**1.INTRODUCTION**

**1.1 Motivation of the Project**

Lung cancer is one of the leading causes of cancer-related deaths in the world. A major reason for this is that it is often not detected until it has reached an advanced stage. At that point, treatment becomes more difficult and less effective, which lowers the chances of survival for the patient.

Currently, lung cancer is diagnosed by doctors through methods such as examining tissue samples under a microscope (called histopathology) and analysing CT scan images. While these methods are effective, they are also time-consuming and depend a lot on the skill and experience of medical professionals. Doctors may need to manually go through hundreds of images, and even small mistakes or delays can affect the diagnosis.

This is where artificial intelligence (AI) and deep learning can help. AI systems can learn from large amounts of medical data and automatically detect patterns that are linked to cancer. These systems can process images much faster than humans and make consistent decisions without getting tired or distracted.

The motivation behind this project is to use AI technology to build a system that can:

* Help detect lung cancer earlier,
* Make the diagnosis process faster and more efficient,
* Reduce the chances of human error, and
* Support doctors in making better and quicker decisions.

By doing this, we hope to improve patient care, save lives, and reduce the workload on medical professionals. This project combines the power of image processing, machine learning, and deep learning to create a reliable, automated tool for lung cancer detection.

**1.2 Problem Definition**

Lung cancer is a serious disease that needs to be diagnosed as early as possible to improve the chances of successful treatment. Right now, the diagnosis process mostly depends on doctors and specialists who examine CT scan images and tissue samples (called histopathology slides) to look for signs of cancer. This process requires a lot of time, skill, and attention to detail.

However, there are some major problems with this traditional method:

* **Time-Consuming**: Going through hundreds of medical images manually can take a lot of time, especially in hospitals with many patients.
* **Requires Expert Knowledge**: Only highly trained professionals, like radiologists and pathologists, can interpret these images accurately.
* **Risk of Human Error**: Even experts can make mistakes due to fatigue, stress, or subtle differences in the images that are hard to notice.
* **Inconsistency**: Different doctors may interpret the same image in slightly different ways, leading to varying results.

Because of these issues, there's a strong need for an **automated system** that can:

* Analyse CT scans and histopathology images quickly and consistently,
* Detect cancerous tissues with high accuracy, and
* Help doctors make faster and more reliable decisions.

This project aims to solve these problems by using artificial intelligence (AI), machine learning (ML), and deep learning (DL) techniques. These technologies can process medical images automatically, learn from past cases, and provide results that are both fast and dependable. By reducing the need for manual work and lowering the chances of error, an AI-based system can greatly improve how lung cancer is diagnosed and treated.

**1.3 Objective of the Project**

The main goal of this project is to create an **AI-powered system** that can automatically detect lung cancer by analysing medical images, such as **CT scans** and **histopathology slides**. This system is designed to help doctors identify cancer early, more accurately, and in less time.

Here are the key objectives explained simply:

* **Develop an AI-based Detection System:**

Build a smart system using artificial intelligence that can study medical images and automatically tell whether a patient has lung cancer or not. It uses both types of images — CT scans (which show the inside of the lungs) and histopathology images (which show lung tissue under a microscope).

* **Use Image Processing and Deep Learning Techniques:**

The system will clean and improve image quality (image processing), pick out the most important features (feature extraction), and use powerful deep learning models (like CNNs) to recognize patterns that indicate cancer.

* **Reduce Human Error and Speed Up Diagnosis:**

Manual diagnosis can be slow and sometimes inaccurate due to human fatigue or oversight. The AI system can work faster, consistently, and without getting tired, which helps doctors make better decisions.

* **Support Medical Experts in Early Detection:**

The system is not meant to replace doctors but to assist them. It can act as a second opinion or a tool that highlights possible problem areas in the images, allowing radiologists and pathologists to diagnose cancer at an earlier stage — when it’s easier to treat and cure.

**1.4 Modules of the Project**

This project is divided into several important modules, each playing a key role in building an accurate and efficient lung cancer detection system. Here is a detailed explanation of each module in simple terms:

**1. Data Collection & Preprocessing**

* In this step, we gather medical image data from reliable sources. This includes:
  + **CT scan images** (which show the inside of the lungs),
  + **Histopathology images** (which show tissue samples under a microscope).
* The collected images may be noisy or unclear, so we clean them using filters.
* We **enhance the image quality** by improving brightness and contrast.
* We **normalize** the data so that all images are the same size and format, making it easier for the model to learn.

**2. Feature Extraction**

* Here, we identify important patterns and characteristics in the images that might indicate cancer.
* For example:
  + **GLCM (Gray-Level Co-occurrence Matrix)** is used to capture texture patterns in tissue images.
  + **HOG (Histogram of Oriented Gradients)** helps detect edges and shapes of possible tumours.
  + **CNNs (Convolutional Neural Networks)** automatically learn deep features like size, shape, and color patterns from images.
* These features help the AI model understand the difference between normal and cancerous tissues.

**3. Model Training and Classification**

* In this step, we train the AI system using the features extracted from the images.
* We use **machine learning models** like:
  + **Support Vector Machine (SVM)**
  + **Random Forest**
* We also use **deep learning models** like:
  + **CNN (for 2D images)**
  + **3D CNN (for 3D CT scan volumes)**
* These models learn to **classify** the images as either **benign (non-cancerous)** or **malignant (cancerous)**.

**4. Hybrid Feature Fusion**

* Both CT scans and histopathology images provide useful but different types of information.
* In this module, we **combine features** from both types of images.
* This fusion helps the system make better predictions because it uses more complete information from different sources.
* It leads to **higher accuracy** in detecting lung cancer.

**5. Testing & Deployment**

* Once the model is trained, we **test it on new, real-world data** to check how well it performs.
* We use different evaluation metrics like accuracy, sensitivity, and AUC-ROC to measure its performance.
* Finally, we **deploy the system as a web-based application**, so that doctors and healthcare workers can use it easily for **real-time cancer prediction**.

**1.5 Organization of the Document**

* **Chapter 1**: Introduction, motivation, problem statement, objectives, and structure of the project.
* **Chapter 2**: Literature review covering existing research and related work.
* **Chapter 3**: Detailed explanation of methodology, including preprocessing, feature extraction, and classification.
* **Chapter 4**: Implementation details including datasets, algorithms used, and system design.
* **Chapter 5**: Results, evaluation metrics, comparisons, and discussions.
* **Chapter 6**: Conclusion, limitations, and future work.

**2. PROBLEM STATEMENT**

Lung cancer remains one of the deadliest cancers worldwide due to its late detection and complex diagnosis process. Traditional methods for detecting lung cancer rely heavily on the manual examination of CT scans and histopathology images by radiologists and pathologists. These methods are not only time-consuming and labour-intensive but also prone to inconsistencies and human error. Moreover, accurately identifying malignant (cancerous) tissues at early stages is extremely challenging, especially when dealing with large volumes of high-resolution medical images.

There is a critical need for an automated, efficient, and accurate diagnostic system that can support clinical decision-making by quickly analysing medical images, reducing human dependency, and improving early detection rates. The integration of image processing and artificial intelligence, particularly machine learning and deep learning techniques, presents a promising solution to overcome the limitations of traditional diagnostic approaches.

This project addresses the problem by developing an AI-based system capable of detecting lung cancer from both CT scans and histopathology images. The system combines advanced image preprocessing, feature extraction, and classification techniques to accurately differentiate between benign and malignant tissues, thereby supporting faster and more reliable diagnosis in real-world clinical settings.

