel_size=(3,3) , padding = "same" , activation ="relu"),

Conv2D(filters =512, kernel_size=(3,3) , padding = "same" , activation ="relu"),

MaxPooling2D((2,2)),

Conv2D(filters =512 , kernel_size= (3,3) , padding ="same" ,activation="relu" ),

Conv2D(filters =512, kernel_size=(3,3) , padding = "same" , activation ="relu"),

Conv2D(filters =512, kernel_size=(3,3) , padding = "same" , activation ="relu"),

MaxPooling2D((2,2)),

Flatten(),

Dense(256 ,activation= "relu"),

Dense(64 ,activation="relu"),

Dense(256, activation ="relu"),

Dense(64 , activation = "relu"),

Dense(64,activation ="relu"),

Dense(class_count , activation="softmax")])

model.compile(Adamax(learning_rate=0.0001),loss="categorical_crossentropy",metrics=["accur
acy"])
```

6.5.1 Train Sequential Model:

```
# Train model

epochs =50

history = model.fit(x=train_gen ,epochs= epochs , verbose= 1, validation_data = valid_gen
,validation_steps =None , shuffle =False)
```

output:-

Epoch 49/50

313/313 [==============================] - 306s 978ms/step - loss: 0.0267 - accuracy: 0.9909 - val_loss: 0.0393 - val_accuracy: 0.9864

Epoch 50/50

313/313 [==============================] - 301s 961ms/step - loss: 0.0215 - accuracy: 0.9930 - val_loss: 0.0485 - val_accuracy: 0.9836

6.5.2 Evalute Model:

```
ts_length= len(test_df)

test_batch_size =max(sorted([ts_length//n for n in range (1, ts_length +1)if ts_length%n == 0 and ts_length/n <=80]))

test_steps =ts_length // test_batch_size

train_score =model.evaluate(train_gen ,steps= test_steps , verbose =1 )

valid_score =model.evaluate(valid_gen ,steps= test_steps , verbose =1 )

test_score =model.evaluate(test_gen ,steps= test_steps , verbose =1 )

print("Train Loss:" , train_score[0])

print("Traing Accuracy :",train_score[1])

print("-"*20)

print("Valid Loss:" , valid_score[0])

print("Valid Accuracy :", valid_score[1])

print("-"*20)

print("Test Loss:" , test_score[0])

print("Test Accuracy :",test_score[1])
```

output:-

50/50 [==============================] - 18s 363ms/step - loss: 0.0187 - accuracy: 0.9941

39/50 [=====================>.....] - ETA: 4s - loss: 0.0486 - accuracy: 0.9836

50/50 [==============================] - 14s 283ms/step - loss: 0.0485 - accuracy: 0.9836

39/50 [=====================>.....] - ETA: 3s - loss: 0.0733 - accuracy: 0.9800

50/50 [==============================] - 15s 283ms/step - loss: 0.0732 - accuracy: 0.9800

Train Loss: 0.01871645078063011

Traing Accuracy : 0.9940624833106995

Valid Loss: 0.04851894825696945

Valid Accuracy : 0.9836000204086304

Test Loss: 0.07319372892379761

Test Accuracy : 0.9800000190734863

6.5.3 Confusion Matrix For Classification Support:

```
g_dict =test_gen.class_indices

classes=list(g_dict.keys())

cm = confusion_matrix(test_gen.classes, y_pred)

plt.figure(figsize=(10,10))

plt.imshow(cm,interpolation="nearest",cmap = plt.cm.Blues)

plt.title("Confusion Matrix")

plt.colorbar()

tick_marks =np.arange(len(classes))

plt.xticks(tick_marks, classes ,rotation =45)

plt.yticks(tick_marks ,classes)
```

```
thresh =cm.max()/2
for i ,j in itertools.product (range(cm.shape[0]),range(cm.shape[1])):
    plt.text(j,i,cm[i,j],horizontalalignment = "center",color = "white" if cm[i,j]>thresh else "black")
    plt.tight_layout()
    plt.ylabel("True Label")
    plt.xlabel("predicted Label")
    plt.show
```

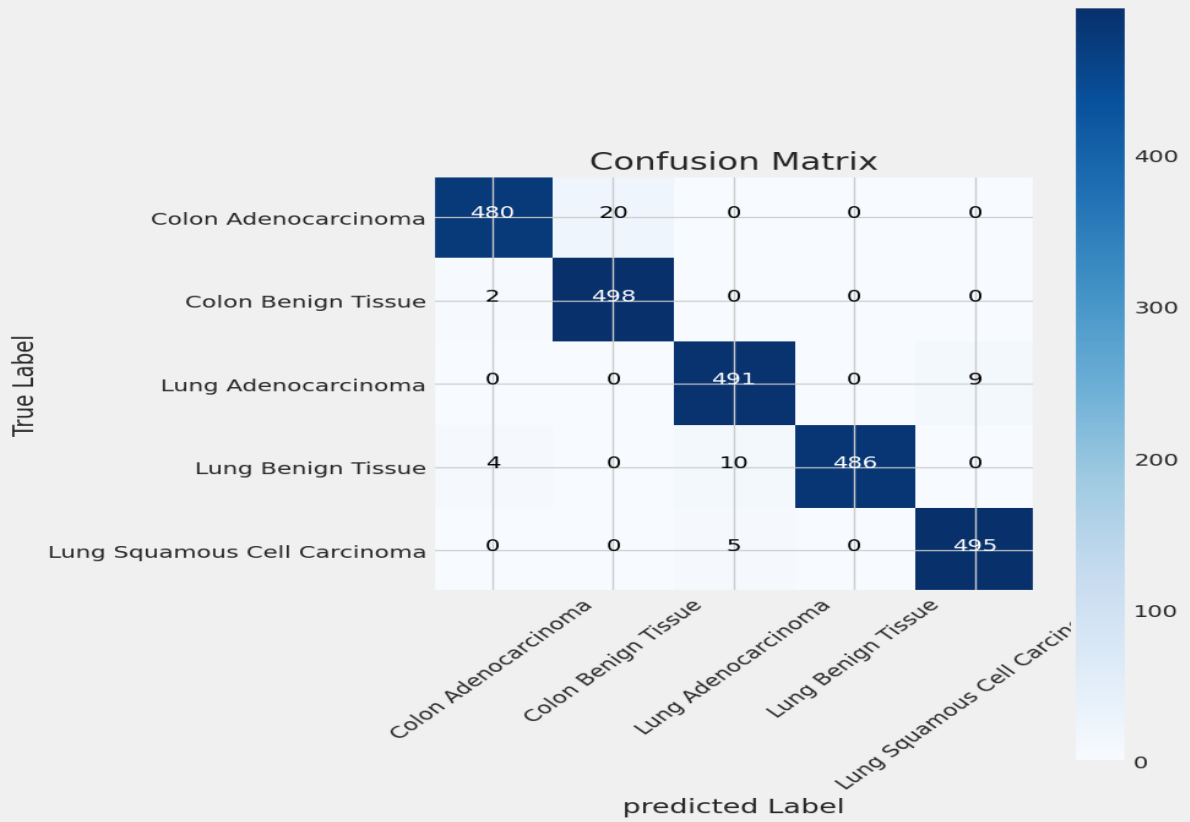

Fig 6.5.3: Confusion Matrix for Sequential

```
print(classification_report(test_gen.classes ,y_pred , target_names=classes))
```

output:-

	precision	recall	f1-score	support

Colon Adenocarcinoma	0.99	0.96	0.97	500
Colon Benign Tissue	0.96	1.00	0.98	500
Lung Adenocarcinoma	0.97	0.98	0.98	500
Lung Benign Tissue	1.00	0.97	0.99	500
Lung Squamous Cell Carcinoma	0.98	0.99	0.99	500
accuracy			0.98	2500
macro avg	0.98	0.98	0.98	2500
weighted avg	0.98	0.98	0.98	2500

6.5.4 Save Sequential Model:

model.save("/content/drive/MyDrive/fullcode/**8fullseq.h5**")

6.5.5 Prediction With Model:

loaded_model=tf.keras.models.load_model(r"/content/drive/MyDrive/**8fullseq.5h**", compile=False)

loaded_model.compile(Adamax(learning_rate=0.0001),loss="categorical_crossentropy", metrics= ["accuracy"])

image_path = r"/content/drive/MyDrive/Colab Notebooks/Lung and Colon cancer /lc25k/lung_colon_image_set/colon_image_sets/colon_aca/**colonca4949.jpeg**"
image = Image.open(image_path)

img = image.resize((224, 224))

img_array = tf.keras.preprocessing.image.img_to_array(img)

img_array = tf.expand_dims(img_array, 0)

predictions = loaded_model.predict(img_array) **#1st model test result**

class_labels = classes

score = tf.nn.softmax(predictions[0])

Get the predicted class index and the corresponding label

predicted_class_index = tf.argmax(score)

