COMPUTER VISION

PNEUMONIA DETECTION CHALLENGE

Capstone Project
Group 1

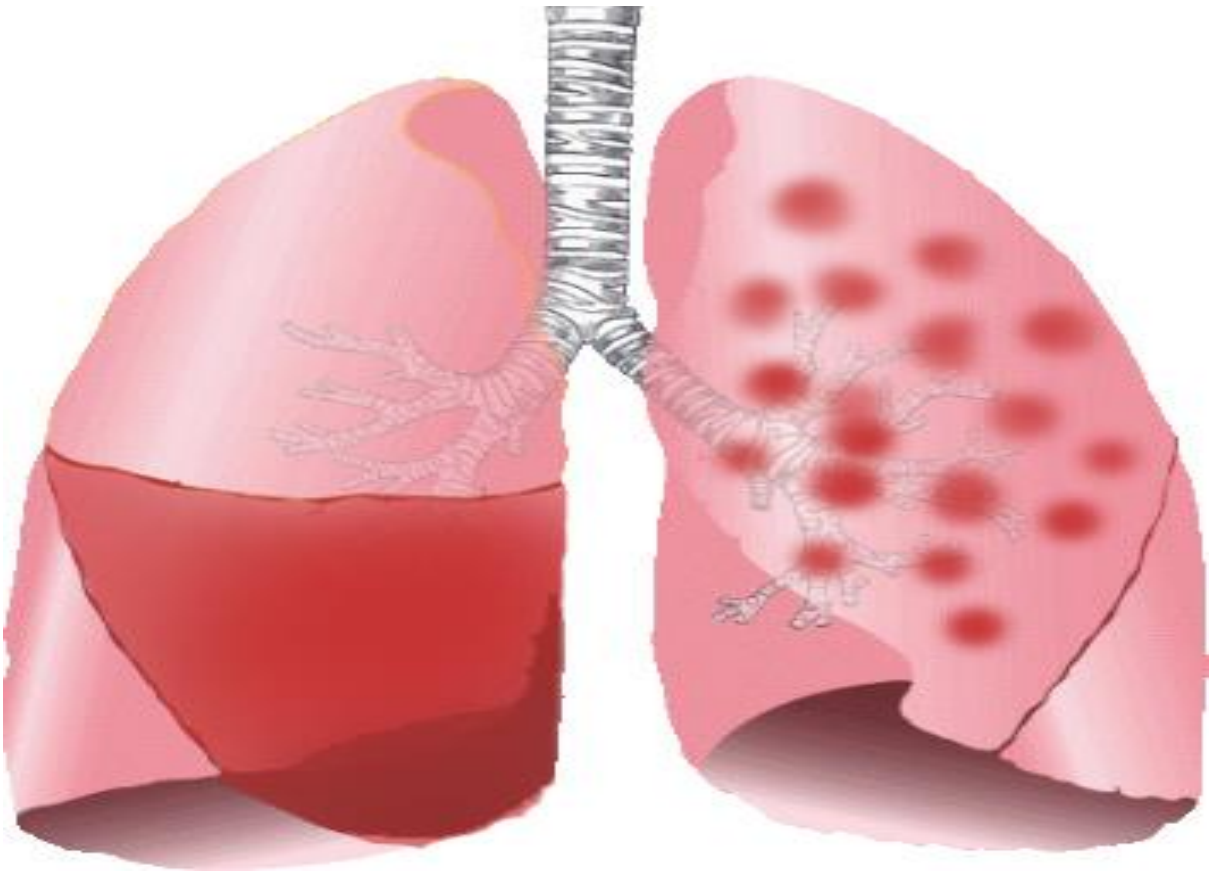
Team

Jyant Mahara	**Capstone Project Mentor**
Sneh Sweta	Capstone Group 1 Team
Harsha Bhat	Capstone Group 1 Team
Renjini Jayakumar	Capstone Group 1 Team
Shivee	Capstone Group 1 Team
Revathy Jayakumar	Capstone Group 1 Team

Table of Contents

Executive Summary

Pneumonia is a life-threatening lung infection that affects millions of people worldwide. Traditional pneumonia diagnosis methods can be time-consuming and expensive. Artificial Intelligence (AI) has the potential to revolutionize pneumonia detection by enabling accurate and timely diagnoses.



AI algorithms can analyze chest X-rays and CT scans to identify patterns and anomalies associated with pneumonia. These algorithms can detect pneumonia with high accuracy, potentially reducing the need for invasive and expensive diagnostic procedures.

AI-powered pneumonia detection systems can also help healthcare professionals prioritize treatment for patients with severe cases of pneumonia, ensuring that they receive timely and appropriate care.

However, AI-powered pneumonia detection systems are not without their challenges. One major concern is the potential for bias in AI algorithms, which could lead to misdiagnosis or underdiagnosis of certain patient populations. Addressing these concerns will be crucial for ensuring the safe and effective use of AI in pneumonia diagnosis.