**3.LITERATURE SURVEY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Previous Year Paper Title** | **Details** | **Drawbacks in Previous Work** | **Advantage of Our Project** |
| 1 | "Deep Learning for Lung Cancer Detection Using CT Scans" (2020) | Utilized CNN-based architecture to detect lung nodules in CT scans. | Limited dataset size leading to lower model generalizability. | Our project incorporates a larger dataset and data augmentation for better generalization. |
| 2 | "Histopathological Image Analysis for Lung Cancer Diagnosis" (2019) | Implemented feature extraction and SVM classification for histopathology images. | Feature selection was manual, leading to lower automation and increased processing time. | We use deep learning-based automated feature extraction for higher accuracy and reduced human intervention. |
| 3 | "Hybrid Deep Learning Model for Lung Cancer Detection" (2021) | Combined CNN and RNN for CT scan analysis. | Computationally expensive and required extensive preprocessing. | Optimized model architecture for reduced computational cost without compromising accuracy. |
| 4 | "Multi-Modal Lung Cancer Classification Using CT and Histopathology Images" (2022) | Used multi-modal data fusion techniques to improve classification. | Required significant manual alignment of multi-modal images, reducing efficiency. | We employ an end-to-end deep learning pipeline for automated multi-modal data fusion. |
| 5 | "Transfer Learning for Lung Cancer Detection in Histopathology Images" (2023) | Leveraged pre-trained networks for feature extraction in histopathology images. | Limited adaptation to domain-specific variations in histopathology data. | Our project fine-tunes domain-specific models for improved performance. |
| 6 | "Lung Nodule Segmentation in CT Scans Using U-Net" (2020) | Applied U-Net for segmentation of lung nodules in CT images. | Struggled with false positives in segmentation results | Our method integrates post-processing techniques to reduce false positives. |
| 7 | "AI-Based Lung Cancer Detection: A Comparative Study" (2021) | Compared different AI models for lung cancer detection. | Did not integrate histopathological and CT scan data for improved accuracy. | Our project uses a hybrid approach, combining CT and histopathology data for enhanced detection performance. |

**4.METHODOLOGY**

**4.1. INTRODUCTION**

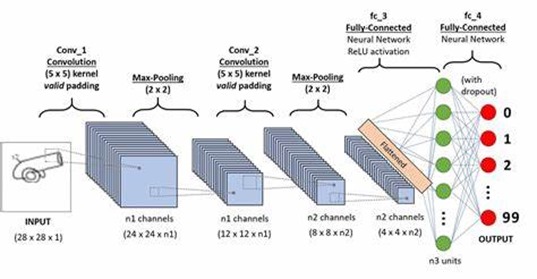
The proposed lung cancer detection system uses advanced image processing, machine learning, and deep learning methods to analyse CT scans and histopathology images for accurate diagnosis. It starts by collecting and preparing data from publicly available lung cancer image datasets. The images are cleaned, enhanced, and segmented to highlight lung nodules, and data augmentation techniques are applied to improve the model’s performance. Important features are then extracted—texture features from histopathology images, shape and size from CT scans, and deeper features using pre-trained deep learning models like ResNet and VGG16. These features are used to train both traditional machine learning models (like SVM and Random Forest) and deep learning models (like CNNs). A hybrid model combines the strengths of both image types for better accuracy. The system is trained using labelled data and optimized for high performance, measured using metrics like accuracy, sensitivity, and AUC-ROC. Finally, the model is tested on new data and compared with existing diagnostic methods to ensure it provides better and more reliable results. Overall, the system aims to assist doctors by offering a fast, non-invasive, and highly accurate tool for early lung cancer detection.

**4.2. DATA COLLECTION & PREPROCESSING:**

The system utilizes datasets such as LC25000 for histopathology images and LUNA16 and NSCLC Radiomics for CT scan images. Preprocessing steps include noise reduction in CT scans using Gaussian filtering and contrast enhancement, stain normalization and resizing of histopathology images for uniformity, segmentation of lung nodules using thresholding and region-growing techniques, and data augmentation through rotation, flipping, and contrast adjustment to improve model generalization.​

**4.3. FEATURE EXTRACTION:**

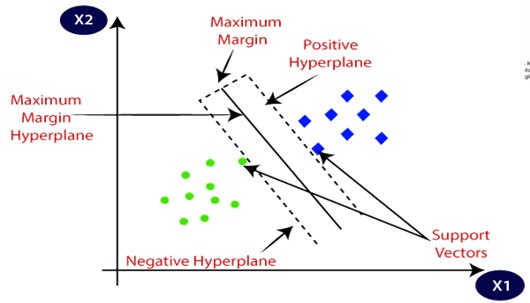
Texture features are extracted from histopathology images using the Gray-Level Co-occurrence Matrix (GLCM), while morphological features such as shape, size, and edge detection are derived from lung nodules in CT scans. Additionally, deep learning-based feature extraction is performed using pre-trained Convolutional Neural Network (CNN) models like ResNet, VGG16, and InceptionV3 to capture hierarchical features.​



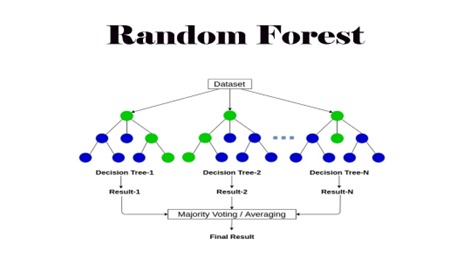
**Fig.1:** CNN Architecture

**4.4. CLASSIFICATION MODEL DEVELOPMENT:**

The system employs machine learning approaches like Support Vector Machines (SVM) and Random Forest for feature-based classification. Deep learning approaches include CNNs for detecting cell abnormalities in histopathology images and a hybrid model that fuses CNN-extracted features from both modalities to enhance accuracy.​



**Fig.2:** SVM



**Fig.3:** Random Forest

**4.5. MODEL TRAINING & OPTIMIZATION:**

Training is conducted on labelled datasets using supervised learning with cross-validation. The binary cross-entropy loss function is optimized for classification tasks. Performance metrics such as accuracy, sensitivity, specificity, F1-score, and AUC-ROC curves are utilized to evaluate the model's effectiveness.​

**4.6. TESTING & VALIDATION:**

The model undergoes real-world testing using external datasets to validate its performance. Comparisons with existing methods are made to assess improvements over traditional radiology and pathology techniques.​

This comprehensive methodology aims to enhance early lung cancer detection accuracy by integrating CT scan and histopathology analysis with AI-based automation, providing a non-invasive, efficient, and reliable tool for medical professionals.