```
predicted_class_label = class_labels[predicted_class_index]

# Get the confidence score (percentage)

confidence_percentage = score[predicted_class_index] * 100

print(f"Predicted Class: {predicted_class_label}")

print(f"Confidence: {confidence_percentage:.2f}%")    # Check if the predicted class is related to lung or colon cancer

    lung_classes = ["Lung Adenocarcinoma", "Lung Benign Tissue", "Lung Squamous Cell Carcinoma"]

   colon_classes = ["Colon Adenocarcinoma", "Colon Benign Tissue"]

   if predicted_class_label in lung_classes:

      print("The uploaded image is related to lung cancer.")

   elif predicted_class_label in colon_classes:

      print("The uploaded image is related to colon cancer.")

   else:

      print("The uploaded image does not appear to be a histopathological image of lung or colon cancer.")
```

output:-

1/1 [==============================] - 1s 983ms/step

Predicted Class: Colon Adenocarcinoma

Confidence: 97.20%

The uploaded image is related to colon cancer.

6.6 Building Resnet Model:

```
# Define constants
img_shape = (224, 224, 3)
class_count = 5
epochs = 6
batch_size = 36
# Define the input layer
inputs = Input(shape=img_shape)
# Define the convolutional layers
conv1 = Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu')(inputs)
conv2 = Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv2)
conv3 = Conv2D(filters=128, kernel_size=(3, 3), padding='same', activation='relu')(pool1)
conv4 = Conv2D(filters=128, kernel_size=(3, 3), padding='same', activation='relu')(conv3)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv4)
conv5 = Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation='relu')(pool2)
conv6 = Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation='relu')(conv5)
```

```
conv7 = Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation='relu')(conv6)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv7)
conv8 = Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu')(pool3)
conv9 = Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu')(conv8)
conv10 = Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu')(conv9)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv10)
conv11 = Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu')(pool4)
conv12 = Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu')(conv11)
conv13 = Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu')(conv12)
pool5 = MaxPooling2D(pool_size=(2, 2))(conv13)
# Flatten layer
flatten = Flatten()(pool5)
# Define the dense layers
dense1 = Dense(256, activation='relu')(flatten)
dropout1 = Dropout(0.5)(dense1)
dense2 = Dense(64, activation='relu')(dropout1)
dropout2 = Dropout(0.5)(dense2)
# Output layer
outputs = Dense(class_count, activation='softmax')(dropout2)
# Create the model
model2 = Model(inputs=inputs, outputs=outputs)
# Compile the model
model2.compile(optimizer=Adamax(learning_rate=0.0001),
               loss='categorical_crossentropy',
              metrics=['accuracy'])
```

6.6.1 Train The Resnet Model:

history = model2.fit(train_gen, epochs=epochs, batch_size=batch_size, verbose=1, validation_data=valid_gen)

output:-

Epoch 5/6

313/313 [==============================] - 4692s 15s/step - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.0267 - val_accuracy: 0.9844

Epoch 6/6

313/313 [==============================] - 4557s 15s/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0039 - val_accuracy: 0.99

6.6.2 Evalute Resnet Model:

output:-

40/50 [=========================>.....] - ETA: 2:18 - loss: 0.0179 - accuracy: 0.9928

50/50 [==============================] - 555s 11s/step - loss: 0.0179 - accuracy: 0.9928

Train Loss: 0.0006370760384015739

Traing Accuracy : 0.99

Valid Loss: 0.01783294789493084

Valid Accuracy : 0.9927999973297119

Test Loss: 0.017920156940817833

Test Accuracy : 0.9927999973297119

6.6.3 Confusion Matrix For Classification Support:

```
preds = model2.predict(test_gen)

y_pred = np.argmax(preds , axis=1 )

from sklearn.metrics import classification_report

g_dict =test_gen.class_indices

classes=list(g_dict.keys())

cm2 = confusion_matrix(test_gen.classes, y_pred)

plt.figure(figsize=(10,10))

plt.imshow(cm2,interpolation="nearest",cmap = plt.cm.Blues)

plt.title("Confusion Matrix")

plt.colorbar()

tick_marks =np.arange(len(classes))

plt.xticks(tick_marks, classes ,rotation =45)

plt.yticks(tick_marks ,classes)

thresh =cm2.max()/2

for i ,j in itertools.product (range(cm2.shape[0]),range(cm2.shape[1])):

   plt.text(j,i,cm22[i,j],horizontalalignment = "center",color = "white" if cm2[i,j]>thresh else "black")

   plt.tight_layout()

   plt.ylabel("True Label")

   plt.xlabel("predicted Label")

   plt.show
```

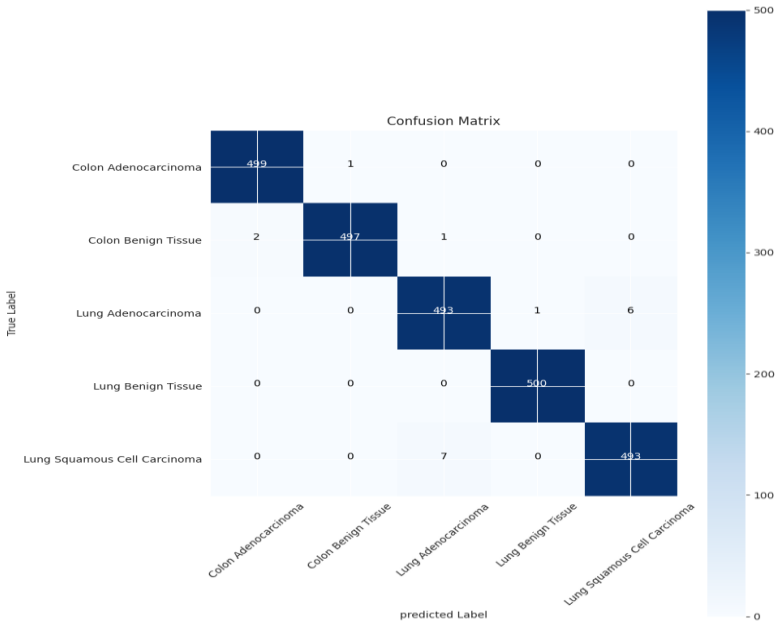

Fig 6.6.3: Confusion Matrix for ResNet

Print(classification_report(test_gen.classes, y_pred, target_names=classes))

output:-

	precision	recall	f1-score	support
Colon Adenocarcinoma	1.00	1.00	1.00	500
Colon Benign Tissue	1.00	0.99	1.00	500
Lung Adenocarcinoma	0.98	0.99	0.99	500
Lung Benign Tissue	1.00	1.00	1.00	500
Lung Squamous Cell Carcinoma	0.99	0.99	0.99	500
accuracy			0.99	2500
macro avg	0.99	0.99	0.99	2500
weighted avg	0.99	0.99	0.99	2500

6.6.4 Save ResNet Model:

model2.save("/content/drive/MyDrive/**ResnetModel3.5h**")

6.6.5 Predict With ResNet Model:

loaded_model2=tf.keras.models.load_model("/content/drive/MyDrive/**ResnetModel3.5h"** ,compile=False)

loaded_model2.compile(Adamax(learning_rate=0.0001),loss="categorical_crossentropy", metrics= ["accuracy"])

from PIL import Image

image_path = r"/content/drive/MyDrive/Colab Notebooks/**lungn23.jpeg**"

image = Image.open(image_path)

img = image.resize((224, 224))

img_array = tf.keras.preprocessing.image.img_to_array(img)

img_array = tf.expand_dims(img_array, 0)

test result

predictions2 = loaded_model2.predict(img_array)

class_labels = classes

score = tf.nn.softmax(predictions2[0])

Get the predicted class index and the corresponding label **#2nd model**

predicted_class_index = tf.argmax(score)

predicted_class_label = class_labels[predicted_class_index]

Get the confidence score (percentage)

confidence_percentage = score[predicted_class_index] * 100

print(f"Predicted Class: {predicted_class_label}")

print(f"Confidence: {confidence_percentage:.2f}%")

output:-

1/1 [==============================] - 2s 2s/step

Predicted Class: Lung Benign Tissue

Confidence: 97.46%

6.7 Building VGG Model:

```
# Define constants
img_shape = (224, 224, 3)
class_count = 5
epochs = 5
# Define the input layer
inputs = Input(shape=img_shape)
# Define the convolutional layers
conv1 = Conv2D(filters=64, kernel_size=(3, 3), padding="same", activation="relu")(inputs)
conv2 = Conv2D(filters=64, kernel_size=(3, 3), padding="same", activation="relu")(conv1)
pool1 = MaxPooling2D((2, 2))(conv2)
conv3 = Conv2D(filters=128, kernel_size=(3, 3), padding="same", activation="relu")(pool1)
```

```
conv4 = Conv2D(filters=128, kernel_size=(3, 3), padding="same", activation="relu")(conv3)

pool2 = MaxPooling2D((2, 2))(conv4)

conv5 = Conv2D(filters=256, kernel_size=(3, 3), padding="same", activation="relu")(pool2)

conv6 = Conv2D(filters=256, kernel_size=(3, 3), padding="same", activation="relu")(conv5)

conv7 = Conv2D(filters=256, kernel_size=(3, 3), padding="same", activation="relu")(conv6)

pool3 = MaxPooling2D((2, 2))(conv7)

conv8 = Conv2D(filters=512, kernel_size=(3, 3), padding="same", activation="relu")(pool3)

conv9 = Conv2D(filters=512, kernel_size=(3, 3), padding="same", activation="relu")(conv8)

conv10 = Conv2D(filters=512, kernel_size=(3, 3), padding="same", activation="relu")(conv9)
```

#3rd model

```
pool4 = MaxPooling2D((2, 2))(conv10)

conv11 = Conv2D(filters=512, kernel_size=(3, 3), padding="same", activation="relu")(pool4)

conv12 = Conv2D(filters=512, kernel_size=(3, 3), padding="same", activation="relu")(conv11)

conv13 = Conv2D(filters=512, kernel_size=(3, 3), padding="same", activation="relu")(conv12)

pool5 = MaxPooling2D((2, 2))(conv13)

# Flatten layer

flatten = Flatten()(pool5)

# Dense layers

dense1 = Dense(256, activation="relu")(flatten)

dense2 = Dense(64, activation="relu")(dense1)

dense3 = Dense(256, activation="relu")(dense2)

dense4 = Dense(64, activation="relu")(dense3)

dense5 = Dense(64, activation="relu")(dense4)

# Output layer

outputs = Dense(class_count, activation="softmax")(dense5)
```

```
# Create the model
model3 = Model(inputs=inputs, outputs=outputs)
# Compile the model
model3.compile(optimizer=Adamax(learning_rate=0.0001),
        loss="categorical_crossentropy",
        metrics=["accuracy"])
```

6.7.1 Train The VGG Model:

```
history = model3.fit(train_gen, epochs=epochs, verbose=1, validation_data=valid_gen)
```

output:-

```
Epoch 4/5
313/313 [==============================] - 269s 859ms/step - loss: 0.2510 - accuracy: 0.9004 - val_loss: 0.2368 - val_accuracy: 0.9044
Epoch 5/5
313/313 [==============================] - 269s 857ms/step - loss: 0.2137 - accuracy: 0.9143 - val_loss: 0.1659 - val_accuracy: 0.9336
```

6.7.2 Evalute Model:

Output:-

```
50/50 [==============================] - 26s 519ms/step - loss: 0.2069 - accuracy: 0.9241
39/50 [======================>.......] - ETA: 5s - loss: 0.2002 - accuracy: 0.9267
50/50 [==============================] - 20s 386ms/step - loss: 0.2001 - accuracy: 0.9268
39/50 [======================>.......] - ETA: 5s - loss: 0.2044 - accuracy: 0.9303
```

50/50 [==============================] - 21s 406ms/step - loss: 0.2043 - accuracy: 0.9304

Train Loss: 0.20688501000404358

Traing Accuracy : 0.9240624904632568

Valid Loss: 0.2000991255044937

Valid Accuracy : 0.926800012588501

Test Loss: 0.20427671074867249

Test Accuracy : 0.930400013923645

6.7.3 Confusion Matrix For Classification Support:

```
g_dict =test_gen.class_indices
classes=list(g_dict.keys())
cm3 = confusion_matrix(test_gen.classes, y_pred)
plt.figure(figsize=(10,10))
plt.imshow(cm2,interpolation="nearest",cmap = plt.cm.Blues)
plt.title("Confusion Matrix")
plt.colorbar()
tick_marks =np.arange(len(classes))
plt.xticks(tick_marks, classes ,rotation =45)
plt.yticks(tick_marks ,classes)
thresh =cm3.max()/2
for i ,j in itertools.product (range(cm3.shape[0]),range(cm3.shape[1])):
    plt.text(j,i,cm2[i,j],horizontalalignment = "center",color = "white" if cm2[i,j]>thresh else "black")
```

```
    plt.tight_layout()
    plt.ylabel("True Label")
    plt.xlabel("predicted Label")
    plt.show
# Evaluate the model on the test generator
#prediction
preds=model3.predict(test_gen)
y_pred = np.argmax(preds , axis=1 )
```

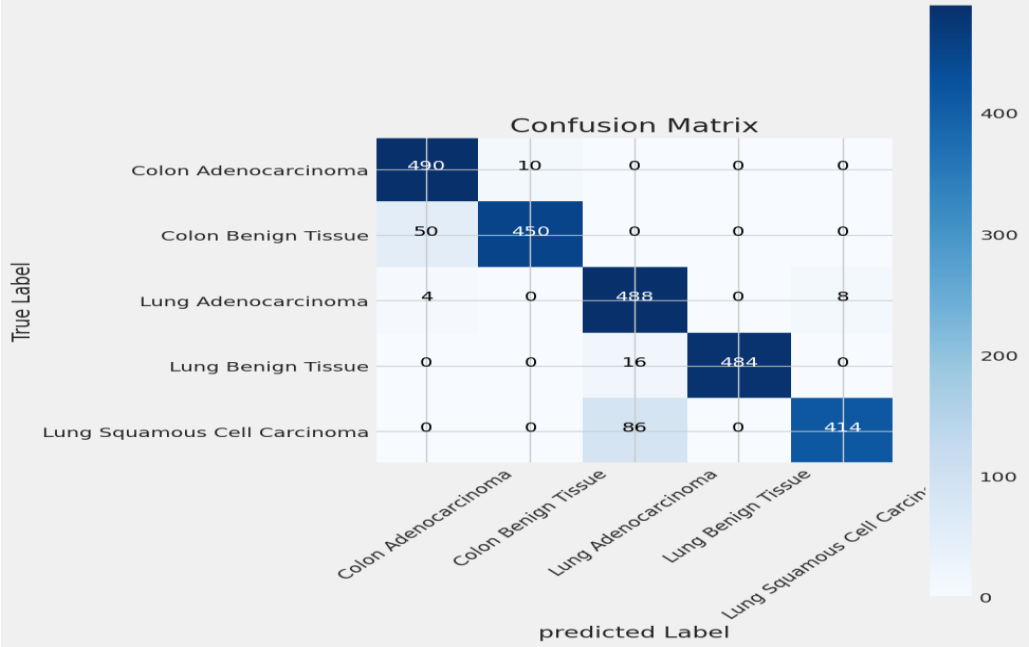

Fig 6.7.3: Confusion Matrix for VGG Model