Overall, AI-powered pneumonia detection systems have the potential to improve the accuracy and speed of diagnosis. AI systems can analyze medical images and patient data to identify patterns and make predictions, allowing for more accurate diagnosis and treatment. This technology can also help healthcare professionals to make more informed decisions about patient care, reducing the risk of misdiagnosis and improving patient outcomes. However, there are still challenges to be addressed, such as the need for high-quality data and the need for AI algorithms to be validated in clinical settings. With continued development and refinement, AI- based pneumonia detection has the potential to revolutionize the field of medical imaging and improve the care of patients with respiratory infections.

PROBLEM STATEMENT

DOMAIN: Health Care

CONTEXT: Computer vision can be used in health care for identifying diseases. In Pneumonia detection we need to detect Inflammation of the lungs. In this challenge, you're required to build an algorithm to detect a visual signal for pneumonia in medical images. Specifically, your algorithm needs to automatically locate lung opacities on chest radiographs.

DATA DESCRIPTION: The goal is to build a computer vision algorithm to detect pneumonia by identifying lung opacities in chest radiograph images. The dataset includes medical images in DICOM format, containing both pixel data and metadata. The classification task involves three classes:

- Pneumonia - Lung opacities indicating pneumonia.
- Not Normal No Lung Opacity - Abnormalities without pneumonia that might resemble pneumonia in appearance. This extra third class indicates that while pneumonia was determined not to be present, there was nonetheless some type of abnormality on the image and oftentimes this finding may mimic the appearance of true pneumonia.
- Normal - Healthy cases with no visible abnormalities.

The challenge is to automatically locate and identify lung opacities accurately, distinguishing true pneumonia cases from other abnormalities to assist in diagnosis.

Dicom original images: - Medical images are stored in a special format called DICOM files (*.dcm).

You can refer to the details of the dataset in the above link – Acknowledgements: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview/acknowledgements.

1. Summary of the Problem statement

Project Objective:

The primary objective of this project is to develop a robust machine learning pipeline for analysing chest radiographs, specifically to detect pneumonia by identifying lung opacities in DICOM images. The tasks include:

1. Fine-tuning basic CNN models for binary classification of chest radiographs (Pneumonia vs. Non-Pneumonia).
2. Applying transfer learning for improved accuracy and generalization.
3. Designing and testing advanced object detection models, such as RCNN and its hybrids, to impose bounding boxes or masks over the affected regions.
4. Creating a reusable model by saving it for future predictions.

The data and the dataset contain:

- stage_2_detailed_class_info.csv – contains attributes patientId and class information(Normal/Not Normal/Lung Opacity)
- stage_2_train_labels.csv – contains attributes patientId, x, y, width, height and Target
- stage_2_train_images – 22684 images in .dcm format, to be used for training the model
- stage_2_test_images – 3000 images in. dcm format.

Findings:

To effectively manage and analyse this data, we can break it down into the following components:

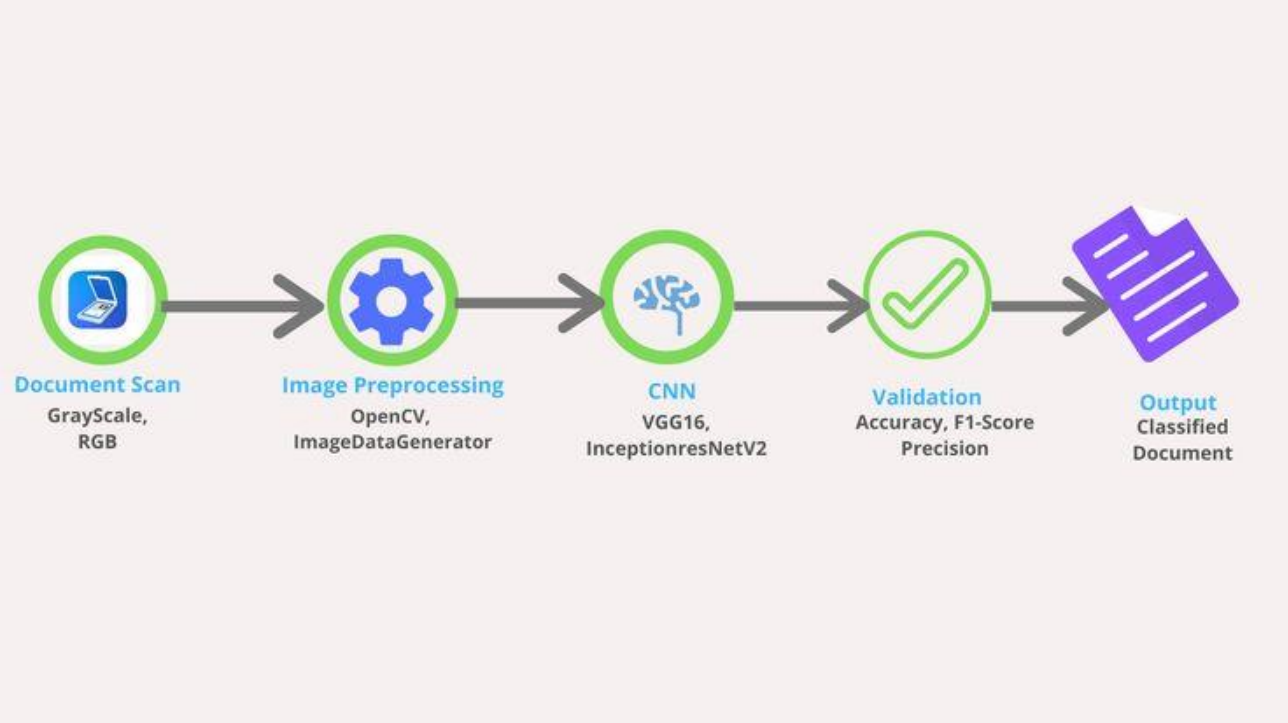
- **Patient IDs:** Some of which are duplicated due to multiple bounding boxes per patient
- **Bounding box coordinates:** Defined by x, y, width, and height
- **Target:** Target = 1 indicates the presence of pneumonia. Target = 0 indicates no definitive evidence of pneumonia.
- **Class Labels:** The dataset also includes a class label with three possible categories:
    - Lung Opacity: Suggests the presence of pneumonia (corresponding to Target = 1).
    - Normal or No Lung Opacity / Not Normal: Corresponds to Target = 0 (indicating no pneumonia).
- **Bounding Box Distribution:** A large proportion of patient IDs (23,286 or 87%) have only one bounding box. A small number of patients (13 individuals) have up to four bounding boxes, potentially indicating multiple findings within a single image.