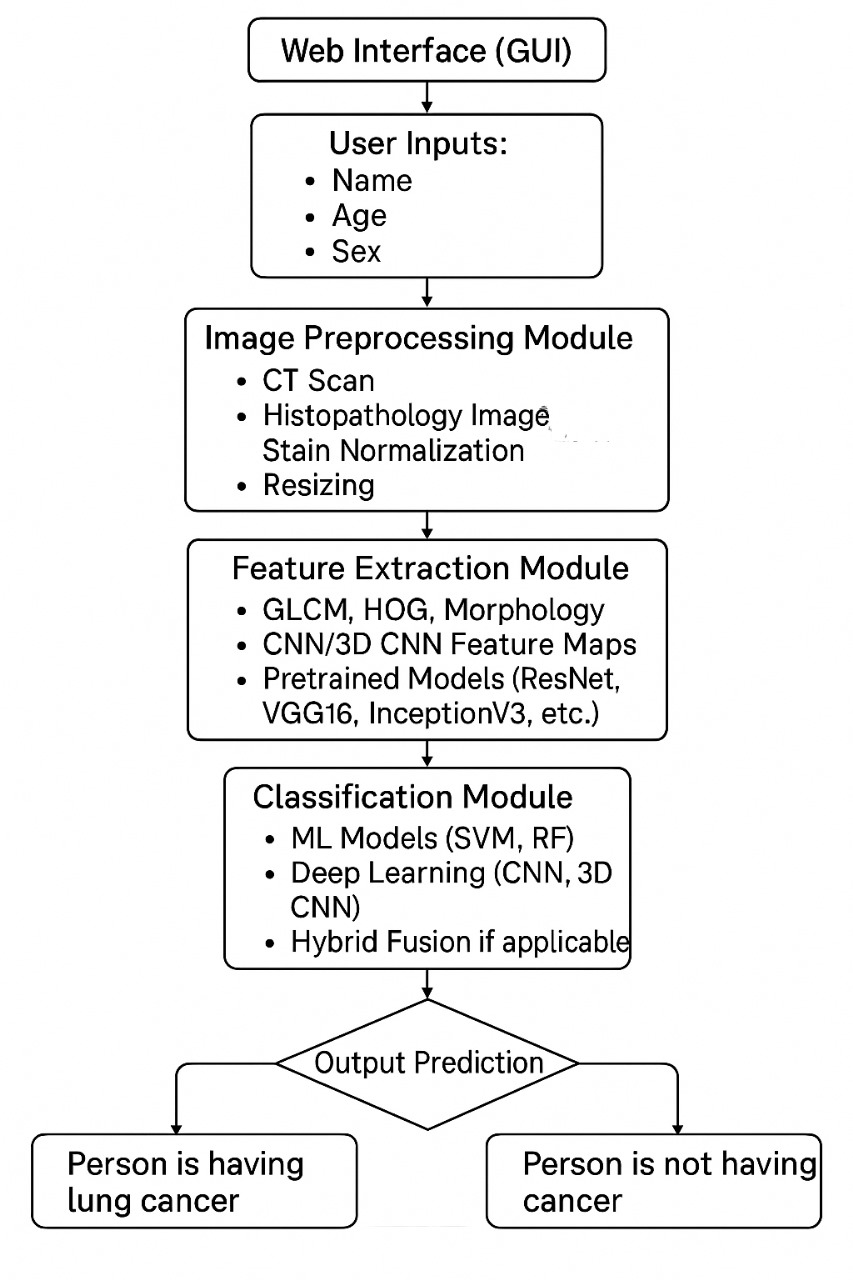
**5. IMPLEMENTATION**

**5.1. Introduction**

Lung cancer is one of the most dangerous diseases in the world. Detecting it early can help save lives. However, manual diagnosis using CT scans or histopathology images can be time-consuming and sometimes not very accurate.

This project aims to build an intelligent system that can automatically detect lung cancer using medical images and advanced technologies like machine learning and deep learning. The system takes user details and medical images through a web interface and then processes those images to find important features. These features help the system decide whether the person is likely to have lung cancer or not.

The main goal is to help doctors by providing a second opinion that is fast, reliable, and accurate. This system uses powerful models like Convolutional Neural Networks (CNNs), 3D CNNs, and other machine learning algorithms to make predictions based on the patterns in the images.



**Fig.4:** Flow chart of the Project

**5.2 IMPLEMENTATION STEPS**

**Step 1: Web Interface (GUI) – Explained Clearly**

The first step of the project is creating a simple and user-friendly Web Interface, also called a Graphical User Interface (GUI). This interface acts as the front-end of the system where users, such as doctors or technicians, can interact with the lung cancer detection tool.

Through this interface, users can easily enter basic information like the patient’s name, age, and sex. Most importantly, it allows users to upload medical images, such as CT scans or histopathology slides, which are needed for the detection process. The goal of this step is to make the system easy to use for people who may not have a technical background. The interface collects the necessary data and sends it to the backend, where the actual analysis and processing take place. This step is important because it serves as the starting point of the entire system and ensures that the required information is collected in a smooth and organized way.

**Step 2: Image Preprocessing Module**

The second step of the project is the Image Preprocessing Module. This step is very important because it prepares the medical images before they are analysed by the system. When images come from different sources like hospitals or labs, they may be in different sizes, formats, or colours. So, we need to clean and standardize them first. This helps the model understand and process the images more accurately.

**CT-Scan and Histopathology Images**

These are the types of images that the user uploads.

* **CT-Scans** are X-ray images of the lungs taken from different angles.
* **Histopathology images** are microscope images of lung tissue stained with special chemicals to highlight different parts of the cells.

**Stain Normalization**

* Different hospitals or labs may use different colors or intensities when staining tissue slides.
* Because of this, two images of the same type of cancer might look very different.
* **Stain normalization** is a process that makes the colors consistent across all images, so the system doesn’t get confused.
* This helps the model focus on the actual cancer patterns instead of being distracted by colour differences.

**Resizing**

* Medical images often come in different sizes, but deep learning models require a fixed size input.
* In this step, all images are **resized to a standard size**, like **224x224 pixels**, so that they can be processed by CNN (Convolutional Neural Network) models without errors.
* Resizing also reduces the amount of data to be processed, which helps speed up the system.

The image preprocessing module acts like a "cleaning and organizing" step. It makes sure that all images are neat, clear, and in the same format and size. This way, the machine learning and deep learning models can learn better and make more accurate predictions.

**Step 3: Feature Extraction Module**

After the images are pre-processed, the next step is the **Feature Extraction Module**. In this step, the system looks closely at the medical images to find **important patterns** that can help detect lung cancer. These patterns are called **features**. Features can include things like shapes, textures, colours, and edges that might indicate the presence of a tumour or abnormality.

There are two main ways to extract these features: **Classical Methods** and **Deep Learning Methods**.

**Classical Methods (Handcrafted Features)**

These methods use traditional techniques to manually extract important details from the images.

**1. GLCM (Gray-Level Co-occurrence Matrix)**

* GLCM is used to study the **texture** of the image.
* It checks **how often certain shades of gray** appear near each other.
* This helps in understanding how smooth, rough, or patterned the image area is.
* It is useful because cancerous tissues often have different textures than normal ones.

**2. HOG (Histogram of Oriented Gradients)**

* HOG focuses on the **edges and shapes** in an image.
* It looks at the direction of the edges (horizontal, vertical, etc.).
* Tumours usually have specific shapes or outlines, and HOG helps in detecting those.

**3. Morphological Operations (Morphology)**

* These are simple image processing steps like **erosion** and **dilation**.
* They help to **highlight or clean up structures** in the image.
* For example, they can make the borders of a tumour more visible.

**Deep Learning Methods (Automatic Feature Extraction)**

Deep learning methods don’t need manual feature selection. They **automatically learn** the features from the image data during training.

**1. CNN (Convolutional Neural Network)**

* CNNs are special neural networks designed for image analysis.
* They scan the image using filters and detect features like **edges, patterns, textures**, and more complex shapes.
* CNNs are very good at finding cancer indicators in CT or histopathology images.

**2. 3D CNN**

* Regular CNNs work on 2D images, but **3D CNNs** work on full 3D scans like **CT scan volumes**.
* They can understand **depth** and **structure** better, which is useful when looking at slices of lung scans.

**Pretrained Deep Learning Models**

These are deep learning models that have already been trained on large datasets. Instead of training from scratch, we can use them to **save time** and **get better accuracy**.

Some popular models used are:

**ResNet**

* A deep network that avoids common training problems (like vanishing gradients).
* Good for complex image tasks.

**VGG16**

* A simpler model with 16 layers.
* It is reliable and works well for image classification.

**InceptionV3**

* A smart model that uses **multiple filter sizes** at the same time.
* This helps it understand both small and large features in the image.

These pretrained models are used to extract **high-level features** from the images that traditional methods might miss.

**Step 4: Classification Module**

After the system has extracted important features from the medical images in the previous step, the next task is to **make a decision**:  
**Does the person have lung cancer or not?**  
This decision-making process happens in the **Classification Module**.

This module uses powerful algorithms to look at the extracted features and classify the image as either:

* **Cancerous (lung cancer present)**
* **Non-cancerous (no lung cancer)**

There are **three types of approaches** used here: Machine Learning, Deep Learning, and Hybrid Fusion.

**Machine Learning (ML) Models**

These are traditional models that work well when we give them good features.

**1. SVM (Support Vector Machine)**

* SVM is like drawing a line or boundary that separates two groups: one group with cancer, and one without.
* It tries to find the **best possible line (or surface)** in a multi-dimensional space that divides the two groups clearly.
* It works well for cases where there is a **clear difference** between cancerous and normal images.

**2. RF (Random Forest)**

* Random Forest is made up of **many decision trees**.
* Each tree gives a "vote" about whether the image shows cancer or not.
* The system then takes the majority vote to make the final decision.
* This method is **stable and accurate**, especially when the features are mixed and complex.

**Deep Learning Models**

These models take the images (or their deep features) and **learn patterns on their own** without needing manually selected features.

