**LUNG NODULE DETECTION**



Prepared by:

Vecha Sai Badarinadh (21BCE7084)

Somisetty Shivani Lasya (21BCE8300)

Shaik Mohammed Sajidulla (21BCE7526)

Date: 01-03-2025

Under the guidance of:

Dr. R. Shalini

# 1. Introduction

Lung nodule detection plays a crucial role in medical imaging, particularly in the early diagnosis of lung cancer. Lung nodules are small abnormal growths in the lungs, which can be benign or malignant. Early detection is vital because lung cancer is often asymptomatic in its initial stages, making it difficult to diagnose until it has progressed significantly. Traditional diagnostic methods, such as manual analysis of CT scans, can be time-consuming and prone to human error. This challenge has led to the growing use of artificial intelligence (AI) and machine learning (ML) techniques, which can automate the detection process and enhance accuracy. By leveraging deep learning models, medical professionals can improve diagnostic efficiency, reduce false positives and negatives, and ensure patients receive timely medical attention.

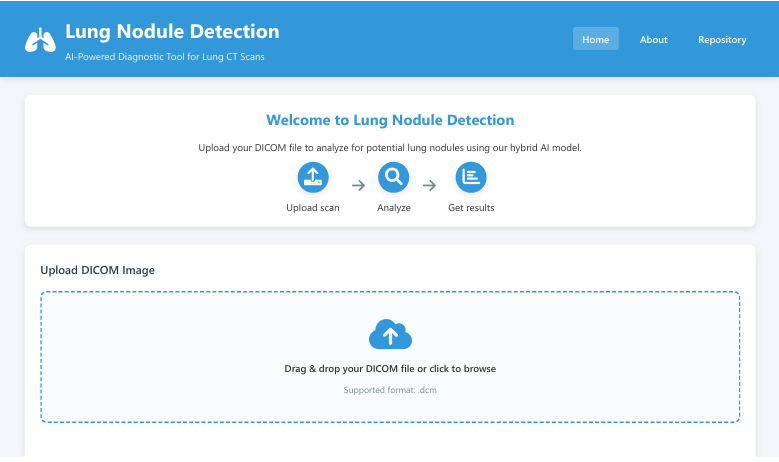
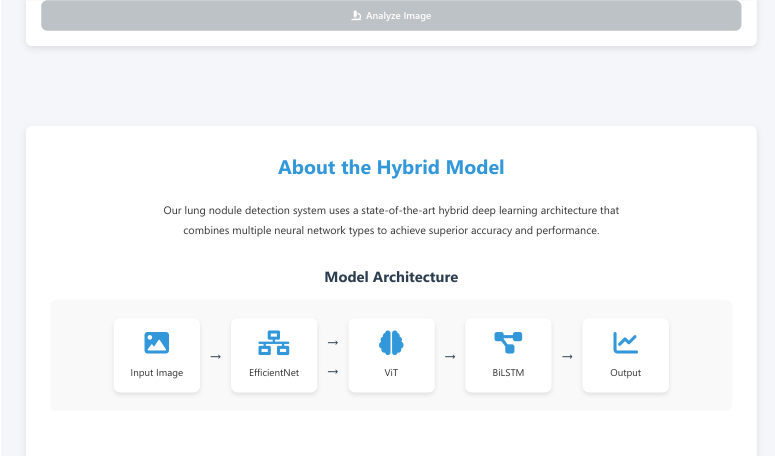
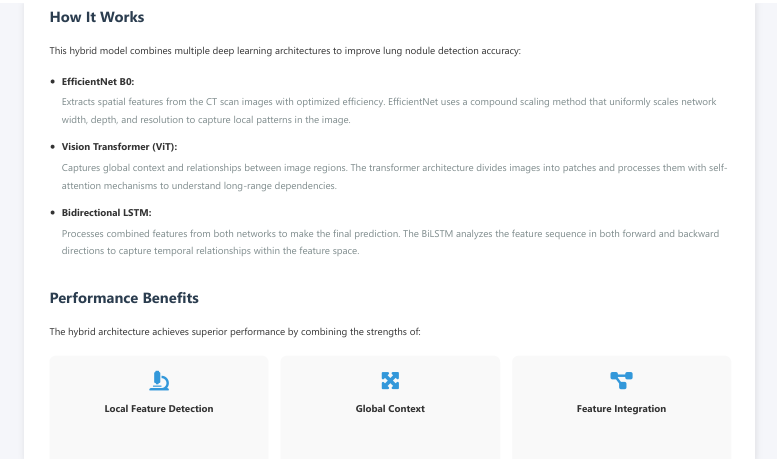
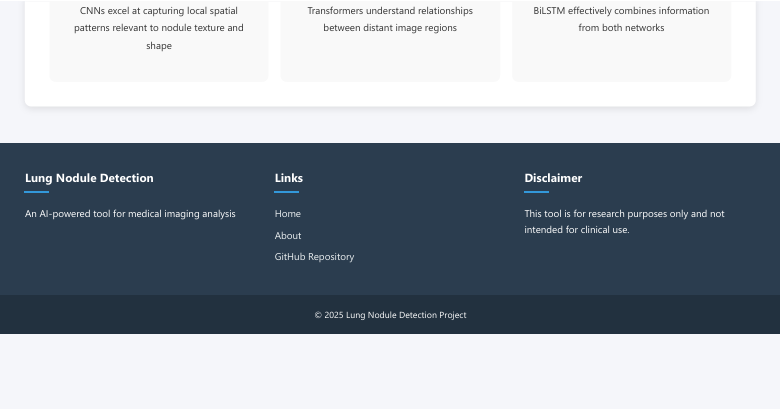
The primary objective of this project is to develop an efficient lung nodule detection system that provides quick and accurate results using a hybrid machine-learning approach. The system integrates deep learning-based image processing with traditional medical imaging techniques to enhance the precision of nodule detection. By allowing users to upload lung CT scans, the model processes and analyzes the images, determining the likelihood of nodule presence. The hybrid algorithm ensures a high confidence level in the detection process, making it a valuable tool for radiologists and healthcare professionals. With further advancements, this system has the potential to become an essential component in the early diagnosis and treatment planning for lung cancer patients.