  Differences observed between training images and no of information given in the .csv files.

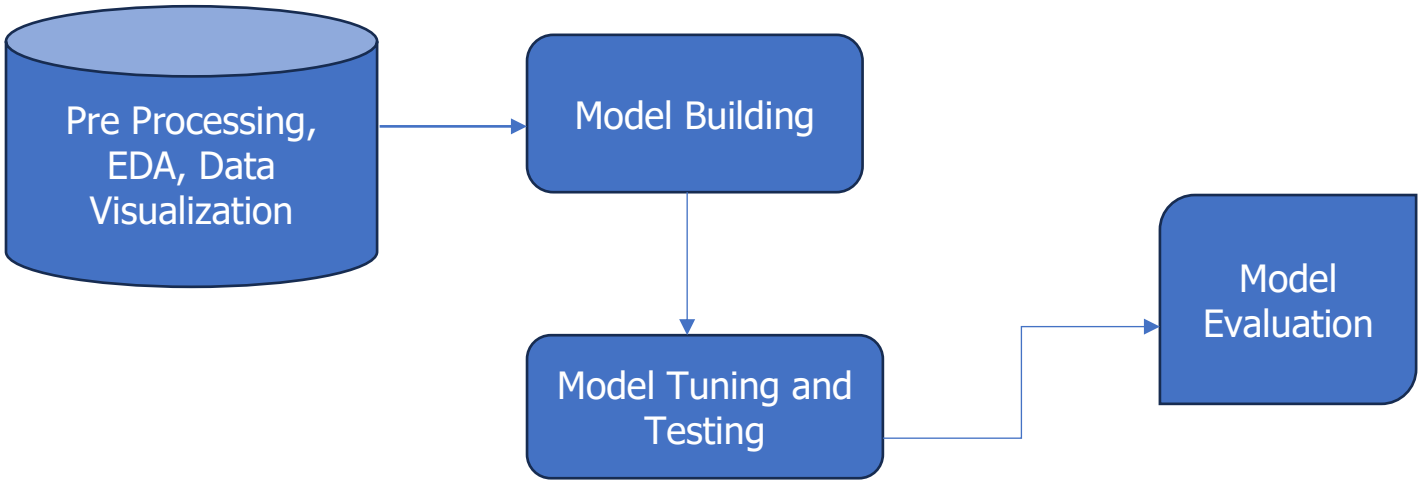
  No. of training Images: 26684

  No. of rows given in csv file:30227

2. OVERVIEW OF THE FINAL PROCESS



Approach:



Algorithm for Pneumonia Detection

STAGE 1: DATA PRE-PROCESSING, DATA VISUALISATIONS AND EDA

• Understand the problem statement and exploring the data.

◦ Merge stage2_train_labels.cs and stage_2_detailed_class_info.csv.

◦ Extract Metadata from DICOM images (stage_2_train_images) and store in a .csv/.pkl format.

◦ Missing value and class Imbalance treatment.

◦ Univariate analysis, bivariate analysis between predictor and target column. Visualize Analysis by histogram, pair-plot, count plot and plot different classes of image and draw conclusions.

