**Lung Risk and Severity Prediction**

**Introduction**

Lung diseases, including lung cancer and other respiratory conditions, are among the leading causes of mortality worldwide. Early detection and severity prediction are crucial for timely treatment and improved patient outcomes. In this research, we aim to analyze available datasets and machine learning models to determine the best approach for predicting lung disease severity and risk.

**Dataset Selection**

We explored multiple publicly available datasets to identify the most suitable one for our study. Among the datasets examined, we found that the dataset quality and annotation accuracy significantly impacted the model's performance. The primary datasets considered are:

**1. LungNet22**

* Source: Available on the internet
* Contains: Annotated chest CT images
* Limitation: Inconsistencies in annotations and varying image quality affected model performance.

**2. Roboflow Datasets**

* [**Chest CT Dataset**](https://universe.roboflow.com/e-jw7f7/chest-ct) (used in this project)
* [**Lung Cancer Dataset**](https://universe.roboflow.com/mendozajpd/lung-cancer-dataset-chgjz)
* Strengths: Easily accessible and diverse data samples.
* Weaknesses: Annotations are not optimal, leading to lower mAP (mean Average Precision) values.

Despite analyzing several datasets, we observed that the model's mAP value was consistently low due to dataset quality and annotation inaccuracies.

**Best Model for Lung Disease Prediction**

After extensive research, we found that **The Sybil AI Model** is currently one of the most effective models used in hospitals for lung disease prediction. This model was developed using high-quality datasets from reputable sources and has demonstrated superior performance in risk assessment and severity prediction.

**Training and Evaluation Data for Sybil Model**

* **Training Data:**
  + 6,839 LDCT (Low-Dose Computed Tomography) scans from the **National Lung Screening Trial (NLST)** dataset.
* **Validation Data:**
  + 6,282 additional LDCT scans from the NLST dataset.
* **External Testing:**
  + **Massachusetts General Hospital (MGH):** 8,821 LDCT scans.
  + **Chang Gung Memorial Hospital (CGMH) in Taiwan:** 12,280 LDCT scans, including participants with varying smoking histories, including non-smokers.

Despite our best efforts, we could not obtain access to the **NLST, MGH, and CGMH datasets** within the short time available. These datasets are crucial for building a highly accurate lung disease prediction model.

**Dataset Preparation for Training**

To build a robust model, we need to structure the dataset with both categorical labels and corresponding CT scans. The dataset will be annotated with bounding boxes to detect lung health status and estimate lifespan predictions. Our dataset should follow these categories based on **The Sybil Model’s** methodology:

**Lung Cancer Risk Prediction Categories**

The model will predict lung cancer risk probabilities across six time windows:

1. **1 Year**
2. **2 Years**
3. **3 Years**
4. **4 Years**
5. **5 Years**
6. **6 Years**

These predictions will be made by analyzing a single low-dose chest CT scan to assess the likelihood of developing lung cancer within the given timeframe.

**CT Scan Image Resolutions and Size Standardization**

To ensure consistency and improve model accuracy, we need to define the following:

* **Resolution Standardization:** Optimal CT scan resolution for training should be standardized to ensure uniformity across datasets.
* **Image Size Organization:** Images should be resized to maintain clarity and ensure compatibility with deep learning models.
* **Date-Time Metadata:** Every CT scan should be labeled with its corresponding date and time to track disease progression over time.
* **Hospital Data Collection:** Collaboration with hospitals to obtain high-quality, well-annotated scans for model training.

**Tech Stack**

1. **Programming Language**: Python
2. **Framework**: Streamlit (for UI)
3. **Deep Learning**: PyTorch, Ultralytics YOLO (for object detection)
4. **Computer Vision**: OpenCV, PIL (for image processing)
5. **Data Handling**: NumPy (for array manipulations)
6. **Deployment**: Local system execution with model inference

**Solution Code Summary**

1. **Model Loading**: The system loads a YOLO model trained on lung disease detection from a local directory.
2. **Prediction Pipeline**:
   * Accepts an input chest X-ray or CT scan.
   * Detects regions affected by lung diseases like cancer or nodules.
   * Calculates the affected lung area and classifies severity into **Mild, Moderate, or Severe**.
3. **Visualization**:
   * Draws bounding boxes around detected abnormalities.
   * Displays the processed image along with a severity assessment.
4. **UI & Customization**:
   * Built using Streamlit with an intuitive image upload option.
   * Includes animated GIF backgrounds and a styled UI for an enhanced user experience.