```
Print(classification_report(test_gen.classes, y_pred, target_names=classes))
```

output:-

	precision	recall	f1-score	support
Colon Adenocarcinoma	0.90	0.98	0.94	500
Colon Benign Tissue	0.98	0.90	0.94	500
Lung Adenocarcinoma	0.83	0.98	0.90	500

Lung Benign Tissue	1.00	0.97	0.98	500
Lung Squamous Cell Carcinoma	0.98	0.83	0.90	500
accuracy			0.93	2500
macro avg	0.94	0.93	0.93	2500
weighted avg	0.94	0.93	0.93	2500

6.7.4 Save VGG Model:

model3.save("/content/drive/MyDrive/**vggModel3.5h**")

6.7.5 Predict With VGG Model:

loaded_model3=tf.keras.models.load_model("/content/drive/MyDrive/**vggModel3.5h**" ,compile=False)

loaded_model3.compile(Adamax(learning_rate=0.0001),loss="categorical_crossentropy", metrics= ["accuracy"])

from PIL import Image

image_path = r"/content/drive/MyDrive/Colab Notebooks/**colonca72.jpeg**"

predictions3 = loaded_model3.predict(img_array)

output:-

1/1 [==============================] - 1s 874ms/step

Predicted Class: Colon Adenocarcinoma

Confidence: 33.64%

6.8 Results And Performance Metrices:

Performance metrics are important tools for evaluating the effectiveness and quality of machine learning and deep learning models, particularly in classification tasks. These metrics provide insights into how well a model is performing, both during training and when applied to new, unseen data. Here are some common performance metrics used in classification tasks:

Here's a summary of common performance metrics used in classification tasks, along with their formulas:

1. **Accuracy**:
    - Accuracy measures the proportion of correctly classified instances out of the total number of instances.
    - Formula:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

2. **Precision**:
    - Precision measures the proportion of true positive predictions out of all positive predictions made by the model.
    - Formula:

$$Precision = \frac{TP}{TP + FP}$$

3. **Recall**:
    - Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that the model correctly identifies.
    - Formula:

$$Recall = \frac{TP}{TP + FN}$$

4. **F1-Score**:
    - The F1-Score is the harmonic mean of precision and recall, providing a balanced measure that considers both false positives and false negatives.
    - Formula:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

5. **Support**:
    - Support refers to the total number of actual instances of a class in the dataset.
    - It is important for understanding the prevalence of each class and is used to weigh other metrics like precision and recall.
    - It is not a formula but rather a count of actual instances for each class.

These metrics help evaluate the model's performance, especially for tasks involving imbalanced datasets. By considering accuracy, precision, recall, and F1-score, you can gain a comprehensive view of the model's effectiveness in classifying different classes.

6.9 Comparing Accuracies For Three Models:

Model Name	Accuracy
Sequential Model	0.9800000190734863
Resnet50	0.9727999973297119
VGG	0.9304000139236453

6.10 Gradio Code:

2gradio.py

```
import gradio as gr

import tensorflow as tf

import numpy as np

from PIL import Image

import io

# Load the pre-trained models

loaded_model1 = tf.keras.models.load_model("8fullseq.h5", compile=False)

loaded_model1.compile(tf.keras.optimizers.Adamax(learning_rate=0.0001),
loss="categorical_crossentropy", metrics=["accuracy"])

loaded_model2 = tf.keras.models.load_model("ResnetModel3.5h ", compile=False)

loaded_model2.compile(tf.keras.optimizers.Adamax(learning_rate=0.0001),
loss="categorical_crossentropy", metrics=["accuracy"])

# Define the classes for each model

classes_model1 = ["Lung Adenocarcinoma", "Lung Benign Tissue", "Lung Squamous Cell
Carcinoma", "Colon Adenocarcinoma", "Colon Benign Tissue"]
```

```
classes_model2 = ["Lung Adenocarcinoma", "Lung Benign Tissue", "Lung Squamous Cell Carcinoma", "Colon Adenocarcinoma", "Colon Benign Tissue"]

# Define the prediction function

def predict_with_model(image, model_choice):

    # Convert the image to a PIL Image object

    image_rgb = image.convert("RGB")

    img = image_rgb.resize((224, 224))

    img_array = tf.keras.preprocessing.image.img_to_array(img)

    img_array = tf.expand_dims(img_array, 0)

    # Choose the model and classes based on user selection

    if model_choice == "Sequential":

        model = loaded_model1

        classes = classes_model1

    elif model_choice == "ResNet50":

        model = loaded_model2

        classes = classes_model2

    # Perform the prediction

    predictions = model.predict(img_array)

    predicted_class_index = np.argmax(predictions)

    predicted_class_label = classes[predicted_class_index]

    confidence_percentage = predictions[0][predicted_class_index] * 100

    # Define the lists of lung and colon cancer classes

    lung_classes = ["Lung Adenocarcinoma", "Lung Benign Tissue", "Lung Squamous Cell Carcinoma"]

    colon_classes = ["Colon Adenocarcinoma", "Colon Benign Tissue"]

    # Determine the cancer type
```

```
    if predicted_class_label in lung_classes:
        cancer_type = "Lung Cancer"
    elif predicted_class_label in colon_classes:
        cancer_type = "Colon Cancer"
    else:
        cancer_type = "Other"
    return f"Predicted Cancer type: {predicted_class_label}, Confidence: {confidence_percentage:.2f}%, Cancer: {cancer_type}"

# Create the Gradio interface
image_input = gr.Image(label="Upload Image", type="pil", width=850, height=750)  # Set the input image size
output_text = gr.Textbox(label="Prediction")

# Create the model selection dropdown
model_choice = gr.Dropdown(["Sequential", "ResNet50"], label="Select Model", value="Sequential")
theme = gr.themes.Default()
theme.background_image = "archieture.png"
# Create the Gradio interface and launch it
gr.Interface(
    fn=predict_with_model,
    inputs=[image_input, model_choice],
    outputs=output_text,
    title="Histopathological Image Classifier",
    theme=theme  # Apply the custom theme
).launch(debug=True
```

6.11 FRONTEND CODE:

Front_test.html

```
</html><!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cancer Detection</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>
<body style="background: linear-gradient(to bottom right, #6366f1, #8b5cf6);
font: 15px/1.5 Arial, Helvetica, sans-serif; text-align: center;">

  <div class="bg-gradient-to-br from-blue-400 to-purple-500 min-h-screen">
   <!-- <div class="max-w-lg mx-auto p-8 bg-white rounded-lg shadow-lg text-
center main-container"> -->
      <h1 class="text-3xl font-bold italic mb-4" style="text-decoration:
underline double; position: relative;">
        <span class="background-text" style="position: absolute; top: 0; left: 0;
right: 0; bottom: 0; ; z-index: -1;"></span>
        <div style="background: linear-gradient(to right, #007bff, #28a745);
            display: inline-block;
            color: #151a15;
            font-style: bold  ;
            padding: 10px;
```

```
            font-family: 'Arial', sans-serif;
            font-size: 18px;
            font-weight: 900;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
            text-transform: uppercase;">
  Lung and Colon Cancers Detection
</div>

      </h1><br>

      <p class="text-sm text-gray-500 mb-4 pt-2 text-description">
        Upload histopathological images to detect lung or colon cancer.
      </p>
      <!-- Placeholder for prediction results -->
      <div id="predictionResults" class="border-dashed border-2 border-gray-300
p-8 mb-4">
        <!-- Prediction results will be displayed here -->
      </div>
      <form id="uploadForm" enctype="multipart/form-data" action="/predict"
method="POST">
        <input type="file" id="fileInput" accept="image/*" style="display: none;"
name="file">
        <div class="upload-container">
          <div class="image-container">
            <img src="https://www.saintjohnscancer.org/thoracic/wp-
content/uploads/sites/5/2023/08/4-stages-of-lung-cancer-Saint-Johns-Cancer-
Institute.png" alt="Left Image" width="500" height="450" style="float: left;">
            <div id="overlayText" class="overlay-text">Smoking is the number one
cause of lung cancer.
              It causes about <b style="color: rgb(255, 0, 0); font-weight:
bold;">90 percent</b> of lung cancer cases.
              <b style="color: rgb(236, 15, 15); font-weight: bold;">Tobacco
smoke contains many chemicals that are known to cause lung cancer.</b>
              If you still smoke, quitting smoking is the single best thing you
can do for your lung health</div> <!-- Add overlay text -->
          </div>
          <img
src="https://cdnintech.com/media/chapter/59926/1512345123/media/F1.png"
alt="cancer Image" id="previewImage" class="ml-4 w-16 h-16 rounded-full"
onclick="document.getElementById('fileInput').click();">
          <div class="image-container">
            <img src="{{ url_for('static', filename='1.jpeg') }}" alt="Right
Image" width="500" height="400">
```