STAGE 2: Model Building

«Split the data to train, test and validation set.

◦ Train the model with basic CNN first and calculate score.

◦ Saving the weights and model for further improvement.

◦ Apply different advanced algorithms like VGG-Net, Mask R-CNN, YOLO. Choose the model which performs better.

STAGE 3 and 4- Model Tuning, Testing and Evaluation

Plot AUC curve -which indicates False Positives and True Negatives.

« Improve the model by applying different:

◦ optimization techniques - Adam and RMSPROP.

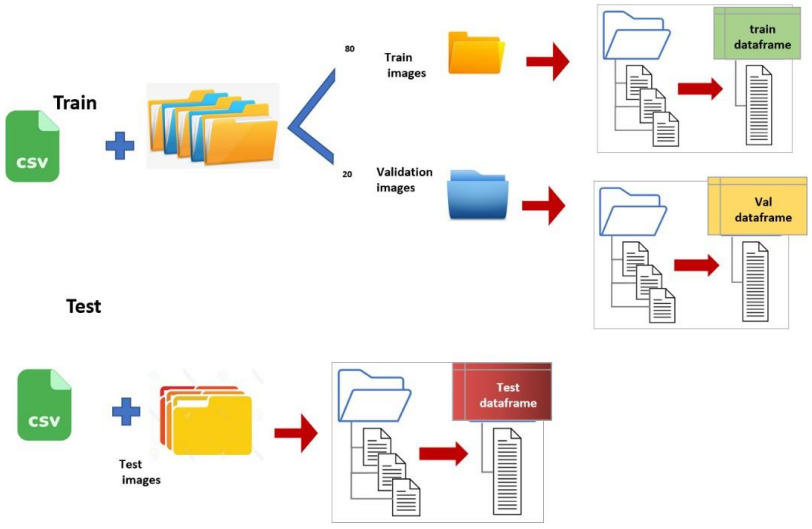
◦ Batch Normalization, Dropout, learning rate.

◦ Early Stopping to prevent over fitting.

Evaluate and observe the behaviour of model on tuning different hyper parameters.

3. Step-by-step walk through the solution

Data Modeling Solution Overview



SOLUTION OVERVIEW

- **Data Preprocessing**: Involves loading and preparing images and annotations, resizing images, and scaling bounding boxes for data modeling.
- **Transfer Learning**: Pre-trained models like VGG16 and ResNet50 are used to enhance performance. VGG16 performed best in terms of testing accuracy.
- **Model Development**: Includes fine-tuning CNN models, applying bounding box detection, and creating a GUI to visualize predictions.

This solution leverages the power of transfer learning to achieve high accuracy and robust predictions for both classification and bounding box detection.

To solve this challenge, transfer learning was employed using pre-trained models available in TensorFlow. By fine-tuning specific layers of these pre-built models, we

aimed to enhance accuracy and optimize bounding box predictions. Transfer learning offers flexibility, allowing pre-trained models to be used as feature extractors or as components of entirely new architectures.

In this project, several pre-trained models—vgg16, ResNet50 were evaluated Both VGG16 and ResNet outperform the baseline and fine-tuned models on validation and testing datasets. VGG16 has the highest testing accuracy, making it the best performer.

Step 1: Problem Understanding

- **Objective**: Automatically detect pneumonia in chest radiographs by identifying lung opacities.
- **Challenge**: Handling DICOM medical images, preprocessing data efficiently, leveraging transfer learning, and integrating the solution into a usable system (e.g., GUI).
- **Approach**: Use a combination of transfer learning and custom deep learning techniques.

Step 2: Data Loading and Data Cleaning:

Check Shape of the dataset

```
[4] train_labels = pd.read_csv('/content/drive/MyDrive/stage_2_train_labels.csv')
    print('First five rows of Training set:\n', train_labels.head())
```

```
First five rows of Training set:
                                patientId      x      y  width  height  Target
0  0004cfab-14fd-4e49-80ba-63a80b6bddd6    NaN    NaN    NaN     NaN       0
1  00313ee0-9eaa-42f4-b0ab-c148ed3241cd    NaN    NaN    NaN     NaN       0
2  00322d4d-1c29-4943-afc9-b6754be640eb    NaN    NaN    NaN     NaN       0
3  003d8fa0-6bf1-40ed-b54c-ac657f8495c5    NaN    NaN    NaN     NaN       0
4  00436515-870c-4b36-a041-de91049b9ab4  264.0  152.0  213.0   379.0       1
```

Shape of dataset:

```
[8] print(f'Class labels dataframe has {class_labels.shape[0]} rows and {class_labels.shape[1]} columns')
```

```
Class labels dataframe has 30227 rows and 2 columns
```

Check for Missing Values

```
# Checking nulls in class_labels:
print('Number of nulls in class columns: {}'.format(class_labels['class'].isnull().sum()))
```

```
Number of nulls in class columns: 0
```