# 2. Project Overview

This lung nodule detection system is designed to assist users in analyzing lung CT scans by providing an automated diagnosis of potential nodules. Users can upload their scans through a user-friendly interface, where the system processes the images using a hybrid model that integrates deep learning techniques with traditional image processing methods. By leveraging convolutional neural networks (CNNs) for feature extraction and machine learning classifiers for decision-making, the system enhances detection accuracy while minimizing false positives. The hybrid approach ensures a robust and reliable analysis, offering a high-confidence assessment of whether lung nodules are present. This not only aids in early detection but also supports medical professionals in making informed diagnostic decisions.

# 3. System Interface

The user-friendly web interface is designed to simplify the process of lung nodule detection for both medical professionals and general users. It allows users to easily upload lung CT scans and receive real-time diagnostic results, including a confidence score indicating the likelihood of nodule presence. The interface is intuitive, ensuring a seamless interaction with minimal effort required from the user. Clear instructions and an organized layout guide users through the process, from scan upload to result interpretation. The results are displayed in a structured format, highlighting whether nodules have been detected along with a visual representation of the confidence level. Additionally, the interface provides warnings and recommendations, encouraging users to seek professional medical consultation for further evaluation. This design ensures that the system is accessible to a wide range of users, enhancing its practicality and usability in real-world healthcare applications.

# 4. Code and Algorithm

The hybrid model utilized in this system combines the strengths of deep learning and traditional image processing techniques to enhance the accuracy and reliability of lung nodule detection. Deep learning, particularly convolutional neural networks (CNNs), is employed for feature extraction, allowing the system to recognize intricate patterns and anomalies within lung CT scans. This automated learning approach helps in distinguishing between normal lung tissue and potential nodules with high precision. Additionally, traditional image processing techniques such as edge detection, thresholding, and morphological operations are integrated to refine the detection process, reducing false positives and improving sensitivity. By leveraging both methodologies, the hybrid model ensures a balanced approach—deep learning provides intelligent pattern recognition, while traditional methods enhance structural analysis. This combination results in a highly accurate and efficient system that aids in early lung cancer detection, ultimately improving patient outcomes.