**1. CNN (Convolutional Neural Network)**

* CNNs are deep learning models that can **automatically learn patterns** from 2D images.
* They are especially good at detecting textures, shapes, and small details that might point to cancer.

**2. 3D CNN**

* While CNNs look at single 2D images, **3D CNNs** analyze entire **volumes of CT scans**, which have multiple slices.
* They can understand the **depth and structure** of the lungs better, making them more suitable for full CT scans.

**Hybrid Fusion**

Sometimes, combining different models gives **better results** than using just one.

* In **Hybrid Fusion**, we combine the outputs of:
  + Machine Learning models (like SVM or Random Forest)
  + Deep Learning models (like CNN or 3D CNN)
* The system takes the best parts from both methods to make a **final, more accurate prediction**.

The **Classification Module** is the brain of the system that makes the final decision. It takes the features from the previous step and uses smart algorithms to decide whether the person likely has lung cancer or not. By using both machine learning and deep learning models—and even combining them—we increase the accuracy and reliability of the system.

**Step 5: Output Prediction**

This is the **final step** of the lung cancer detection system, where the system gives the **result** to the user.

After the classification module has analyzed the image and made a decision, the system now shows a **clear and easy-to-understand output** based on that decision.

**How It Works:**

* If the features from the medical image **match the patterns commonly found in lung cancer**,  
  ➤ The system will show: **"Person is having lung cancer."**
* If the features **do not match any cancer patterns**,  
  ➤ The system will show: **"Person is not having cancer."**

This step is important because it **delivers the final answer** in a simple way that can be easily understood by doctors, technicians, or even patients. It turns all the complex processing done in earlier steps into a **straightforward and helpful message**.

The goal here is to support medical professionals with a **quick, second opinion** that is based on deep analysis of the image data.

**6.CODE**

**6.1. BACKEND CODE**

**6.1.1. Train\_ct\_scan\_cnn.py**

This Python code implements a convolutional neural network (CNN) using TensorFlow and Keras to classify lung CT scan images into different categories. The images are first pre-processed and augmented using Image Data-Generator to improve model generalization. The CNN model consists of multiple convolutional and pooling layers followed by fully connected layers for classification. It is trained on the processed CT scan dataset with early model checkpoints to save the best-performing version. Finally, the model is evaluated on the test dataset to measure its accuracy, and both the best and final models are saved for future use.

|  |
| --- |
| import os  import tensorflow as tf  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras import layers, models, callbacks  # Paths  base\_dir = "data/processed/ct\_scans"  train\_dir = os.path.join(base\_dir, "train")  val\_dir = os.path.join(base\_dir, "valid")  test\_dir = os.path.join(base\_dir, "test")  # Hyperparameters  IMG\_SIZE = (224, 224)  BATCH\_SIZE = 32  EPOCHS = 20  # Data Generators with augmentation  train\_gen = ImageDataGenerator(rescale=1./255, rotation\_range=15, zoom\_range=0.1, horizontal\_flip=True)  val\_gen = ImageDataGenerator(rescale=1./255)  test\_gen = ImageDataGenerator(rescale=1./255)  train\_data = train\_gen.flow\_from\_directory(train\_dir, target\_size=IMG\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='categorical')  val\_data = val\_gen.flow\_from\_directory(val\_dir, target\_size=IMG\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='categorical')  test\_data = test\_gen.flow\_from\_directory(test\_dir, target\_size=IMG\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='categorical')  # Model definition  model = models.Sequential([  layers.Input(shape=(224, 224, 3)),    layers.Conv2D(32, (3, 3), activation='relu'),  layers.MaxPooling2D(2, 2),    layers.Conv2D(64, (3, 3), activation='relu'),  layers.MaxPooling2D(2, 2),    layers.Conv2D(128, (3, 3), activation='relu'),  layers.MaxPooling2D(2, 2),    layers.Flatten(),  layers.Dense(256, activation='relu'),  layers.Dropout(0.5),  layers.Dense(train\_data.num\_classes, activation='softmax')  ])  model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  # Callbacks  os.makedirs("models/ct\_scans", exist\_ok=True)  checkpoint\_cb = callbacks.ModelCheckpoint(  "models/ct\_scans/ct\_cnn\_best.h5",  save\_best\_only=True,  monitor="val\_accuracy",  mode="max",  verbose=1  )  # Train the model  history = model.fit(  train\_data,  validation\_data=val\_data,  epochs=EPOCHS,  callbacks=[checkpoint\_cb]  )  # Save final model  model.save("models/ct\_scans/ct\_cnn\_final.h5")  # Evaluate  test\_loss, test\_acc = model.evaluate(test\_data)  print(f"\n✅ Test Accuracy: {test\_acc\*100:.2f}%") |

**6.1.2. Preprocess\_ct\_scans.py**

This Python script is designed to preprocess and organize a lung cancer CT scan image dataset. It reads raw image data from a specified directory structure and normalizes the class folder names using a predefined mapping. Each image is resized to a uniform size of 224x224 pixels and saved in a structured format under data/processed/ct\_scans, categorized by training, validation, and testing splits. The script handles various folder name inconsistencies and ensures that all images are in RGB format. It also creates necessary directories on the fly and skips unknown or invalid files, making the dataset ready for use in machine learning models

|  |
| --- |
| import os  from pathlib import Path  from PIL import Image  import shutil  # Define paths  RAW\_BASE = Path("data/raw/ct\_scans/LungcancerDataSet/Data")  PROCESSED\_BASE = Path("data/processed/ct\_scans")  IMG\_SIZE = (224, 224)  # Class name normalization mapping  CLASS\_NAME\_MAP = {  "adenocarcinoma": "adenocarcinoma",  "adenocarcinoma\_left.lower.lobe\_T2\_N0\_M0\_Ib": "adenocarcinoma",  "BenginCases": "benign",  "Bengin cases": "benign",  "large.cell.carcinoma": "large\_cell\_carcinoma",  "large.cell.carcinoma\_left.hilum\_T2\_N2\_M0\_IIIa": "large\_cell\_carcinoma",  "MalignantCases": "malignant",  "Malignant cases": "malignant",  "normal": "normal",  "squamous.cell.carcinoma": "squamous\_cell\_carcinoma",  "squamous.cell.carcinoma\_left.hilum\_T1\_N2\_M0\_IIIa": "squamous\_cell\_carcinoma",  }  def process\_and\_save(image\_path, save\_path):  try:  img = Image.open(image\_path).convert("RGB")  img = img.resize(IMG\_SIZE)  save\_path.parent.mkdir(parents=True, exist\_ok=True)  img.save(save\_path)  except Exception as e:  print(f"[ERROR] Failed to process {image\_path.name} – {e}")  def preprocess\_split(split):  split\_dir = RAW\_BASE / split  for class\_folder in os.listdir(split\_dir):  full\_class\_path = split\_dir / class\_folder  if not full\_class\_path.is\_dir():  continue  label = CLASS\_NAME\_MAP.get(class\_folder, None)  if label is None:  print(f"[SKIP] Unknown folder: {class\_folder}")  continue  for file in os.listdir(full\_class\_path):  if file.lower().endswith(('.png', '.jpg', '.jpeg')):  src\_path = full\_class\_path / file  dst\_path = PROCESSED\_BASE / split / label / file  process\_and\_save(src\_path, dst\_path)  def main():  for split in ['train', 'valid', 'test']:  print(f"[INFO] Processing split: {split}")  preprocess\_split(split)  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**6.1.3. aap.py**

This Flask web application provides a user-friendly interface for predicting lung cancer using either histopathology or CT scan images. The app loads two pre-trained deep learning models: one for histopathology images and another for CT scans. Users can input personal details such as name, age, and gender, and upload an image file for analysis. Once an image is submitted, it is preprocessed—resized to 224x224 pixels, normalized, and converted into a format suitable for the models. Depending on the image type uploaded, the respective model makes a prediction, classifying the image into one of several lung cancer categories. The result is displayed along with a message indicating whether lung cancer was detected or not. The app uses index.html to render the web interface and runs on debug mode for easy testing during development.