+ Code + Text

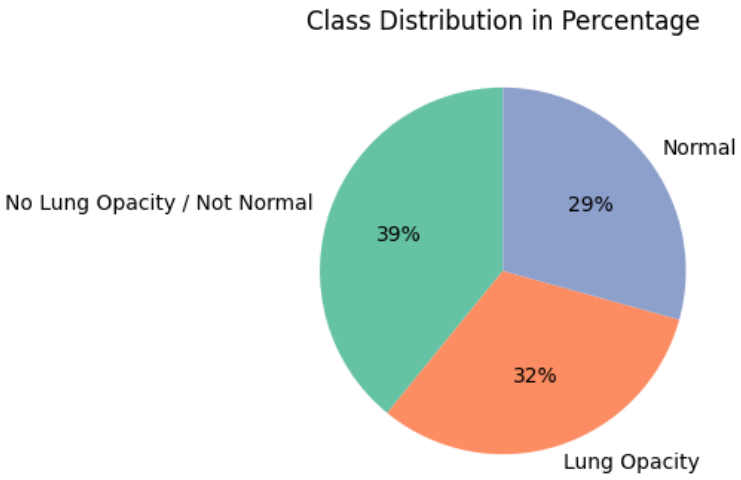
Check for Duplicate Value

```
[9] # Number of duplicates in patients:

    print('Number of duplicates in patientID in class labels dataframe: {}'.format(len(class_labels) - (class_labels['patientId'].nunique())))
    print('Number of unique patientId are: {}'.format(class_labels['patientId'].nunique()))
```

```
Number of duplicates in patientID in class labels dataframe: 3543
Number of unique patientId are: 26684
```

Step 3: Data Preprocessing:

a. Class Distribution



Map Training and Testing Images to Their Classes:

```
# Step 2: Map Training and Testing Images to Their Classes
# Map patient IDs to their respective classes
image_classes = dict(zip(class_labels['patientId'], class_labels['class']))

# Step 2: Display sample mappings to verify
print("Training and testing images mapped to their classes.\n")
print("Sample mappings:")

# Show the first 5 mappings as a sample
sample_mappings = list(image_classes.items())[:5]
for patient_id, class_label in sample_mappings:
    print(f"Patient ID: {patient_id} => Class: {class_label}")
```

```
Training and testing images mapped to their classes.

Sample mappings:
Patient ID: 0004cfab-14fd-4e49-80ba-63a80b6bddd6 => Class: No Lung Opacity / Not Normal
Patient ID: 00313ee0-9eaa-42f4-b0ab-c148ed3241cd => Class: No Lung Opacity / Not Normal
Patient ID: 00322d4d-1c29-4943-afc9-b6754be640eb => Class: No Lung Opacity / Not Normal
Patient ID: 003d8fa0-6bf1-40ed-b54c-ac657f8495c5 => Class: Normal
Patient ID: 00436515-870c-4b36-a041-de91049b9ab4 => Class: Lung Opacity
```

Dealing with DICOM Images: : DICOM images are typically used for storing medical images and are rich in information. Alongside the image data, they include important patient details such as the patient's name, age, sex, and the physician's name, among others.

To preview DICOM images without extracting any information, you can utilize the following code. First, make sure to install the pydicom Python package by running pip install pydicom.

```
import pydicom
sample_patientId = train_labels['patientId'][0]
dcm_file = '/content/drive/MyDrive/zipOut/stage_2_train_images/'+'{}.dcm'.format(sample_patientId)
dcm_data = pydicom.dcmread(dcm_file)

print('Metadata of the image consists of \n', dcm_data)
```