import os

import pandas as pd

import torch

import pydicom

import numpy as np

import albumentations as A

from torchvision import transforms

from torch.utils.data import Dataset, DataLoader

import timm

import torch.nn as nn

import torch.optim as optim

from PIL import Image

# -------------------------

# Path setup

# -------------------------

BASE\_DIR = "./dataset"

IMAGE\_DIR = os.path.join(BASE\_DIR, "images/images")

ANNOTATION\_DIR = os.path.join(BASE\_DIR, "annotations/annotations/tcia-lidc-xml")

CSV\_FILE = os.path.join(BASE\_DIR, "lidc\_metadata.csv")

# -------------------------

# Load metadata

# -------------------------

metadata = pd.read\_csv(CSV\_FILE)

metadata['findings'] = metadata['findings'].fillna('')  # Replace NaN with empty string

metadata['label'] = metadata['findings'].apply(lambda x: 1 if 'Nodules' in str(x) else 0)

label\_dict = dict(zip(metadata['image\_id'], metadata['label']))

# -------------------------

# Get all DICOM images

# -------------------------

image\_paths = [os.path.join(IMAGE\_DIR, fname) for fname in os.listdir(IMAGE\_DIR) if fname.endswith(".dcm")]

# -------------------------

# Custom Dataset Class

# -------------------------

class LIDCDataset(Dataset):

    def \_\_init\_\_(self, image\_paths, label\_dict, transform=None):

        self.image\_paths = image\_paths

        self.label\_dict = label\_dict

        self.transform = transform

    def \_\_len\_\_(self):

        return len(self.image\_paths)

    def \_\_getitem\_\_(self, index):

        img\_path = self.image\_paths[index]

        img\_id = os.path.basename(img\_path).replace(".dcm", "")

        # Load DICOM Image

        dicom\_image = pydicom.dcmread(img\_path)

        image = dicom\_image.pixel\_array  # Extract pixel data

        # Convert grayscale to RGB

        if len(image.shape) == 2:

            image = np.stack([image] \* 3, axis=-1)

        # Normalize to 0-255

        image = (image - np.min(image)) / (np.max(image) - np.min(image)) \* 255.0

        image = image.astype(np.uint8)

        # Get label (default: 0 if not found)

        label = self.label\_dict.get(img\_id, 0)

        # Apply transformations

        if self.transform:

            image = self.transform(image=image)["image"]

        # Convert to PyTorch tensor

        image = torch.tensor(image).permute(2, 0, 1)  # (H, W, C) → (C, H, W)

        label = torch.tensor(label, dtype=torch.float32)

        return image, label

# -------------------------

# Define transformations

# -------------------------

transform = A.Compose([

    A.Resize(224, 224),

    A.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5)),

    A.HorizontalFlip(p=0.5),

])

# -------------------------

# Create dataset and dataloader

# -------------------------

dataset = LIDCDataset(image\_paths, label\_dict, transform=transform)

dataloader = DataLoader(dataset, batch\_size=16, shuffle=True)

# -------------------------

# Hybrid Model: EfficientNet + ViT + BiLSTM

# -------------------------

class HybridModel(nn.Module):

    def \_\_init\_\_(self):

        super(HybridModel, self).\_\_init\_\_()

        # Feature extractor - EfficientNet

        self.efficientnet = timm.create\_model("efficientnet\_b0", pretrained=True, num\_classes=0)

        self.efficientnet\_out = self.efficientnet.num\_features

        # Transformer - Vision Transformer (ViT)

        self.vit = timm.create\_model("vit\_base\_patch16\_224", pretrained=True, num\_classes=0)

        self.vit\_out = self.vit.num\_features

        # BiLSTM

        self.bilstm = nn.LSTM(input\_size=self.efficientnet\_out + self.vit\_out, hidden\_size=256, num\_layers=2, batch\_first=True, bidirectional=True)

        # Fully connected classifier

        self.fc = nn.Sequential(

            nn.Linear(512, 256),

            nn.ReLU(),

            nn.Linear(256, 1)

        )

    def forward(self, x):