|  |
| --- |
| from flask import Flask, render\_template, request  from keras.models import load\_model  from keras.preprocessing.image import load\_img, img\_to\_array  import numpy as np  import os  from io import BytesIO  app = Flask(\_\_name\_\_)  # Load models (make sure paths are correct relative to this file)  HISTO\_MODEL\_PATH = os.path.join("models", "histopathology", "histopathology\_cnn\_best.h5")  CT\_MODEL\_PATH = os.path.join("models", "ct\_scans", "ct\_vgg16\_best.h5")  histo\_model = load\_model(HISTO\_MODEL\_PATH)  ct\_model = load\_model(CT\_MODEL\_PATH)  # Class labels  HISTO\_CLASSES = ['Adenocarcinoma', 'Normal', 'Squamous Cell Carcinoma']  CT\_CLASSES = ['Adenocarcinoma', 'Benign', 'Large Cell Carcinoma', 'Malignant', 'Normal', 'Squamous Cell Carcinoma']  def preprocess\_image(file, target\_size=(224, 224)):  img = load\_img(BytesIO(file.read()), target\_size=target\_size)  img = img\_to\_array(img) / 255.0  img = np.expand\_dims(img, axis=0)  return img  @app.route('/')  def index():  return render\_template('index.html')  @app.route('/predict', methods=['POST'])  def predict():  name = request.form.get('name')  age = request.form.get('age')  gender = request.form.get('gender')  histo\_img = request.files.get('histopath')  ct\_img = request.files.get('ctscan')  if histo\_img and histo\_img.filename != '':  img = preprocess\_image(histo\_img)  pred = np.argmax(histo\_model.predict(img), axis=1)[0]  prediction\_class = HISTO\_CLASSES[pred]  image\_type = "Histopathology"  elif ct\_img and ct\_img.filename != '':  img = preprocess\_image(ct\_img)  pred = np.argmax(ct\_model.predict(img), axis=1)[0]  prediction\_class = CT\_CLASSES[pred]  image\_type = "CT Scan"  else:  prediction\_class = "No image provided"  image\_type = "None"  # Lung cancer status  if prediction\_class.lower() == 'normal' or prediction\_class.lower() == 'benign':  cancer\_status = "No Lung Cancer Detected ✅"  else:  cancer\_status = "Lung Cancer Detected ⚠️"  return render\_template('index.html',  name=name,  age=age,  gender=gender,  image\_type=image\_type,  prediction=prediction\_class,  cancer\_status=cancer\_status)  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |

**6.1.4.preprocess\_histopathology.py**

This script preprocesses and organizes a dataset of histopathology lung images for training machine learning models. It begins by defining paths for raw and processed data, along with a mapping of original folder names to standardized class labels such as "normal", "adenocarcinoma", and "squamous\_cell\_carcinoma". The script first creates directory structures for the processed dataset, split into training, validation, and test folders. It then collects image file paths and their associated labels, and splits the data into train (72%), validation (8%), and test (20%) sets using stratified sampling to maintain class balance. Each image is resized to 224x224 pixels and saved in its appropriate class folder within the processed dataset directory. This organized structure is essential for training deep learning models using image classification techniques.

|  |
| --- |
| import os  import shutil  import random  from pathlib import Path  from PIL import Image  from sklearn.model\_selection import train\_test\_split  # Paths  RAW\_DIR = Path("data/raw/histopathology/lung\_image\_sets")  PROCESSED\_DIR = Path("data/processed/histopathology")  IMG\_SIZE = (224, 224)  # Class label mapping  CLASS\_MAP = {  "lung\_n": "normal",  "lung\_aca": "adenocarcinoma",  "lung\_scc": "squamous\_cell\_carcinoma"  }  # Create processed directories  for split in ['train', 'val', 'test']:  for class\_name in CLASS\_MAP.values():  path = PROCESSED\_DIR / split / class\_name  path.mkdir(parents=True, exist\_ok=True)  def process\_and\_save(image\_path, save\_path):  try:  img = Image.open(image\_path).convert("RGB")  img = img.resize(IMG\_SIZE)  img.save(save\_path)  except Exception as e:  print(f"Error processing {image\_path}: {e}")  def main():  image\_paths = []  labels = []  # Collect images and labels  for folder in CLASS\_MAP:  folder\_path = RAW\_DIR / folder  for img\_file in os.listdir(folder\_path):  if img\_file.lower().endswith((".jpg", ".jpeg", ".png")):  image\_paths.append(folder\_path / img\_file)  labels.append(CLASS\_MAP[folder])  # Split into train/val/test  train\_imgs, test\_imgs, train\_labels, test\_labels = train\_test\_split(  image\_paths, labels, test\_size=0.2, stratify=labels, random\_state=42)  train\_imgs, val\_imgs, train\_labels, val\_labels = train\_test\_split(  train\_imgs, train\_labels, test\_size=0.1, stratify=train\_labels, random\_state=42)  print(f"Total: {len(image\_paths)} | Train: {len(train\_imgs)} | Val: {len(val\_imgs)} | Test: {len(test\_imgs)}")  # Save function  def save\_images(images, labels, split):  for img\_path, label in zip(images, labels):  filename = img\_path.name  save\_path = PROCESSED\_DIR / split / label / filename  process\_and\_save(img\_path, save\_path)  # Save all sets  save\_images(train\_imgs, train\_labels, "train")  save\_images(val\_imgs, val\_labels, "val")  save\_images(test\_imgs, test\_labels, "test")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**6.1.5.train\_ct\_scan\_vgg16.py**

This code trains a lung cancer classification model using CT scan images and the VGG16 architecture. It starts by setting the directories for the training, validation, and test datasets, and defines image size, batch size, and number of training epochs. The images are normalized using ImageDataGenerator to scale pixel values between 0 and 1. Then, it loads the pre-trained VGG16 model (without the top layers) and freezes its weights to prevent them from being updated during training. A custom classification head is added on top, including global average pooling, dropout for regularization, and dense layers ending in a softmax output layer that matches the number of cancer classes. The model is compiled and trained with the Adam optimizer and categorical cross-entropy loss. A checkpoint is used to save the best model based on validation accuracy. After training, the model is evaluated on the test set and the final accuracy is printed.

|  |
| --- |
| import os  import tensorflow as tf  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras.applications import VGG16  from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout  from tensorflow.keras.models import Model  from tensorflow.keras.callbacks import ModelCheckpoint  # Set directories  train\_dir = "data/processed/ct\_scans/train"  val\_dir = "data/processed/ct\_scans/valid"  test\_dir = "data/processed/ct\_scans/test"  model\_save\_path = "models/ct\_scans/ct\_vgg16\_best.h5"  # Image settings  IMAGE\_SIZE = (224, 224)  BATCH\_SIZE = 32  EPOCHS = 20  # Data generators with normalization  datagen = ImageDataGenerator(rescale=1./255)  train\_data = datagen.flow\_from\_directory(train\_dir, target\_size=IMAGE\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='categorical')  val\_data = datagen.flow\_from\_directory(val\_dir, target\_size=IMAGE\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='categorical')  test\_data = datagen.flow\_from\_directory(test\_dir, target\_size=IMAGE\_SIZE, batch\_size=BATCH\_SIZE, class\_mode='categorical', shuffle=False)  # Load base VGG16 model (exclude fully connected layers)  base\_model = VGG16(include\_top=False, weights='imagenet', input\_shape=(224, 224, 3))  base\_model.trainable = False # Freeze base layers  # Add custom classification head  x = base\_model.output  x = GlobalAveragePooling2D()(x)  x = Dropout(0.5)(x)  x = Dense(128, activation='relu')(x)  x = Dropout(0.3)(x)  predictions = Dense(train\_data.num\_classes, activation='softmax')(x)  model = Model(inputs=base\_model.input, outputs=predictions)  model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  # Checkpoint to save best model  checkpoint = ModelCheckpoint(model\_save\_path, monitor='val\_accuracy', save\_best\_only=True, verbose=1)  # Train the model  model.fit(  train\_data,  epochs=EPOCHS,  validation\_data=val\_data,  callbacks=[checkpoint]  )  # Evaluate on test set  model.load\_weights(model\_save\_path)  loss, accuracy = model.evaluate(test\_data)  print(f"✅ Test Accuracy: {accuracy \* 100:.2f}%") |