Here’s a breakdown of how the **YOLO model prediction methodology** works and how the **severity classification** is determined.

**1️. YOLO Model Prediction Workflow**

The **YOLO (You Only Look Once)** model is used for detecting lung diseases from chest CT scans or X-rays. Here's the step-by-step process:

**🔹 Step 1: Load the Pretrained Model**

* The YOLO model (e.g., best.pt) is loaded using the Ultralytics YOLO library (yolov5 or yolov8).
* The model has been trained on labeled lung disease datasets where bounding boxes and classes (e.g., "cancerous", "nodule", "normal") are predefined.

from ultralytics import YOLO

# Load the trained model

model = YOLO("best.pt") # Load the custom-trained lung disease model

**🔹 Step 2: Make Predictions on an Input Image**

* The model takes an input chest CT scan or X-ray and processes it through the YOLO architecture.
* The model outputs:
  1. **Bounding boxes** around detected abnormalities.
  2. **Class labels** (e.g., lung cancer, nodules, etc.).
  3. **Confidence scores** (probability of detection accuracy).

import cv2

# Load an input image

image\_path = "test\_image.jpg"

image = cv2.imread(image\_path)

# Perform prediction

results = model(image)

# Extract detections

for result in results:

boxes = result.boxes # Bounding boxes

labels = result.names # Class labels

scores = result.probs # Confidence scores

**2️. Severity Classification Methodology**

The severity of lung disease is determined based on the following factors:

**🔹 Step 1: Extract Affected Lung Area**

* The **bounding box dimensions** from YOLO help calculate the **total affected area** in pixels.
* We compare the affected area against the total lung area in the scan.

# Calculate affected area from YOLO bounding boxes

affected\_area = 0

total\_area = image.shape[0] \* image.shape[1] # Total pixels in the scan

for box in boxes:

x1, y1, x2, y2 = box.xyxy[0] # Get bounding box coordinates

box\_area = (x2 - x1) \* (y2 - y1) # Compute bounding box area

affected\_area += box\_area # Sum up affected areas

# Compute affected percentage

affected\_percentage = (affected\_area / total\_area) \* 100

**🔹 Step 2: Define Severity Levels**

Based on the **affected percentage**, we classify the severity as:

| **Affected Area %** | **Severity Level** |
| --- | --- |
| 0 - 10% | **Mild** |
| 10 - 30% | **Moderate** |
| >30% | **Severe** |

# Severity classification logic

if affected\_percentage < 10:

severity = "Mild"

elif affected\_percentage < 30:

severity = "Moderate"

else:

severity = "Severe"

print(f"Lung disease severity: {severity}")

**3️. Displaying Results**

* The final results, including **bounding boxes, labels, and severity level**, are displayed on the image using OpenCV.

import matplotlib.pyplot as plt

# Draw bounding boxes on the image

for box in boxes:

x1, y1, x2, y2 = map(int, box.xyxy[0])

cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2) # Green box for detection

cv2.putText(image, severity, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

# Display the image

plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

plt.axis("off")

plt.show()

**4️. Key Takeaways**

1. **YOLO detects lung abnormalities** by drawing bounding boxes on CT scans.
2. **Affected lung area is computed** from bounding box sizes.
3. **Severity is determined** based on the percentage of affected lung tissue.
4. **Results are visualized** with bounding boxes and severity classification.

From the training log in the image, here’s a breakdown of the model's performance and key metrics:

**🔹 Model Overview**

* **Dataset Size:**
  + **Total images:** 10,780
  + **Training set:** 9,419 images
  + **Validation set:** 907 images
  + **Test set:** 454 images
* **Classes:**
  + 0: Cancer Malignant
  + 1: Nodule
  + 2: Multiple Nodules
* **Training Duration:** ~5.15 hours for 50 epochs
* **Hardware Used:** NVIDIA RTX 3050 (6GB VRAM) with CUDA

**🔹 Model Performance Metrics**

The model was evaluated using the following metrics:

**1️⃣ Loss Metrics**

These values indicate how well the model is learning:

* **Box Loss**: Measures how well the predicted bounding box aligns with the ground truth.
* **Classification Loss**: Measures how well the model classifies detected objects.
* **DFL (Distribution Focal Loss)**: Measures the quality of object localization.

📌 **Final Values (Epoch 50/50):**

* **Box Loss**: 0.4289
* **Cls Loss (Classification Loss)**: 0.8016
* **DFL Loss**: 0.975

Lower loss values indicate better performance.