Here is the META Data from sample DICOM image file

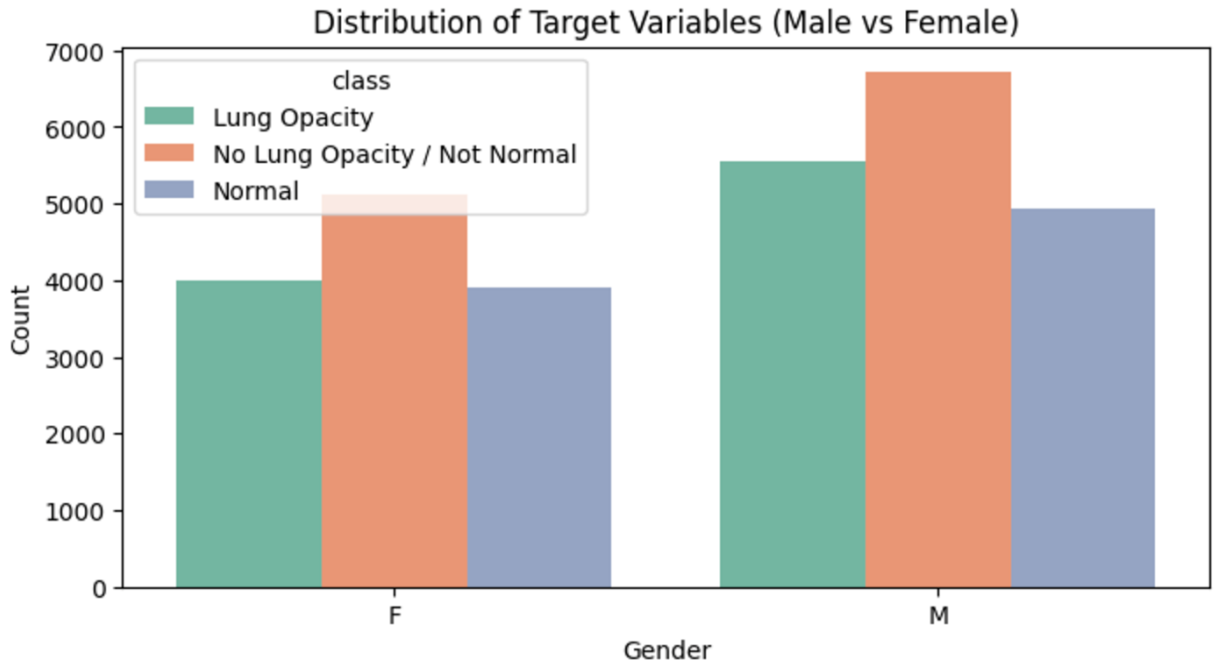
```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 202
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: Secondary Capture Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0002, 0010) Transfer Syntax UID                 UI: JPEG Baseline (Process 1)
(0002, 0012) Implementation Class UID            UI: 1.2.276.0.7230010.3.0.3.6.0
(0002, 0013) Implementation Version Name         SH: 'OFFIS_DCMTK_360'
-------------------------------------------------
(0008, 0005) Specific Character Set              CS: 'ISO_IR 100'
(0008, 0016) SOP Class UID                       UI: Secondary Capture Image Storage
(0008, 0018) SOP Instance UID                    UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0008, 0020) Study Date                          DA: '19010101'
(0008, 0030) Study Time                          TM: '000000.00'
(0008, 0050) Accession Number                    SH: ''
(0008, 0060) Modality                            CS: 'CR'
(0008, 0064) Conversion Type                     CS: 'WSD'
(0008, 0090) Referring Physician's Name          PN: ''
(0008, 103e) Series Description                  LO: 'view: PA'
(0010, 0010) Patient's Name                      PN: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0020) Patient ID                          LO: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0030) Patient's Birth Date                DA: ''
(0010, 0040) Patient's Sex                       CS: 'F'
(0010, 1010) Patient's Age                       AS: '51'
(0018, 0015) Body Part Examined                  CS: 'CHEST'
(0018, 5101) View Position                       CS: 'PA'
(0020, 000d) Study Instance UID                  UI: 1.2.276.0.7230010.3.1.2.8323329.28530.1517874485.775525
(0020, 000e) Series Instance UID                 UI: 1.2.276.0.7230010.3.1.3.8323329.28530.1517874485.775524
(0020, 0010) Study ID                            SH: ''
(0020, 0011) Series Number                       IS: '1'
(0020, 0013) Instance Number                     IS: '1'
(0020, 0020) Patient Orientation                 CS: ''
(0028, 0002) Samples per Pixel                   US: 1
(0028, 0004) Photometric Interpretation          CS: 'MONOCHROME2'
(0028, 0010) Rows                                US: 1024
(0028, 0011) Columns                             US: 1024
(0028, 0030) Pixel Spacing                       DS: [0.14300000000000002, 0.14300000000000002]
(0028, 0100) Bits Allocated                      US: 8
(0028, 0101) Bits Stored                         US: 8
(0028, 0102) High Bit                            US: 7
(0028, 0103) Pixel Representation                US: 0
(0028, 2110) Lossy Image Compression             CS: '01'
(0028, 2114) Lossy Image Compression Method      CS: 'ISO_10918_1'
(7fe0, 0010) Pixel Data                          OB: Array of 142006 elements
```

Visualize Data:

Distribution of Target Variables Male vs Female:

Within Male and Female group, the rate of Pneumonia risk is more in Male compared to Female.