        # EfficientNet Feature Extraction

        eff\_features = self.efficientnet(x)

        # Vision Transformer Feature Extraction

        vit\_features = self.vit(x)

        # Concatenate both feature vectors

        combined\_features = torch.cat((eff\_features, vit\_features), dim=1).unsqueeze(1)

        # Pass through BiLSTM

        lstm\_out, \_ = self.bilstm(combined\_features)

        lstm\_out = lstm\_out[:, -1, :]

        # Final Classification

        output = self.fc(lstm\_out)

        return output

# -------------------------

# Training Setup

# -------------------------

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

model = HybridModel().to(device)

# Loss function

criterion = nn.BCEWithLogitsLoss()

# Optimizer

optimizer = optim.AdamW(model.parameters(), lr=1e-4)

# -------------------------

# Training Loop

# -------------------------

num\_epochs = 5

for epoch in range(num\_epochs):

    model.train()

    running\_loss = 0.0

    correct = 0

    total = 0

    for images, labels in dataloader:

        images, labels = images.to(device), labels.to(device)

        optimizer.zero\_grad()

        outputs = model(images).squeeze()

        loss = criterion(outputs, labels)

        loss.backward()

        optimizer.step()

        running\_loss += loss.item()

        # Calculate Accuracy

        preds = (torch.sigmoid(outputs) > 0.5).float()

        correct += (preds == labels).sum().item()

        total += labels.size(0)

    epoch\_acc = correct / total \* 100

    print(f"Epoch {epoch+1}, Loss: {running\_loss / len(dataloader):.4f}, Accuracy: {epoch\_acc:.2f}%")