**2️⃣ Precision, Recall, and mAP**

These are key detection performance metrics:

| **Metric** | **Value** |
| --- | --- |
| **Precision (P)** | 0.975 |
| **Recall (R)** | 0.975 |
| **mAP@50** | 0.986 |
| **mAP@50-95** | 0.922 |

📌 **Interpretation:**

* **Precision (0.975):** When the model predicts a disease, it is correct 97.5% of the time.
* **Recall (0.975):** The model correctly detects 97.5% of all actual disease cases.
* **mAP@50 (0.986):** The model achieves 98.6% accuracy at an IoU threshold of 50%.
* **mAP@50-95 (0.922):** The model maintains 92.2% accuracy over multiple IoU thresholds (more strict evaluation).

**3️⃣ Class-wise Metrics**

The breakdown of precision, recall, and mAP per class:

| **Class** | **Precision** | **Recall** | **mAP@50** | **mAP@50-95** |
| --- | --- | --- | --- | --- |
| **0 (Cancer Malignant)** | 0.975 | 0.975 | 0.986 | 0.923 |
| **1 (Nodule)** | 0.994 | 0.994 | 0.974 | 0.915 |
| **2 (Multiple Nodules)** | 0.994 | 0.994 | 0.975 | 0.903 |

📌 **Interpretation:**

* Class 1 (Nodule) has slightly better performance than 0 and 2, with very high recall.
* The model performs well across all classes, with minor variations.

**🔹 Final Thoughts**

✅ **Strengths:**

* High **precision (97.5%)** and **recall (97.5%)** indicate strong performance.
* **High mAP (98.6%)** ensures accurate detection.
* Well-optimized training on **NVIDIA RTX 3050 (6GB)**.

⚠️ **Possible Improvements:**

* Fine-tuning hyperparameters to reduce classification loss.
* Adding more diverse samples for better generalization.
* Testing on real-world images to verify model robustness.

**YOLO Model Inference Code with Streamlit UI and Results**

**import streamlit as st**

**import cv2**

**import numpy as np**

**from PIL import Image**

**import torch**

**from ultralytics import YOLO**

**import base64**

**def load\_model():**

**model = YOLO(r"C:\Users\iproat26\Desktop\project\Lung Diesease Prediction\runs\detect\train8\weights\best.pt")  # Replace with actual model path**

**return model**

**def predict(image, model):**

**results = model(image)**

**bounding\_boxes = []**

**total\_area = image.shape[0] \* image.shape[1]**

**affected\_area = 0**

**class\_labels = {0: "Cancer", 1: "Nodule", 2: "Multi Nodule"}**

**detected\_classes = []**

**for result in results:**

**for box in result.boxes:**

**x1, y1, x2, y2 = box.xyxy[0].cpu().numpy()**

**confidence = box.conf[0].cpu().numpy()**

**class\_id = int(box.cls[0].cpu().numpy())**

**box\_area = (x2 - x1) \* (y2 - y1)**

**affected\_area += box\_area**

**bounding\_boxes.append([int(x1), int(y1), int(x2), int(y2), class\_id])**

**detected\_classes.append(class\_labels[class\_id])**

**severity\_percentage = (affected\_area / total\_area) \* 100**

**severity\_label = categorize\_severity(severity\_percentage)**

**return bounding\_boxes, severity\_percentage, severity\_label, detected\_classes**

**def categorize\_severity(severity\_percentage):**

**if severity\_percentage < 10:**

**return "Mild"**

**elif severity\_percentage < 30:**

**return "Moderate"**

**else:**

**return "Severe"**

**def draw\_bounding\_boxes(image, bounding\_boxes):**

**colors = {0: (255, 0, 0), 1: (0, 255, 0), 2: (0, 0, 255)}**

**for box in bounding\_boxes:**

**x1, y1, x2, y2, class\_id = box**

**color = colors[class\_id]**

**cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)**

**return image**

**# Streamlit UI Customization**

**st.set\_page\_config(page\_title="Lung Cancer Prediction", layout="wide")**

**def get\_base64\_of\_gif(file\_path):**

**with open(file\_path, "rb") as file:**

**encoded\_gif = base64.b64encode(file.read()).decode()**

**return encoded\_gif**

**bg\_gif = get\_base64\_of\_gif("bg.gif")**

**# background-image: url("data:image/gif;base64,{bg\_gif}");**

**# Background styling with your GIF**

**page\_bg\_img = f'''**

**<style>**

**.block-container {{**

**padding-top: 0rem;  /\* Or any value you want \*/**

**padding-bottom: 0rem; /\* Optional: adjust bottom padding \*/**

**}}**

**[data-testid="stAppViewContainer"] {{**

**background-image: url("https://media3.giphy.com/media/v1.Y2lkPTc5MGI3NjExcTl2MjFzdG1iYjRqdjcxazdoOWEyNjdjbWM2YWhkYTFrMnZnamVvdiZlcD12MV9pbnRlcm5hbF9naWZfYnlfaWQmY3Q9Zw/dt0KXLj7bzwZuRQBwY/giphy.gif");**