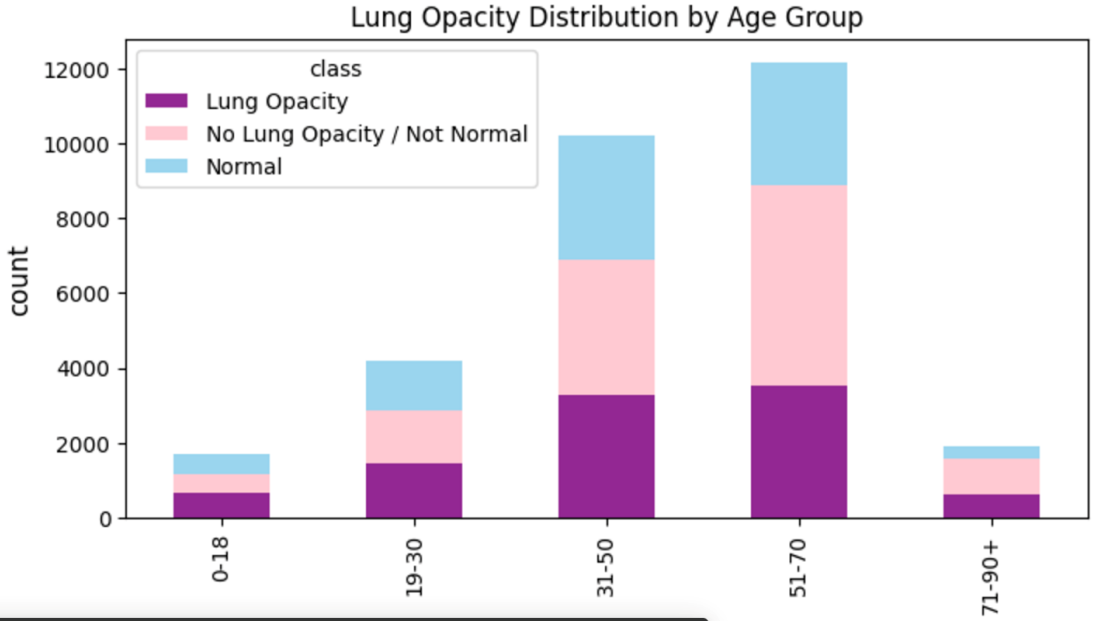
```
Distribution of Target variable by Gender:
class       Lung Opacity  No Lung Opacity / Not Normal  Normal
PatientSex
F                   3995                          5111    3905
M                   5560                          6710    4946
```



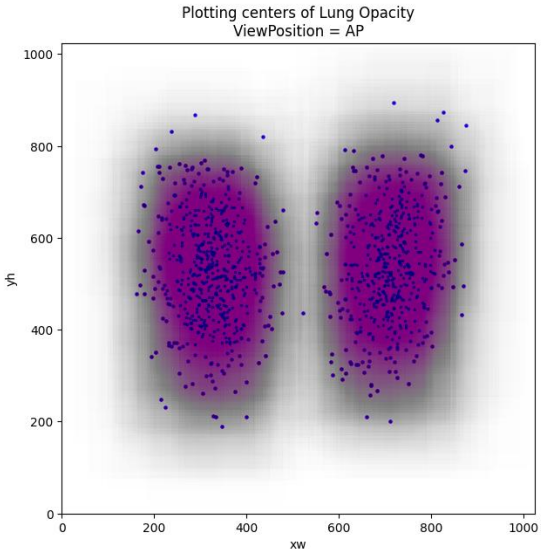
Distribution of Age vs Lung Opacity:

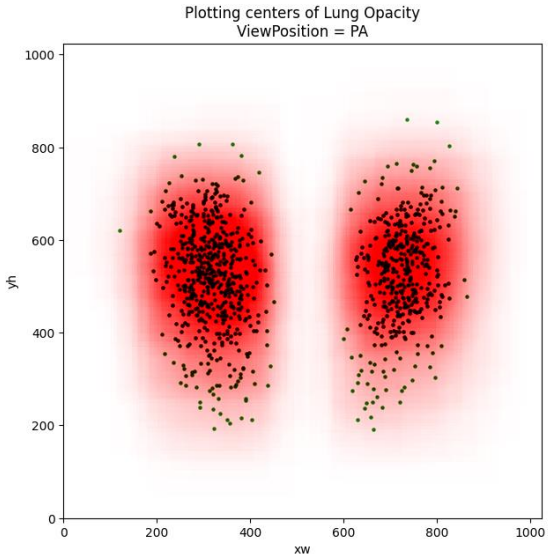
- Lung opacity is more common in older age groups, particularly in the 51-70 age range, suggesting an age-related increase in prevalence or risk.
- Younger age groups (0-18 and 19-30) are predominantly "Normal," indicating lower susceptibility or fewer cases of lung opacity.
- As age increases, the proportion of "Lung Opacity" cases grows, while the proportion of "Normal" cases decreases.
- The "No Lung Opacity / Not Normal" class remains relatively stable across all age groups.
- Healthcare Implications: Screening and preventative measures should target middle-aged and older adults (31-70+) due to higher prevalence of lung opacity in these groups.

```
Distribution of Age vs Lung Opacity:
class     Lung Opacity  No Lung Opacity / Not Normal  Normal
AgeGroup
0-18               678                           490     530
19-30             1451                          1395    1363
31-50             3266                          3631    3325
51-70             3539                          5361    3276
71-90+             621                           944     357
```



Centers of Lung Opacity AP vs PA:





- Body Part examined is unique for all cases and is CHEST in the training dataset and that was also expected.
- Unique in Modality is CR i.e. Computer Radiography
- Overall View Position is almost equally distributed in the training dataset but for cases where. Target=1, most of the view position are AP.