```
            <div id="overlayText" class="overlay-text">Overall, about <b
style="color: rgb(255, 0, 0); font-weight: bold;">90% </b>of colorectal cancers
are caused by environmental exposures, such as a <b style="color: rgb(228, 78,
19); font-weight: bold;">low-fiber and high-fat diet, alcohol consumption, and
tobacco use</b>, which occurred decades before a patient was diagnosed.
Researchers recently linked dietary metabolites to a protective or
procarcinogenic environment in the colon.</div> <!-- Add overlay text -->
          </div>
        </div>
        <br>
        <button class="px-4 py-2 bg-blue-500 text-white rounded-lg hover:bg-blue-
700 focus:outline-none focus:ring-2 focus:ring-blue-500 mb-2" type="button"
onclick="document.getElementById('fileInput').click();">Upload Image</button>
        <!-- Submit button to predict -->
        <button class="px-4 py-2 bg-blue-500 text-white rounded-lg hover:bg-blue-
700 focus:outline-none focus:ring-2 focus:ring-blue-500 mb-2"
type="submit">Submit</button>
      </form>
    </div>
  </div>
  <div class="prediction-result">
    <!-- Prediction results will be displayed here -->
  </div>
  <div style="background: linear-gradient(to bottom right, #63B3ED, #68D391);">
    <h1 font="font-bold" style="font-family: Arial, Helvetica, sans-serif; text-
shadow: 2px 2px 2px rgb(22, 167, 230); padding-right: 0cap; align-items:
center;">About the Cancer</h1>
    <p>Lung and Colon Cancer:</p>
    <h1 font="font-bold" style="color: #121212; text-shadow: 2px 2px 2px rgba(72,
97, 157, 0.942); align-self: center;">Lung Cancer:</h1>
    <p>Lung cancer is a type of cancer that begins in the lungs. It is one of the
leading causes of cancer-related deaths worldwide. The primary cause of lung
cancer is smoking, although non-smokers can also develop the disease due to
exposure to secondhand smoke, environmental toxins, or genetic factors. There are
two main types of lung cancer: non-small cell lung cancer (NSCLC) and small cell
lung cancer (SCLC). Symptoms of lung cancer may include persistent cough, chest
pain, shortness of breath, coughing up blood, and fatigue. Diagnosis typically
involves imaging tests such as X-rays and CT scans, along with biopsy for
confirmation. Treatment options include surgery, chemotherapy, radiation therapy,
targeted therapy, and immunotherapy, depending on the stage and type of
cancer.</p>
    <h1 font="font-bold" style="color: #121212; text-shadow: 2px 2px 2px
rgba(102, 77, 202, 0.942);">Colon Cancer:</h1>
    <p>Colon cancer, also known as colorectal cancer, is a type of cancer that
begins in the colon or rectum. It is the third most common cancer diagnosed in
```

```
both men and women globally. The exact cause of colon cancer is unclear, but it is believed to develop from precancerous growths called polyps in the colon or rectum. Risk factors for colon cancer include age, family history, inflammatory bowel disease, a diet high in red and processed meats, obesity, smoking, and lack of physical activity. Symptoms of colon cancer may include changes in bowel habits, rectal bleeding, abdominal discomfort, fatigue, and unexplained weight loss. Screening for colon cancer is essential for early detection, typically done through colonoscopy, sigmoidoscopy, or stool tests. Treatment options for colon cancer include surgery to remove the tumor, chemotherapy, radiation therapy, targeted therapy, and immunotherapy, depending on the stage and extent of the cancer. Both lung and colon cancer can be life-threatening if not detected and treated early. Therefore, awareness of risk factors, regular screenings, and prompt medical attention for symptoms are crucial for improving outcomes and reducing mortality rates associated with these cancers.</p>
  </div>
  <div class="footer">
    <p>Powered by Hemsun</p>
  </div>

   <script>
    // JavaScript code to handle file input change
    document.getElementById('fileInput').addEventListener('change',
function(event) {
      var file = event.target.files[0];
      if (!file.type.match('image/png') && !file.type.match('image/jpeg')) {
        alert('Please select a PNG or JPG image file.');
        document.getElementById('fileInput').value = ''; // Clear the file input
      } else {
        var reader = new FileReader();
        reader.onload = function(e) {
          document.getElementById('previewImage').src = e.target.result;
          // Hide overlay text when an image is uploaded
          document.querySelectorAll('.overlay-text').forEach(function(element) {
            element.style.display = 'none';
          });
        };
        reader.readAsDataURL(file);
      }
    });
  </script>
</body>
</html>
```

Style.css:

```
button {
    padding: 12px 24px; /* Adjust padding to increase button size */
    font-size: 16px; /* Adjust font size as needed */
    border: 2px solid #5797d7; /* Set border width and color */
    background-color: #6993c7; /* Set background color */
    color: rgb(255, 255, 255); /* Set text color */
  }

  button:hover {
    background-color: #357EC7; /* Change background color on hover */
  }


      .overlay-text {
        position: absolute;
        top: 26%;
        left: 50%;
        transform: translate(-50%, -50%);
        color: rgb(114, 229, 255);

        font-size: 20px;
        font-weight: bold;
        text-shadow: 2px 2px 4px rgba(1, 1, 0, 1); /* Add text shadow for
visibility */
        display: none;
        width: -5rem;
        height: -9rem;/* Set width to 4 times the size of 1rem */
        font-weight: bold; /* Make the text bold */ /* Hide overlay text by
default */

        padding: 10px; /* Add padding to create space between text and border */
        border-radius: 5px; /* Optional: Add border radius for rounded corners */
      }

      .image-container:hover .overlay-text {
        display: block; /* Show overlay text on hover */
      }

      /* Add custom CSS for the layout */
      .main-container {
        display: flex;
```

```
        flex-direction: column;
        align-items: center;
        margin-top: 10px; /* Reduce margin top */
      }
      .image-container {
        padding: 1%; /* Reduce padding */
        text-align: center;
        margin-top: 10px; /* Reduce margin top */
      }
      .upload-container {
        display: flex;
        justify-content: center;
        align-items: center;
        margin-top: 10px; /* Reduce margin top */
      }
      .prediction-result {
        margin-top: 10px; /* Reduce margin top */
        text-align: center; /* Center the prediction results */
      }
      body{
          background-color: pink;
          margin: 0; /* Remove default body margin */
          padding: 0; /* Remove default body padding */

     .text-description {
        margin-top: -23px; /* Adjust this value to move the text higher */
     }

      }
      /* Fixed size for uploaded image */
      #previewImage {
        width: 400px; /* Change this value to set the width */
        height: auto; /* Automatically adjust height to maintain aspect ratio */
      }
```

Prediction1.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Results</title>
    <style>
        body {
            background: linear-gradient(to bottom right, #6366f1, #8b5cf6);
            font: 15px/1.5 Arial, Helvetica, sans-serif;
            text-align: center;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            margin: 0;
        }

        .result-container {
            margin-top: 20px;
        }

        .result-container img {
            max-width: 100%;
            height: auto;
            max-height: 400px;
        }

        .result-card {
            background-color: #fff;
            border-radius: 8px;
            padding: 20px;
            margin-bottom: 20px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }
    </style>
</head>

<body>
    <div>
        <h1>Predicted Results</h1>
        <div class="result-container">
            <img src="{{ image_path }}" alt="Uploaded Image">

            <!-- Results from 8fullseq model -->
            <div class="result-card">
                <h2>Results from Sequential Model</h2>
```

```
                <p>Predicted Class: {{ predicted_class_8fullseq }}</p>
                <p>Confidence: {{ confidence_percentage_8fullseq }}%</p>
                <p>Cancer Type: {{ cancer_type_8fullseq }}</p>
            </div>

            <!-- Results from ResNet50 model -->
            <div class="result-card">
                <h2>Results from ResNet50 Model</h2>
                <p>Predicted Class: {{ predicted_class_resnet50 }}</p>
                <p>Confidence: {{ confidence_percentage_resnet50 }}%</p>
                <p>Cancer Type: {{ cancer_type_resnet50 }}</p>
            </div>
        </div>
        <a href="/">Back to Home Page</a>
    </div>
</body>

</html>
```