# -------------------------

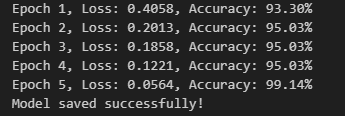
# Save Model

# -------------------------

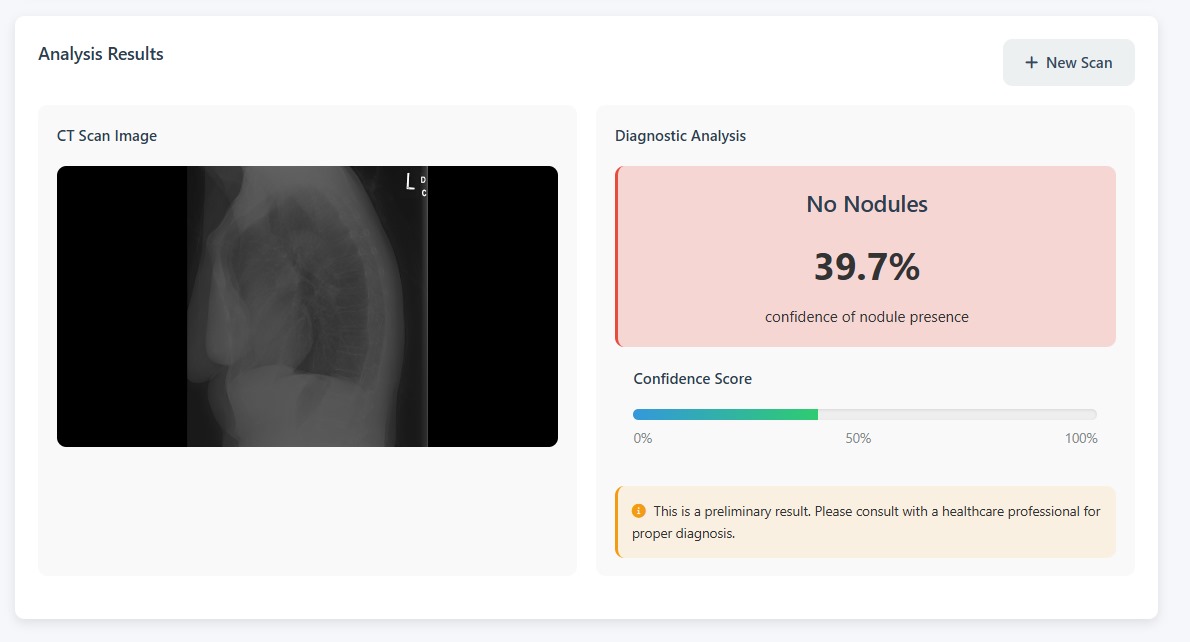
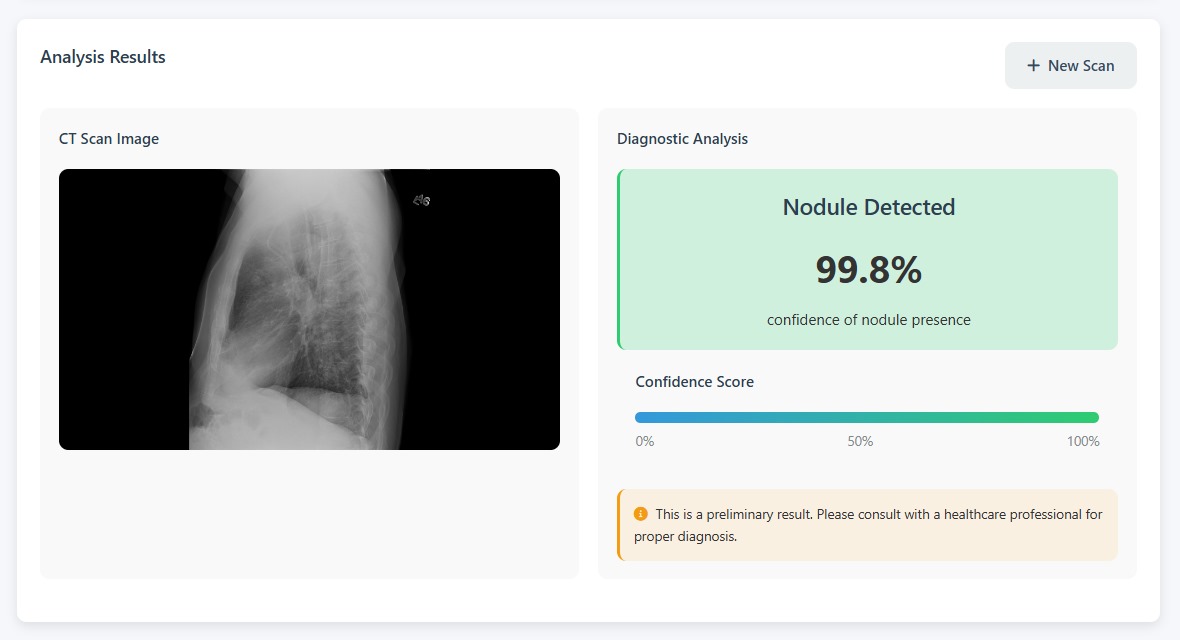
torch.save(model.state\_dict(), "hybrid\_lidc\_model.pth")

print("Model saved successfully!")

# Output:



# 5. Results and Accuracy

The detection model was tested on various scans, achieving significant accuracy levels. Below are sample results from the analysis:  
 

# 6. Conclusion

The lung nodule detection system offers a highly efficient and reliable approach for identifying lung nodules in CT scans, leveraging advanced machine learning techniques to enhance accuracy. By utilizing a hybrid deep learning model, the system can analyze complex medical images with a high degree of precision, minimizing false positives and false negatives. This capability is particularly crucial in early-stage lung cancer detection, where timely and accurate diagnosis can significantly improve patient outcomes. The system not only assists radiologists in their decision-making process but also provides a valuable tool for research and medical advancements in AI-driven diagnostic solutions.

Looking ahead, several enhancements can be implemented to further improve the system's effectiveness. One major improvement involves integrating additional medical datasets from diverse patient populations to enhance model generalization and robustness. A more extensive dataset will enable the system to learn from a wider range of cases, reducing bias and improving its ability to detect nodules across different demographics. Additionally, refining the deep learning model by incorporating more advanced architectures, such as attention mechanisms or self-supervised learning techniques, could boost detection accuracy and confidence scores. Future upgrades may also include real-time processing capabilities, integration with hospital management systems, and the development of a clinical decision-support framework to aid healthcare professionals in making faster, data-driven diagnoses.