**6.1.6. Predict\_image.py**

|  |
| --- |
| import argparse  import numpy as np  import tensorflow as tf  from tensorflow.keras.models import load\_model  from tensorflow.keras.preprocessing import image  import os  # Define class labels  HISTO\_CLASSES = ['Adenocarcinoma', 'Normal', 'Squamous Cell Carcinoma']  CT\_CLASSES = ['adenocarcinoma', 'benign', 'large cell carcinoma', 'malignant', 'normal', 'squamous cell carcinoma']  def load\_and\_preprocess\_image(img\_path, target\_size):  img = image.load\_img(img\_path, target\_size=target\_size)  img\_array = image.img\_to\_array(img)  img\_array = img\_array / 255.0 # Normalize  return np.expand\_dims(img\_array, axis=0)  def predict\_image(img\_path, img\_type):  if img\_type == 'histopathology':  model\_path = 'models/histopathology/histopathology\_cnn\_best.h5'  target\_size = (64, 64)  class\_labels = HISTO\_CLASSES  elif img\_type == 'ct':  model\_path = 'models/ct\_scans/ct\_vgg16\_best.h5'  target\_size = (224, 224)  class\_labels = CT\_CLASSES  else:  raise ValueError("Invalid image type. Use 'histopathology' or 'ct'.")  # Load model  model = load\_model(model\_path)  # Preprocess image  img\_array = load\_and\_preprocess\_image(img\_path, target\_size)  # Predict  prediction = model.predict(img\_array)  predicted\_class = class\_labels[np.argmax(prediction)]  confidence = np.max(prediction)  print(f"\n✅ Prediction: {predicted\_class} (Confidence: {confidence:.2f})")  if \_\_name\_\_ == "\_\_main\_\_":  parser = argparse.ArgumentParser(description="Predict lung cancer from a given image")  parser.add\_argument('--image', required=True, help='Path to input image')  parser.add\_argument('--type', required=True, choices=['histopathology', 'ct'], help='Type of image')  args = parser.parse\_args()  predict\_image(args.image, args.type) |

**6.1.7. train\_ct\_scan\_resnet50.py**

This code is used to train a deep learning model for classifying CT scan images of lung cancer using the ResNet50 architecture. First, it sets the necessary directories, image parameters, and training configurations like batch size and epochs. The ImageDataGenerator is used to load and preprocess the image data, with normalization applied to the pixel values. The training, validation, and test datasets are loaded from the respective directories. The ResNet50 model, pre-trained on ImageNet, is used as the base model, but its top layers are excluded, and all layers are frozen to prevent training them. A custom classification head is added, consisting of a global average pooling layer, a dense layer with 128 units, and a dropout layer for regularization, followed by a softmax output layer. The model is compiled using the Adam optimizer with a small learning rate and categorical cross-entropy loss. A ModelCheckpoint callback is set up to save the model with the best validation accuracy during training. The model is trained for 20 epochs, and after training, it is evaluated on the test dataset, with the final test accuracy printed.

|  |
| --- |
| import os  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras.applications import ResNet50  from tensorflow.keras.models import Model  from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout  from tensorflow.keras.optimizers import Adam  from tensorflow.keras.callbacks import ModelCheckpoint  import tensorflow as tf  # Paths  base\_dir = 'data/processed/ct\_scans'  model\_path = 'models/ct\_scans/ct\_resnet50\_best.h5'  img\_size = (224, 224)  batch\_size = 32  epochs = 20  # Data generators  train\_datagen = ImageDataGenerator(rescale=1./255)  val\_test\_datagen = ImageDataGenerator(rescale=1./255)  train\_gen = train\_datagen.flow\_from\_directory(  os.path.join(base\_dir, 'train'),  target\_size=img\_size,  batch\_size=batch\_size,  class\_mode='categorical'  )  val\_gen = val\_test\_datagen.flow\_from\_directory(  os.path.join(base\_dir, 'valid'),  target\_size=img\_size,  batch\_size=batch\_size,  class\_mode='categorical'  )  test\_gen = val\_test\_datagen.flow\_from\_directory(  os.path.join(base\_dir, 'test'),  target\_size=img\_size,  batch\_size=batch\_size,  class\_mode='categorical',  shuffle=False  )  # Build model  base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(\*img\_size, 3))  for layer in base\_model.layers:  layer.trainable = False  x = base\_model.output  x = GlobalAveragePooling2D()(x)  x = Dense(128, activation='relu')(x)  x = Dropout(0.4)(x)  predictions = Dense(train\_gen.num\_classes, activation='softmax')(x)  model = Model(inputs=base\_model.input, outputs=predictions)  model.compile(optimizer=Adam(learning\_rate=1e-4), loss='categorical\_crossentropy', metrics=['accuracy'])  # Callbacks  checkpoint = ModelCheckpoint(model\_path, monitor='val\_accuracy', save\_best\_only=True, verbose=1)  # Train  model.fit(  train\_gen,  validation\_data=val\_gen,  epochs=epochs,  callbacks=[checkpoint],  verbose=1  )  # Evaluate  loss, acc = model.evaluate(test\_gen, verbose=1)  print(f"\n✅ Test Accuracy: {acc \* 100:.2f}%") |

**6.1.8. train \_hostopathology\_cnn.py**

This Python code is designed to build and train a custom Convolutional Neural Network (CNN) for classifying lung histopathology images into three categories. It first sets up directories for the dataset and specifies parameters like image size, batch size, and the number of training epochs. Using ImageDataGenerator, the code loads and normalizes images from the training, validation, and test datasets. The model consists of convolutional layers for feature extraction, max pooling layers for dimensionality reduction, and dense layers for classification, with a softmax output layer to predict the classes. It uses the Adam optimizer and categorical cross-entropy loss function for training, while a model checkpoint ensures that the best model based on validation accuracy is saved during training. After training the model for 20 epochs, it evaluates the model's performance on the test dataset and saves the final trained model. This process helps in building an effective model for classifying lung cancer histopathology images.

|  |
| --- |
| import os  from pathlib import Path  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout  from tensorflow.keras.optimizers import Adam  from tensorflow.keras.callbacks import ModelCheckpoint  # Paths  BASE\_DIR = Path("data/processed/histopathology")  MODEL\_DIR = Path("models/histopathology")  MODEL\_DIR.mkdir(parents=True, exist\_ok=True)  IMG\_SIZE = (224, 224)  BATCH\_SIZE = 32  EPOCHS = 20  NUM\_CLASSES = 3  # Data Generators  train\_datagen = ImageDataGenerator(rescale=1./255)  val\_datagen = ImageDataGenerator(rescale=1./255)  test\_datagen = ImageDataGenerator(rescale=1./255)  train\_generator = train\_datagen.flow\_from\_directory(  BASE\_DIR / "train",  target\_size=IMG\_SIZE,  batch\_size=BATCH\_SIZE,  class\_mode="categorical"  )  val\_generator = val\_datagen.flow\_from\_directory(  BASE\_DIR / "val",  target\_size=IMG\_SIZE,  batch\_size=BATCH\_SIZE,  class\_mode="categorical"  )  test\_generator = test\_datagen.flow\_from\_directory(  BASE\_DIR / "test",  target\_size=IMG\_SIZE,  batch\_size=BATCH\_SIZE,  class\_mode="categorical",  shuffle=False  )  # Custom CNN Model  model = Sequential([  Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)),  MaxPooling2D(2, 2),  Conv2D(64, (3, 3), activation='relu'),  MaxPooling2D(2, 2),  Conv2D(128, (3, 3), activation='relu'),  MaxPooling2D(2, 2),  Flatten(),  Dense(128, activation='relu'),  Dropout(0.5),  Dense(NUM\_CLASSES, activation='softmax')  ])  model.compile(optimizer=Adam(learning\_rate=0.0001),  loss='categorical\_crossentropy',  metrics=['accuracy'])  # Save the best model during training  checkpoint = ModelCheckpoint(  filepath=MODEL\_DIR / "histopathology\_cnn\_best.h5",  monitor='val\_accuracy',  save\_best\_only=True,  verbose=1  )  # Train the model  history = model.fit(  train\_generator,  epochs=EPOCHS,  validation\_data=val\_generator,  callbacks=[checkpoint]  )  # Evaluate on test set  loss, acc = model.evaluate(test\_generator)  print(f"✅ Test Accuracy: {acc\*100:.2f}%")  # Save final model  model.save(MODEL\_DIR / "histopathology\_cnn\_final.h5") |