6.12 FLASK CODE:

flasktest.py:

```
from flask import Flask, request, jsonify, render_template

import os

import tensorflow as tf

from PIL import Image

import numpy as np


app = Flask(__name__)

# Load the trained model (8fullseq)

model = tf.keras.models.load_model("8fullseq.h5", compile=False)

# Load the ResNet50 model

model2 = tf.keras.models.load_model(r"ResnetModel3.h5", compile=False)

# Define the classes for prediction
```

```
classes = ["Colon Adenocarcinoma", "Colon Benign Tissue", "Lung Adenocarcinoma", "Lung Benign Tissue", "Lung Squamous Cell Carcinoma"]

# Define a function for preprocessing the image

def preprocess_image(image_path):

    try:

        image = Image.open(image_path)

        # Convert the image to RGB

        image = image.convert("RGB")

        # Resize the image to match the model input size

        image = image.resize((224, 224))

        # Normalize the pixel values

        image = np.array(image) / 255.0

        return image

    except Exception as e:

        print("Error processing image:", e)

        return None

# Define a route for the homepage

@app.route('/')

def home():

    return render_template('front_test.html')

@app.route('/predict', methods=['POST'])

def predict_with_model():

    file = request.files['file']

    if file:

        # Save the uploaded file
```

```
        file_path = os.path.join("static", file.filename)
        file.save(file_path)
        image = Image.open(file_path)
        # Preprocess the image
        image_rgb = image.convert("RGB")  # Ensure image is in RGB format
        img = image_rgb.resize((224, 224))
        img_array = tf.keras.preprocessing.image.img_to_array(img)
        img_array = tf.expand_dims(img_array, 0)
        # Call prediction functions and obtain results
        predictions = model.predict(img_array)
        predicted_class_index = np.argmax(predictions)
        predicted_class_label = classes[predicted_class_index]
        confidence_percentage = predictions[0][predicted_class_index] * 100
        predictions2 = model2.predict(img_array)
        predicted_class_index2 = np.argmax(predictions2)
        predicted_class_label2 = classes[predicted_class_index2]
        confidence_percentage2 = predictions2[0][predicted_class_index2] * 100
        # Define function to determine cancer type
        def determine_cancer_type(predicted_class):
            if predicted_class in ["Lung Adenocarcinoma", "Lung Squamous Cell Carcinoma" ,
"Lung Benign Tissue" ]:
                return "Lung Cancer"
            elif predicted_class in ["Colon Adenocarcinoma", "Colon Benign Tissue"]:
                return "Colon Cancer"
            else:
```

```
            return "Other"
        # Determine cancer type for each model's prediction
        cancer_type_8fullseq = determine_cancer_type(predicted_class_label)
        cancer_type_resnet50 = determine_cancer_type(predicted_class_label2)
        # Return results to the template
        return render_template('prediction1.html',
                               image_path=file_path,
                               predicted_class_8fullseq=predicted_class_label,
                               confidence_percentage_8fullseq=confidence_percentage,
                               cancer_type_8fullseq=cancer_type_8fullseq,
                               predicted_class_resnet50=predicted_class_label2,
                               confidence_percentage_resnet50=confidence_percentage2,
                               cancer_type_resnet50=cancer_type_resnet50)
    else:
        return "No file uploaded"

if __name__ == '__main__':
    app.run(debug=True)
```