**background-size: 45% auto;**

**background-repeat: no-repeat;**

**background-position: right center;**

**background-attachment: fixed;**

**padding-top: 0px;**

**}}**

**[data-testid="stHeader"] {{**

**background: rgba(0, 0, 0, 0);**

**}}**

**/\* File uploader styling \*/**

**[data-testid="stFileUploader"] {{**

**width: 50%; /\* Or any desired width \*/**

**float: left; /\* Or any desired layout \*/**

**}}**

**.st-emotion-cache-1gulkj5 {{**

**width: 100% !important;**

**max-width: 100% !important;**

**}}**

**.st-emotion-cache-7ym5gk {{**

**width: 100% !important;**

**max-width: 100% !important;**

**}}**

**/\* Content container \*/**

**.st-emotion-cache-1v0mbdj {{**

**margin: 0 auto;**

**display: block;**

**}}**

**/\* Column styling \*/**

**.st-emotion-cache-1kyxreq {{**

**justify-content: center;**

**}}**

**</style>**

**'''**

**st.markdown(page\_bg\_img, unsafe\_allow\_html=True)**

**st.title("🫁 Lung Cancer Detection & Severity Assessment")**

**st.subheader("Upload a Chest X-ray or CT Scan 🔬")**

**model = load\_model()**

**uploaded\_file = st.file\_uploader("Choose an Image", type=["jpg", "png", "jpeg"])**

**if uploaded\_file:**

**image = Image.open(uploaded\_file)**

**image = np.array(image)**

**bounding\_boxes, severity\_percentage, severity\_label, detected\_classes = predict(image, model)**

**image\_with\_boxes = draw\_bounding\_boxes(image.copy(), bounding\_boxes)**

**col1, col2 = st.columns([3, 2])**

**with col1:**