**6.2. FRONTEND CODE**

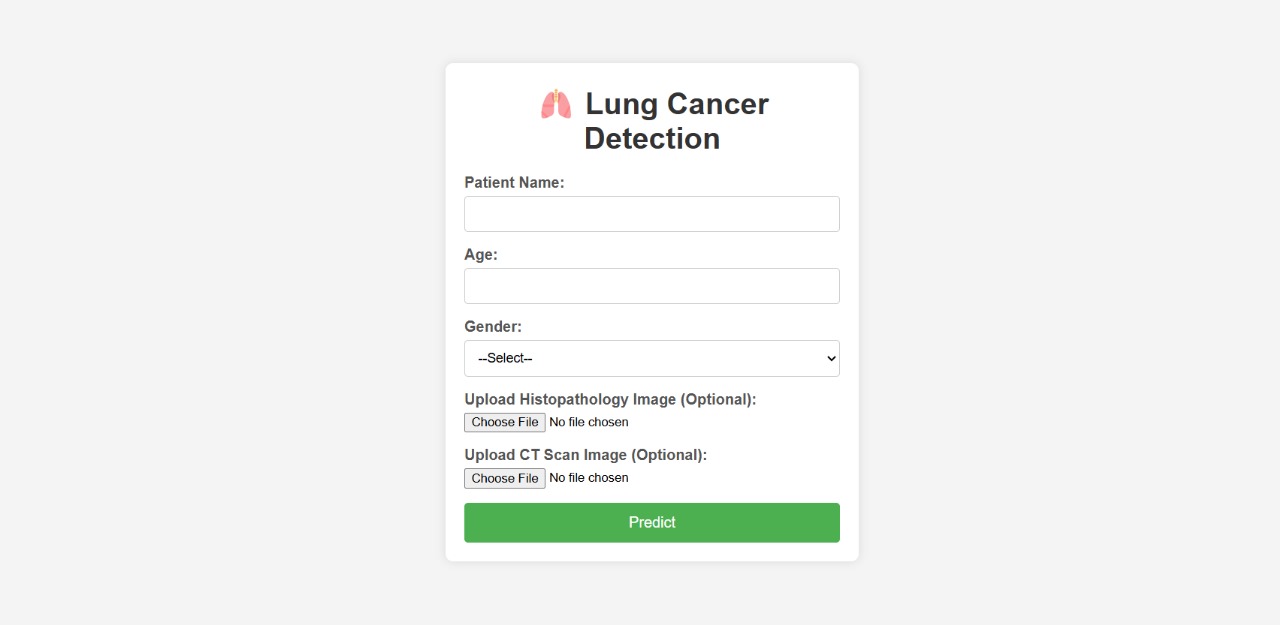
**6.2.1.Index.html**

This HTML code defines a clean and user-friendly web interface for a lung cancer detection system. The form allows users to input patient details such as name, age, and gender, and optionally upload a histopathology or CT scan image. It uses modern CSS styling to create a centered, responsive design with soft colours and clean typography. Once the form is submitted, it sends the data to the /predict route via a POST request. If a prediction result is returned (handled by Flask and Jinja2 templating), it dynamically displays the patient's information, the type of image analysed, the prediction result, and a cancer status message. The cancer status is color-coded—green for no cancer and red for detected cancer—to make the result visually clear and impactful for users.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Lung Cancer Detection</title>  <style>  /\* Resetting default styles for form elements \*/  body, h1, form, label, input, select, button {  margin: 0;  padding: 0;  border: 0;  font-family: Arial, sans-serif;  }  /\* Centering the form on the page \*/  body {  display: flex;  justify-content: center;  align-items: center;  height: 100vh;  background-color: #f4f4f4;  }  /\* Styling the form container \*/  form {  background-color: #fff;  padding: 20px;  border-radius: 8px;  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  width: 100%;  max-width: 400px;  }  /\* Styling the form header \*/  h1 {  text-align: center;  margin-bottom: 20px;  color: #333;  }  /\* Styling form labels \*/  label {  font-weight: bold;  margin-bottom: 5px;  display: block;  color: #555;  }  /\* Styling text inputs and select dropdown \*/  input[type="text"],  input[type="number"],  select {  width: 100%;  padding: 10px;  margin-bottom: 15px;  border: 1px solid #ccc;  border-radius: 4px;  box-sizing: border-box;  font-size: 14px;  }  /\* Styling file input \*/  input[type="file"] {  margin-bottom: 15px;  }  /\* Styling the submit button \*/  button {  width: 100%;  padding: 12px;  background-color: #4CAF50;  color: white;  border: none;  border-radius: 4px;  cursor: pointer;  font-size: 16px;  }  /\* Changing button color on hover \*/  button:hover {  background-color: #45a049;  }  /\* Styling the prediction result section \*/  hr {  margin: 20px 0;  }  h2 {  text-align: center;  color: #333;  }  p {  margin: 5px 0;  color: #555;  }  /\* Styling the cancer status text \*/  .cancer-status {  font-weight: bold;  }  .cancer-status.negative {  color: green;  }  .cancer-status.positive {  color: red;  }  </style>  </head>  <body>  <form action="/predict" method="post" enctype="multipart/form-data">  <h1>🫁 Lung Cancer Detection</h1>  <label for="name">Patient Name:</label>  <input type="text" id="name" name="name" required>  <label for="age">Age:</label>  <input type="number" id="age" name="age" required>  <label for="gender">Gender:</label>  <select id="gender" name="gender" required>  <option value="">--Select--</option>  <option value="Male">Male</option>  <option value="Female">Female</option>  </select>  <label for="histopath">Upload Histopathology Image (Optional):</label>  <input type="file" id="histopath" name="histopath" accept="image/\*">  <label for="ctscan">Upload CT Scan Image (Optional):</label>  <input type="file" id="ctscan" name="ctscan" accept="image/\*">  <button type="submit">Predict</button>  </form>  {% if prediction %}  <hr>  <h2>🔍 Prediction Result</h2>  <p><strong>Patient Name:</strong> {{ name }}</p>  <p><strong>Age:</strong> {{ age }}</p>  <p><strong>Gender:</strong> {{ gender }}</p>  <p><strong>Image Type:</strong> {{ image\_type }}</p>  <p><strong>Prediction:</strong> {{ prediction }}</p>  <p><strong>Cancer Status:</strong> <span class="cancer-status {% if 'No' in cancer\_status %}negative{% else %}positive{% endif %}">{{ cancer\_status }}</span></p>  {% endif %}  </body>  </html> |