CHAPTER 7

OUTPUT SCREENS

7.1 Gradio output screens:

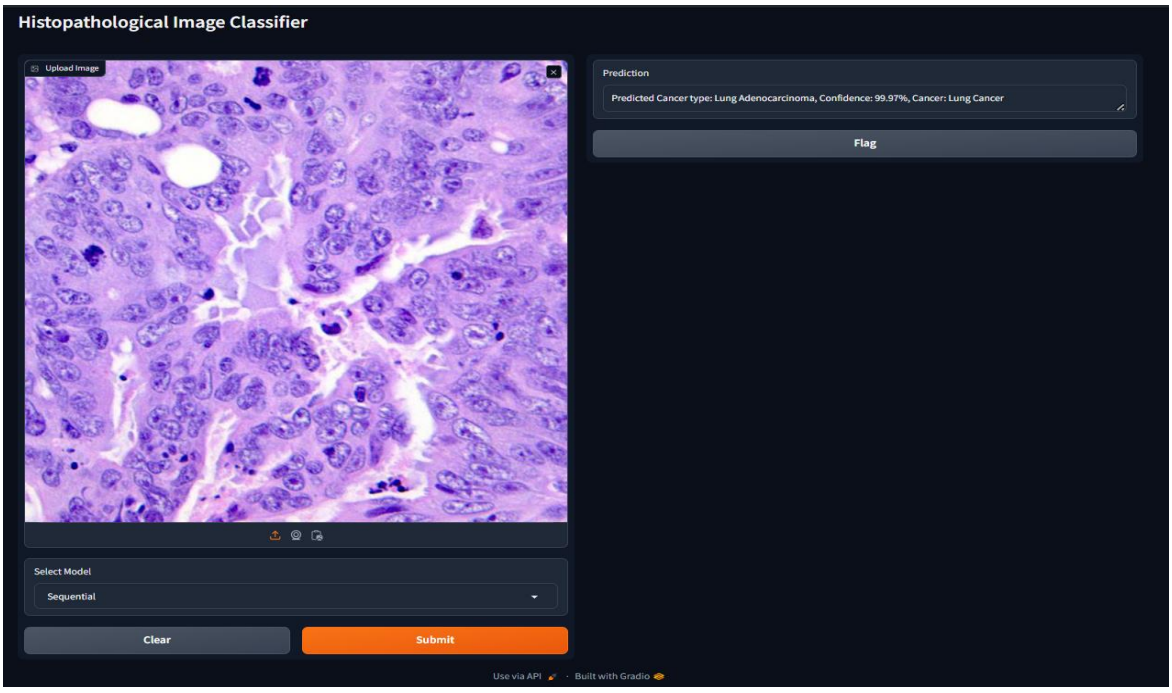

Fig 7.1.A: Sequential Model

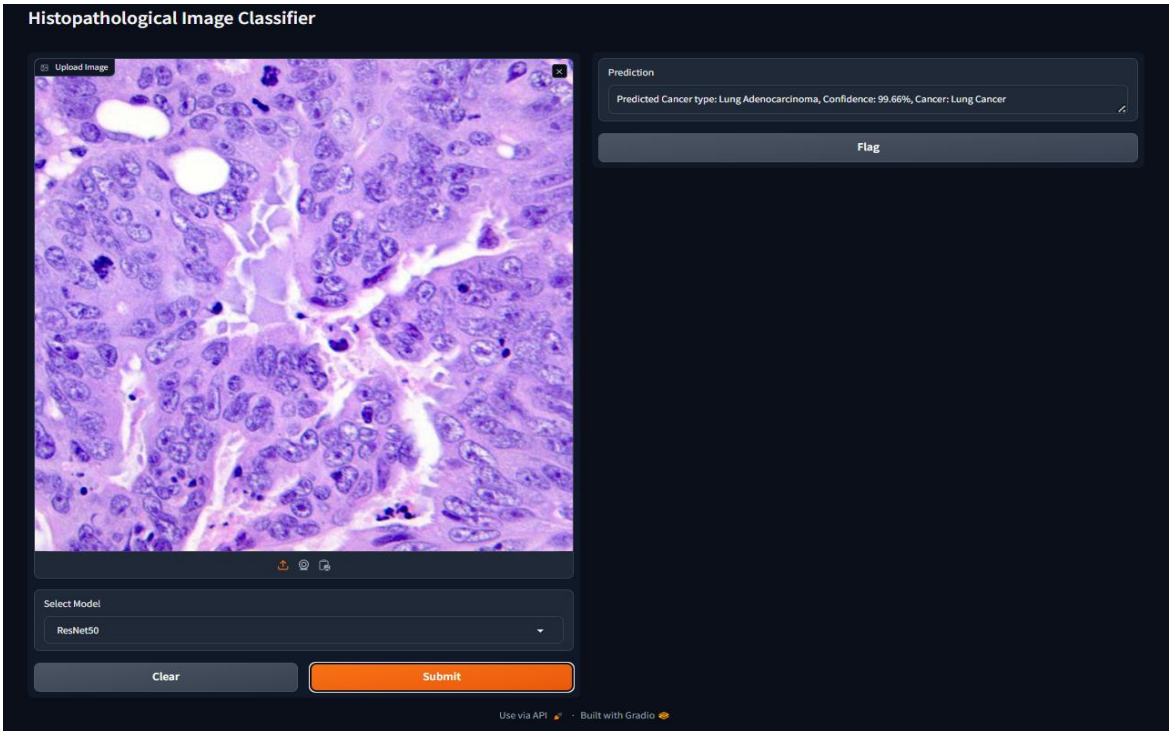

Fig 7.1.B: ResNet Model

7.2 Frontend outputs:

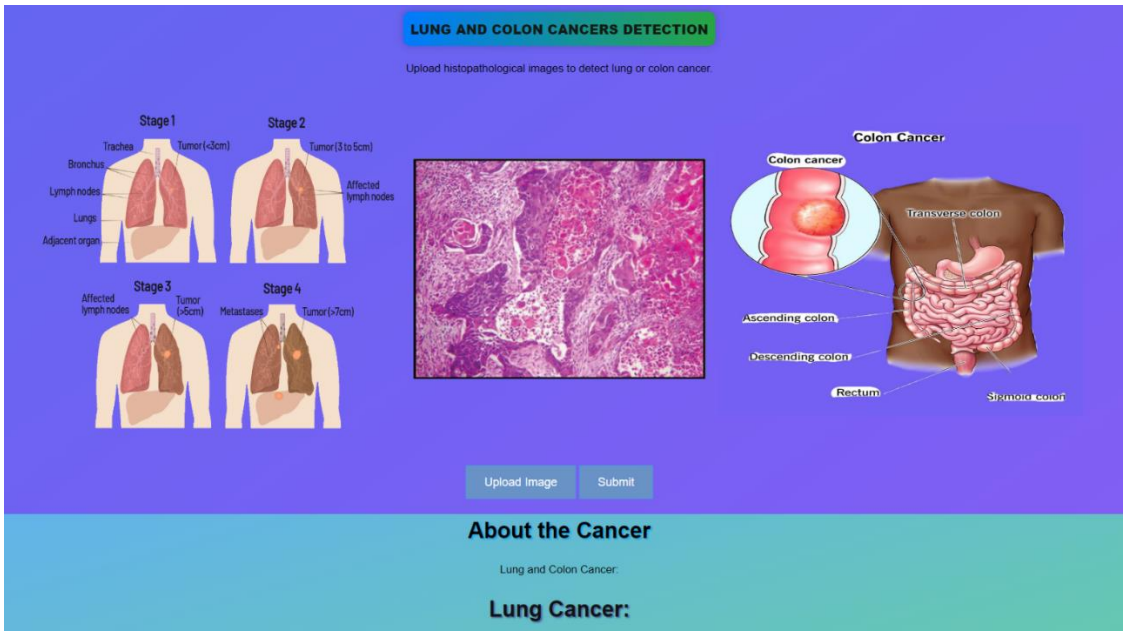

Fig 7.2.A: Interface

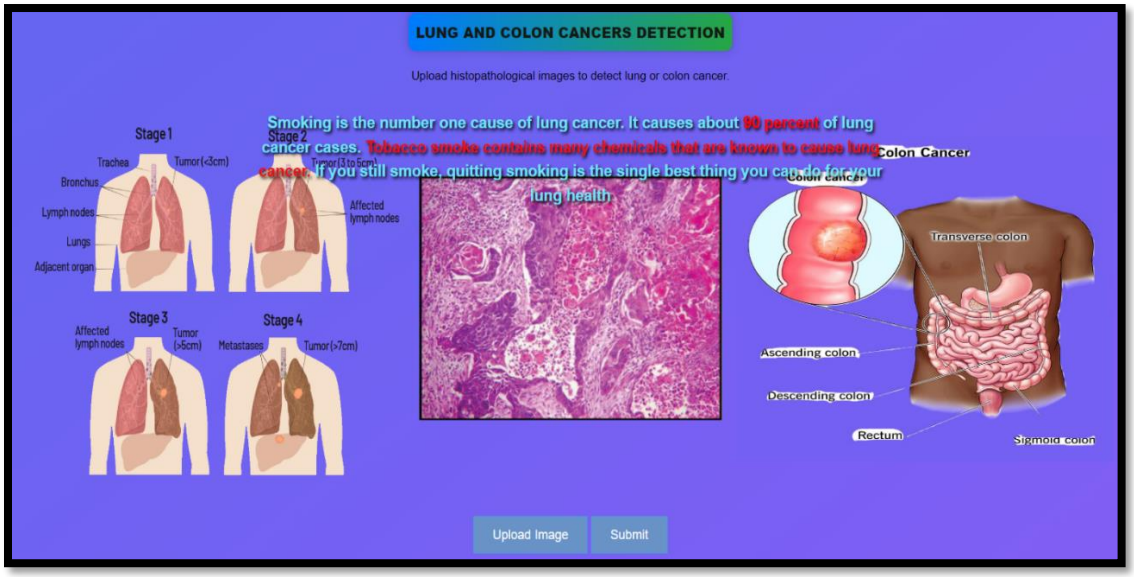

Fig 7.2.B: Cursor at left image

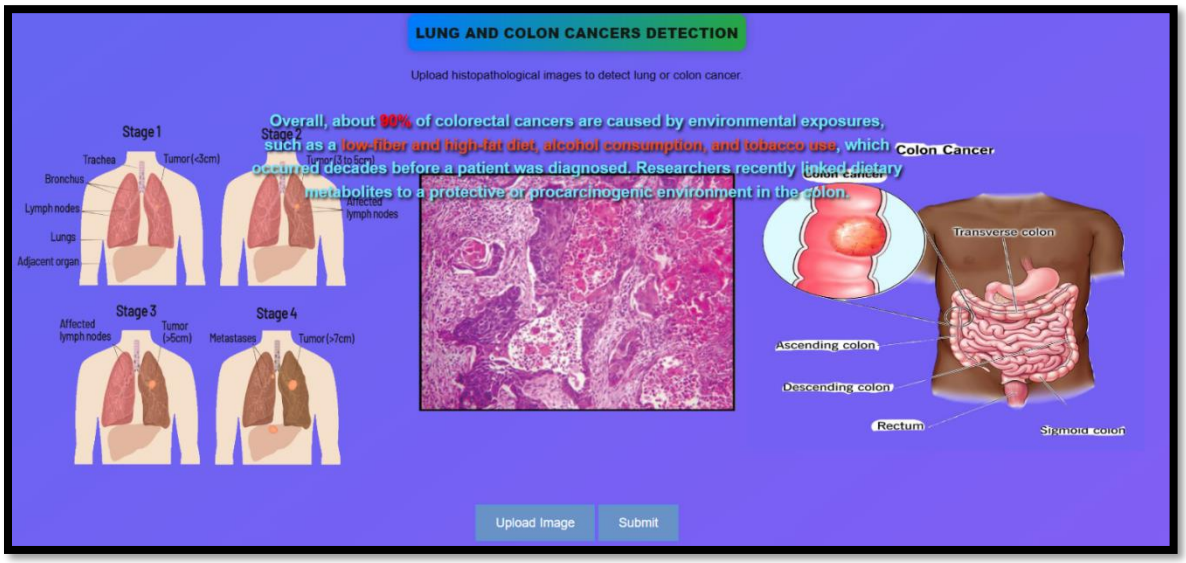

Fig 7.2.C: Cursor at Right image

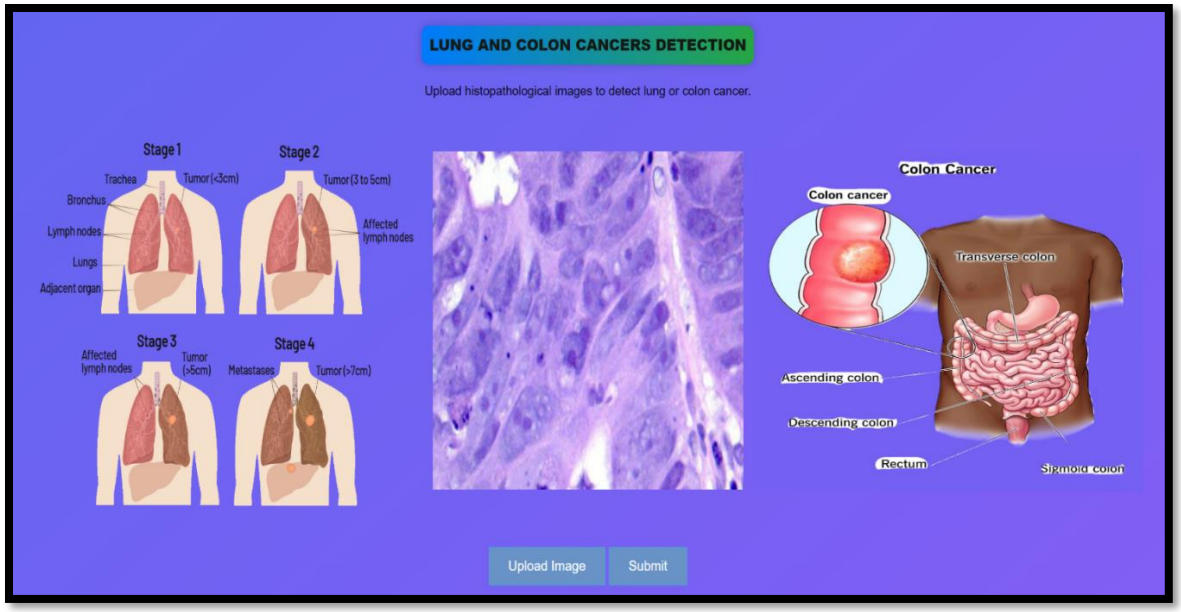

Fig 7.2.D: Image Uploded

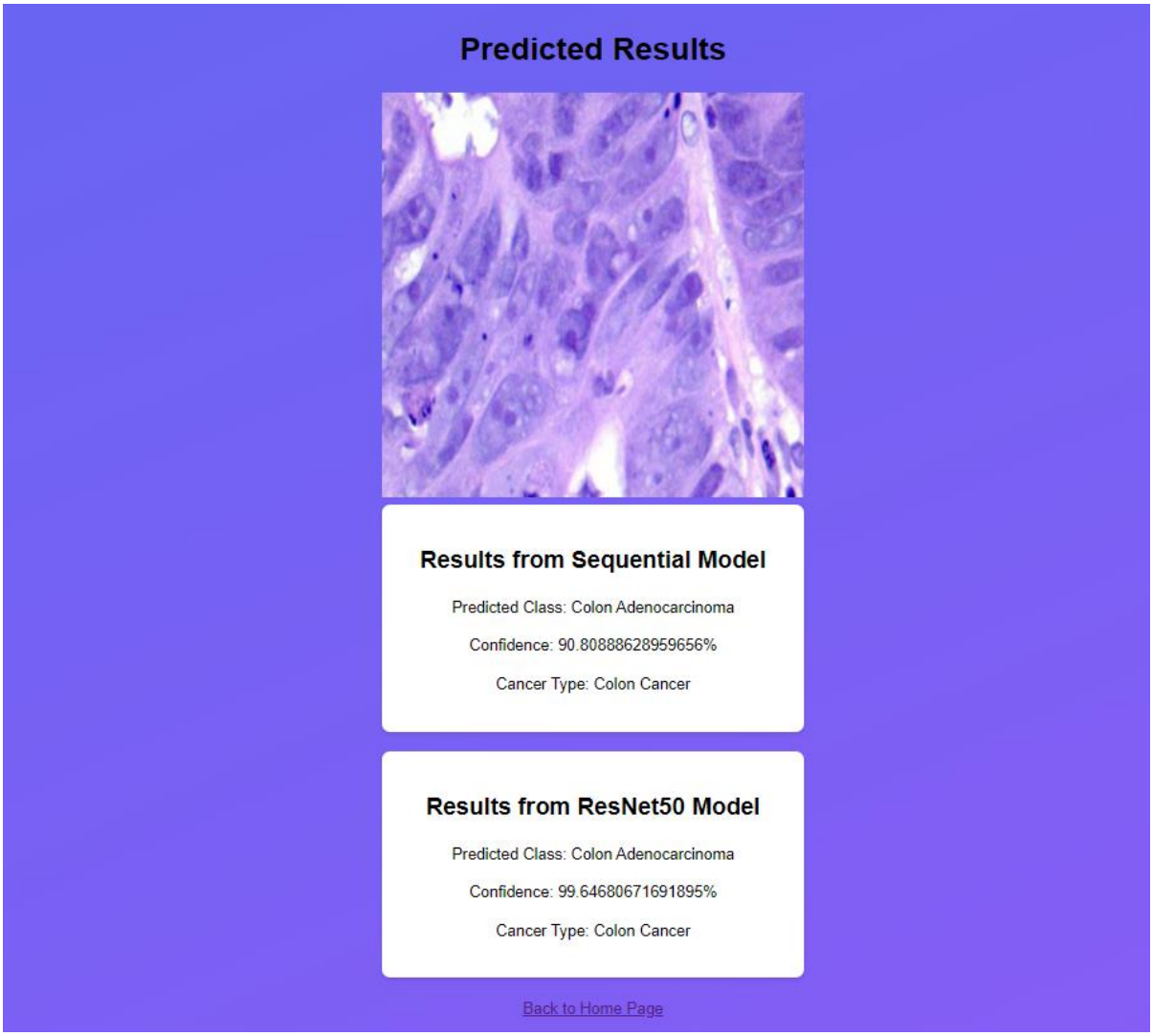

Fig 7.2.E: Predicted Results

CHAPTER 8

SYSTEM TESTING

SYSTEM TESTING

8.1 UNIT TESTING:

Unit testing involves testing individual components or units of the software in isolation to ensure they function correctly according to specifications. In your project, unit testing would involve testing the functionality of key modules such as the uploding images, loading model and classifing. This ensures that each component performs as expected and meets its requirements. Unit tests are typically automated and focus on validating input-output behavior, error handling, and boundary conditions within each unit.

8.2 INTEGRATION TESTING:

Integration testing involves testing the interactions and interfaces between different components or modules of the software to ensure they work together seamlessly. In your project, integration testing would focus on testing the integration of the Frontend and Backend with other system components. This ensures that data is exchanged correctly between modules and that communication between components functions as intended. Integration tests verify the correctness of system interfaces, data flow, and interoperability, helping to identify and resolve integration issues early in the development process.

CHAPTER 9

CONCLUSION

CONCLUSION