**st.image(image\_with\_boxes, caption="Detection Results")**

**with col2:**

**st.markdown(f"### 🏥 Severity Level: {severity\_label}")**

**st.markdown(f"\*\*Affected Area:\*\* {severity\_percentage:.2f}% of Lungs")**

**st.markdown(f"### 🔍 Detected Conditions:")**

**for cls in set(detected\_classes):**

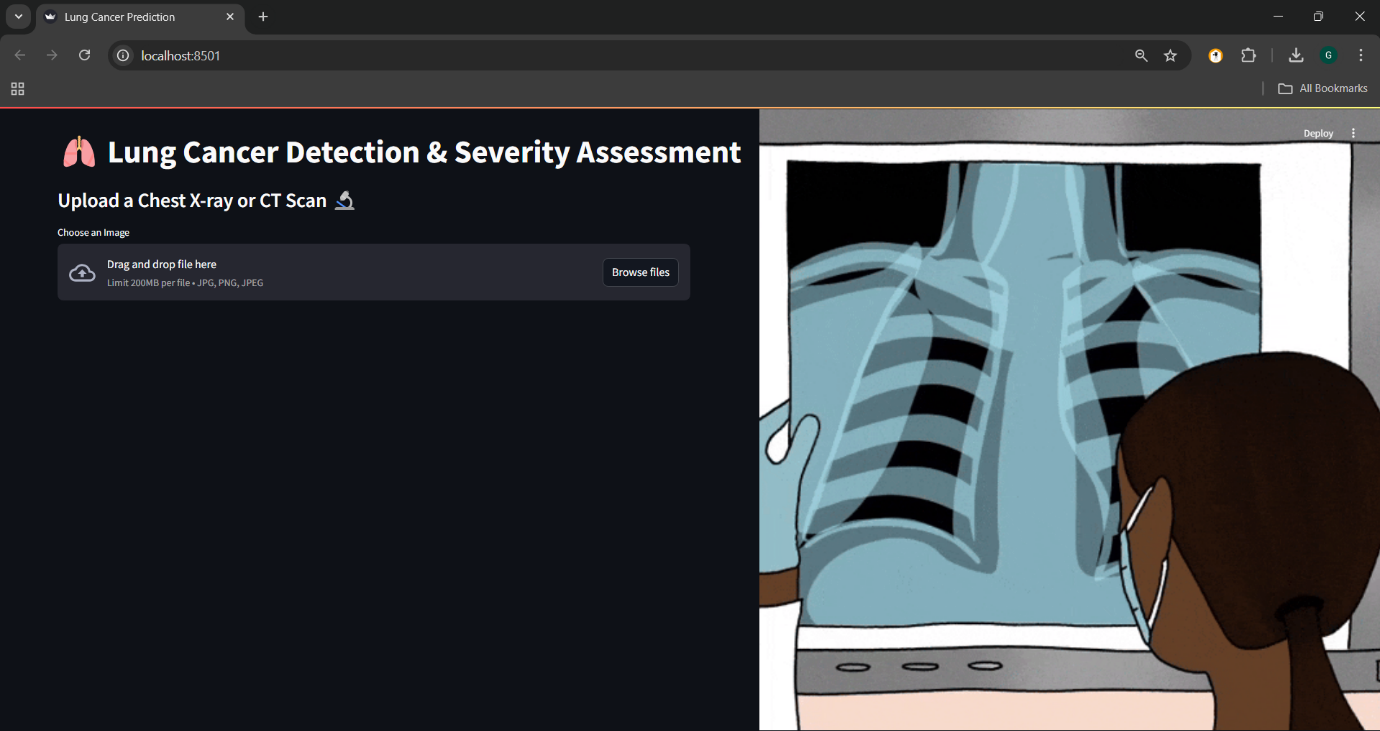
**st.markdown(f"- \*\*{cls}\*\*")**

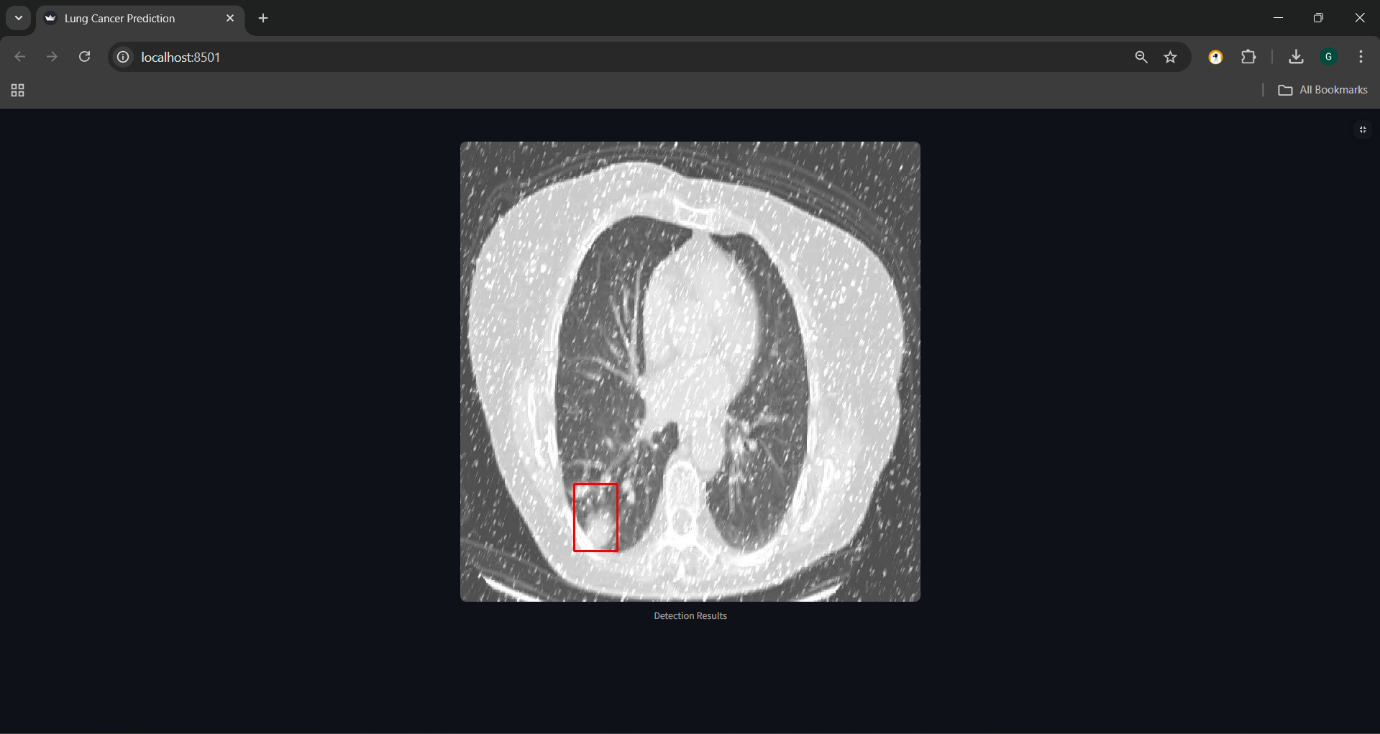
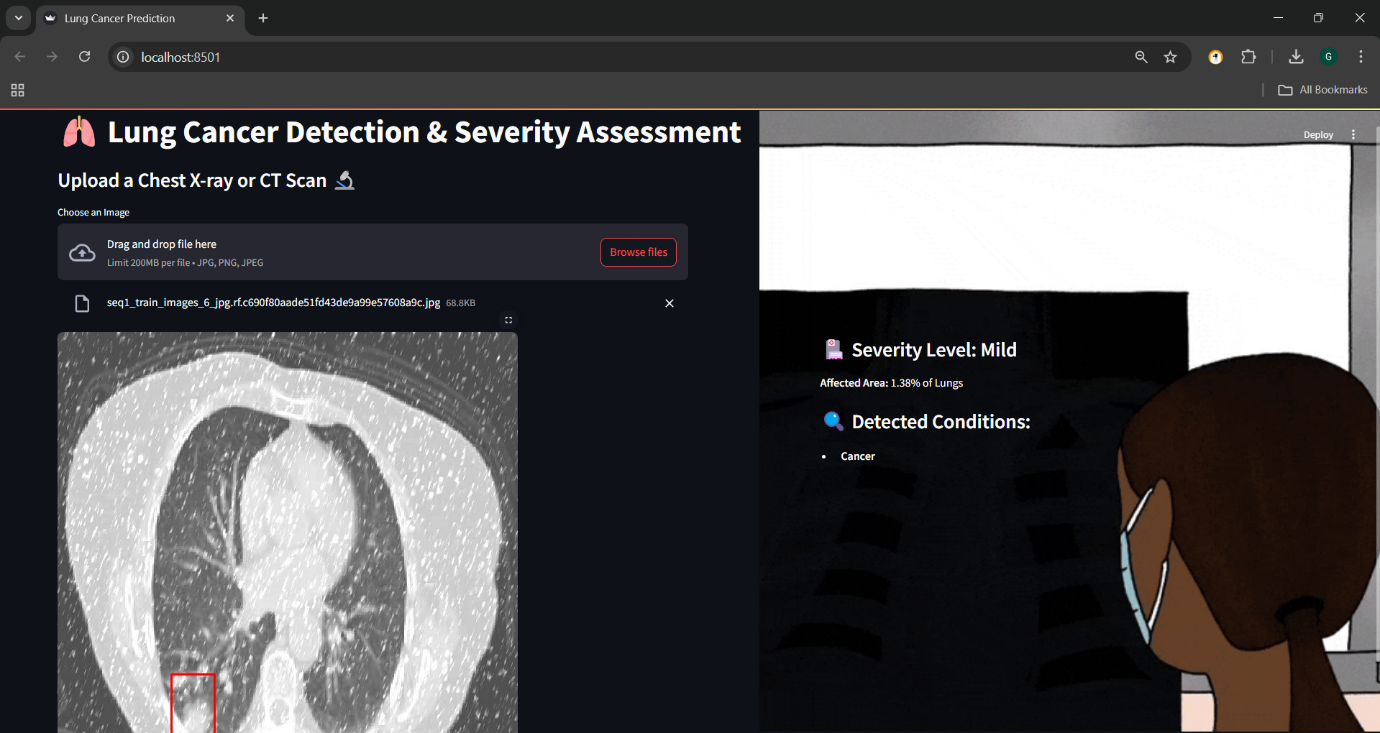
**output:**

**0: 640x640 1 0, 84.4ms**

**Speed: 14.5ms preprocess, 84.4ms inference, 6.8ms postprocess per image at shape (1, 3, 640, 640)**

**OUTPUT IN UI:**





**Git Repo and code details:**

**REPO: https://github.com/K-GOKULAPPADURAI/Lung-Cancer-Prediction-With-Severity.git**