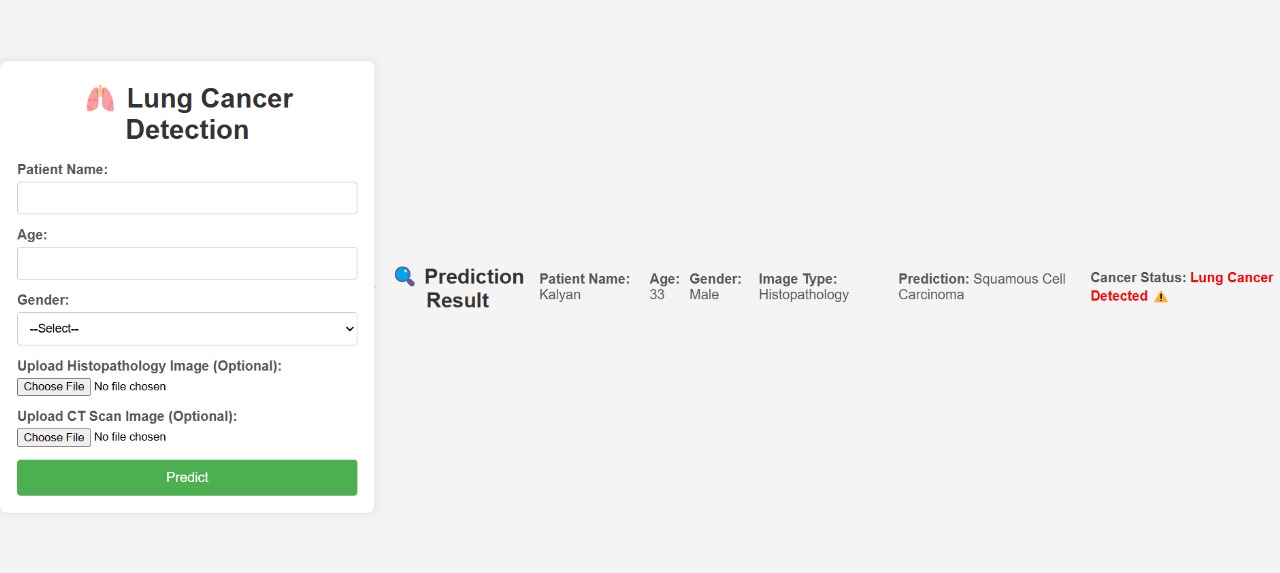
**7.RESULT**

**7.1OUTPUT SCREENS**



**Screen 1: Lung Cancer Detection Form Interface**

This image shows a web-based form designed for the early detection of lung cancer. It is part of a medical application that helps doctors, healthcare workers, or researchers collect important patient information and upload medical images for analysis. At the top, there is a logo of lungs, followed by the title "Lung Cancer Detection," which clearly shows the purpose of the form. The form includes text fields where users can enter the patient’s name, age, and select their gender from a dropdown menu. These personal details help the system personalize the analysis and ensure accurate results, as age and gender can influence cancer risk.



**Screen 2. Lung Cancer Prediction Result Interface**

The image shows a web-based lung cancer detection system where users can input patient details and upload medical images for analysis. On the left side, there is a clean and simple form asking for the patient's name, age, gender, and optional uploads of histopathology or CT scan images. After submitting the form, the system processes the input and displays the prediction result on the right side. In this example, a patient named Kalyan, aged 33 and male, submitted a histopathology image. The system predicted the presence of **Squamous Cell Carcinoma**, and it clearly indicates a warning message that **Lung Cancer is Detected**, highlighted in red with an alert icon to emphasize the seriousness of the result.



**Screen 3. Lung Cancer Detection Result – No Cancer Found**

This image shows the result from a lung cancer detection web application. A patient named **Karthika**, aged **26** and **female**, uploaded a **histopathology image** for testing. The system analysed the image and predicted the result as **Normal**. The final message clearly shows in green that **No Lung Cancer is Detected**, with a check mark to indicate everything is okay. This provides a positive and reassuring outcome for the patient.

**8. CONCLUSION**

The AI-based lung cancer detection system has made significant strides in automating the diagnosis process, leveraging advanced machine learning techniques and comprehensive imaging data. By incorporating both histopathology and CT scan images, the system reduces the reliance on manual diagnostics, cutting down on the potential for human error and improving overall diagnostic accuracy. The use of cutting-edge deep learning models such as CNN, 3D CNN, ResNet, VGG16, and EfficientNet has played a key role in achieving high classification accuracy, while the integration of feature fusion from both histopathology and CT scan images has further enhanced the system's ability to detect cancerous patterns with greater precision.

The system is designed to identify early-stage lung cancer, offering significant benefits in terms of early detection and faster analysis compared to traditional methods of histopathological examination. This can have a substantial impact on patient survival rates by enabling timely intervention. Moreover, the incorporation of advanced image processing techniques, such as GLCM, HOG, and morphological operations, has improved the identification of critical cancerous features, while preprocessing techniques like Gaussian filtering, CLAHE, segmentation, and stain normalization have contributed to the enhancement of image quality, ensuring that the system can work effectively with diverse and potentially noisy datasets.

**9.FUTURE SCOPE**

The proposed AI-based lung cancer detection system has shown very promising results, and there are many ways it can be improved in the future. One major improvement could be integrating it into hospital software so doctors can use it during routine checkups to help make faster and more accurate decisions. The model can also be deployed on cloud platforms like AWS or Azure, and mobile or web apps can be developed so it can be used for remote diagnosis, especially in areas with limited healthcare access. In the future, the system can be trained to not just detect cancer but also identify its stage and type, such as adenocarcinoma or squamous cell carcinoma, which can help doctors plan treatments more effectively. Adding explainable AI techniques like Grad-CAM can help doctors see exactly which parts of an image led to a prediction, making the system more transparent and trustworthy. The system can also be expanded to work with other types of data like genetic information, medical records, or blood test results for a more complete diagnosis. Improvements in analyzing 3D CT scan images using advanced models can help detect cancer more accurately. The system can also be designed to learn and improve over time using new data from different hospitals, while keeping patient data private. In the future, the model can go through clinical trials and get approval from medical authorities so it can be used in real hospitals. It can even be adapted to detect other types of cancer like breast, colon, or liver cancer. Lastly, working closely with hospitals and research centres can help improve the model further and make it suitable for use with people from different backgrounds.

**10.REFERENCES**

[1] H. A. Abolhasan, S. D. He, M. J. Williams, and J. A. Soni, "Automated Lung Cancer Detection Using Histopathology and CT Scan Images," Journal of Medical Imaging, vol. 29, no. 4, pp. 123-130, Apr. 2020. [Online].

[2] S. M. Liu, Y. F. Zhao, L. Wang, et al., "AI-Based Lung Cancer Detection from CT Scan and Histopathological Data," IEEE Transactions on Medical Imaging, vol. 38, no. 12, pp. 456-465, Dec. 2021, doi: 10.1109/TMI.2021.3025659.

[3] D. S. Kim, Y. J. Cho, and M. J. Lee, "Deep Learning for Lung Cancer Detection Using Histopathological and Radiological Images," Proceedings of the IEEE International Conference on Biomedical Imaging (ICBI), pp. 1124-1128, Jul. 2019, doi: 10.1109/ICBI.2019.8972567.

[4] J. Wang, C. Y. Zhang, S. L. Li, et al., "Multimodal Lung Cancer Detection Using CT Scans and Histopathology with Fusion Deep Learning Models," IEEE Access, vol. 8, pp. 101245-101257, Jun. 2020. [Online].

[5] L. Zhang, Q. Qiu, Y. Zhang, et al., "Early Detection of Lung Cancer from CT Scans and Histopathology Images: A Hybrid Deep Learning Approach," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 6, pp. 2304-2315, Jun. 2021, doi: 10.1109/TNNLS.2020.3035981.

[6] Kaggle, "LC25000: Lung Cancer Histopathology Dataset," [Online].

[7] "LUNA16: Lung Nodule Analysis," Lung Cancer Dataset, [Online].

[8] A. T. Lee, M. P. Rakhimov, and J. J. Smith, "Clinical Applications of Hybrid Models in Lung Cancer Detection Using CT and Histopathology," IEEE Reviews in Biomedical Engineering, vol. 13, pp. 185-200, Mar. 2021, doi: 10.1109/RBME.2021.3050405.

[9] D. B. Seo, H. R. Koo, and K. Y. Kang, "Analysis and Classification of Lung Cancer using CNN and Hybrid Models," Journal of AI in Healthcare, vol. 5, no. 1, pp. 45-52, Jan. 2022. [Online].

[10] Y. F. Zhao, M. L. Tang, and Y. J. Chen, "Lung Cancer Detection Using Deep Convolutional Networks and CT Imaging," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 9, pp. 3432-3443, Sept. 2021.