In conclusion, the use of Convolutional Neural Networks (CNNs) for Lung and Colon Cancer Detection has proven to deliver superior results in diagnosing these types of cancer. By using deep CNN models like the **Sequential model**, which **achieved 98% accuracy in training**, and the **ResNet50 model**, which **achieved 97% accuracy** ,the system enhances the accuracy and efficiency of classifying histopathological images compared to conventional machine learning models. This significant advancement in medical imaging has the potential to drastically reduce the time required for diagnosing cancer, providing more reliable performance than existing approaches and enabling doctors to treat patients more effectively and promptly. Ultimately, this innovative approach represents a major step forward in the early detection and treatment of lung and colon cancers, leading to better patient outcomes and increased survival rates.

REFERENCES

[1] Nishio M., Nishio M., Jimbo N., Nakane K. Homology-Based Image Processing for Automatic Classification of Histopathological Images of Lung Tissue. *Cancers.* 2021;13:1192. Doi: 10.3390/cancers13061192

[2] Mangal S., Chaurasia A., Khajanchi A. Convolution neural networks for diagnosing colon and lung cancer histopathological images. *arXiv.* 20202009.03878

[3] Mehmood S., Ghazal T.M., Khan M.A., Zubair M., Naseem M.T., Faiz T., Ahmad M. Malignancy detection in lung and colon histopathology images using transfer learning with class selective image processing.

[3] Tharsanee, R.M., Soundariya, R.s., Kumar, A.S., Karthiga, M., & Sountharrajan, S. (2021). Deep Convolutional neural network-based image classification for COVID-19 diagnosis. In Data Science for COVID-19 (pp. 117-145). Academic Press.

[4] Sarwinda D., Paradisa R.H., Bustamam A., Anggia P. Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer. *Procedia Comput. Sci.* 2021;179:423–431.

[5] Shim W.S., Yim K., Kim T.-J., Sung Y.E., Lee G., Hong J.H., Chun S.H., Kim S., An H.J., Na S.J., et al. DeepRePath: Identifying the Prognostic Features of Early-Stage Lung Adenocarcinoma Using Multi-Scale Pathology Images and Deep Convolutional Neural Network.