Step 4: Build the Model:

Model Architecture:

- **Input Dimensions:**
  - Input Shape: (128, 128, 3) (image size is 128×128128 \times 128128×128 with RGB channels).
  - Output Classes: 3 (classification targets).
- **Architecture Overview:**
  - **Convolutional Layers:**
    - 3 convolutional layers with ReLU activation:
      - First layer: 323232 filters of size 3×33 \times 33×3.
      - Second layer: 646464 filters of size 3×33 \times 33×3.
      - Third layer: 128128128 filters of size 3×33 \times 33×3.
    - Each convolutional layer is followed by:
      - **Batch Normalization**: Stabilizes and accelerates training.
      - **MaxPooling**: Down samples feature maps by 2×22 \times 22×2.
  - **Flatten Layer:**
    - Converts the 3D output from the convolutional layers into a 1D vector.
  - **Dense (Fully Connected) Layers:**
    - Two fully connected layers with 128128128 neurons and ReLU activation.
    - Dropout (50%): Applied to both dense layers to mitigate overfitting.

- **Output Layer:**
  - Dense layer with 333 neurons (for the 3 classes) and Softmax activation to output class probabilities.

- **Compilation Details:**
  - Optimizer: **Adam** (adaptive learning rate optimization).
  - Loss Function: **Categorical Crossentropy** (suitable for multi-class classification).
  - Metric: **Accuracy**.

Model Summary:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
batch_normalization (BatchNormalization)	(None, 126, 126, 32)	128
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 61, 61, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 28, 28, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 256)	2,359,552
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 3)	387

Total params: 2,782,147 (10.61 MB)
Trainable params: 2,781,699 (10.61 MB)
Non-trainable params: 448 (1.75 KB)

Overview of Model Building

Base Model:

- Use a pre-trained model (e.g., ResNet50, VGG16) for transfer learning.
- Freeze initial layers of the base model and fine-tune higher layers to adapt to the pneumonia detection task.

2. **Custom Model**:
   - Add task-specific layers like Dense layers for classification or regression.
   - Example:

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(512, 512, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

3. **Compile Model**:
   - Use:
     - **Optimizer**: Adam (adaptive learning rate).
     - **Loss Function**: Binary cross-entropy for binary classification.
     - **Metrics**: Accuracy for performance tracking.

Step 6: Train the Model

1. **Fit the Model**:
   - Train the model using the training dataset while validating on the validation dataset.
   - Example:

```
history = model.fit(
    train_dataset,
    validation_data=val_dataset,
```

```
    epochs=5
)
```

2. **Monitor Training**:
   - Use callbacks like EarlyStopping to stop training when validation loss stops improving.

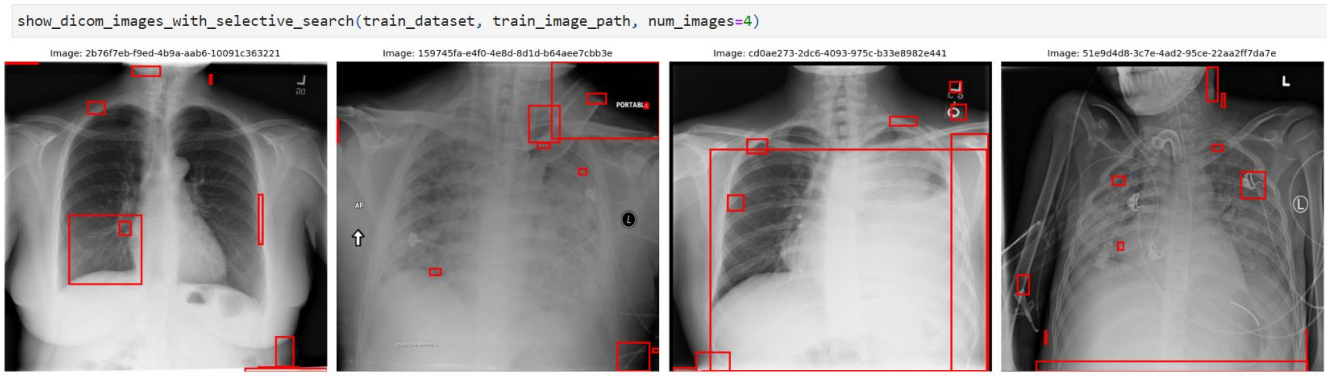
Step 7: Evaluate the Model

1. **Test Dataset**:
   - Evaluate the model's performance on unseen test data.
   - Metrics include accuracy, precision, recall, and F1 score.

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967
3	VGG16 Model	0.842660	0.700992	0.685487
4	Resnet Model	0.865488	0.681147	0.674680

2. **Object Detection (YOLO and RCNN)**

RCNN FOR OBJECT DETECTION

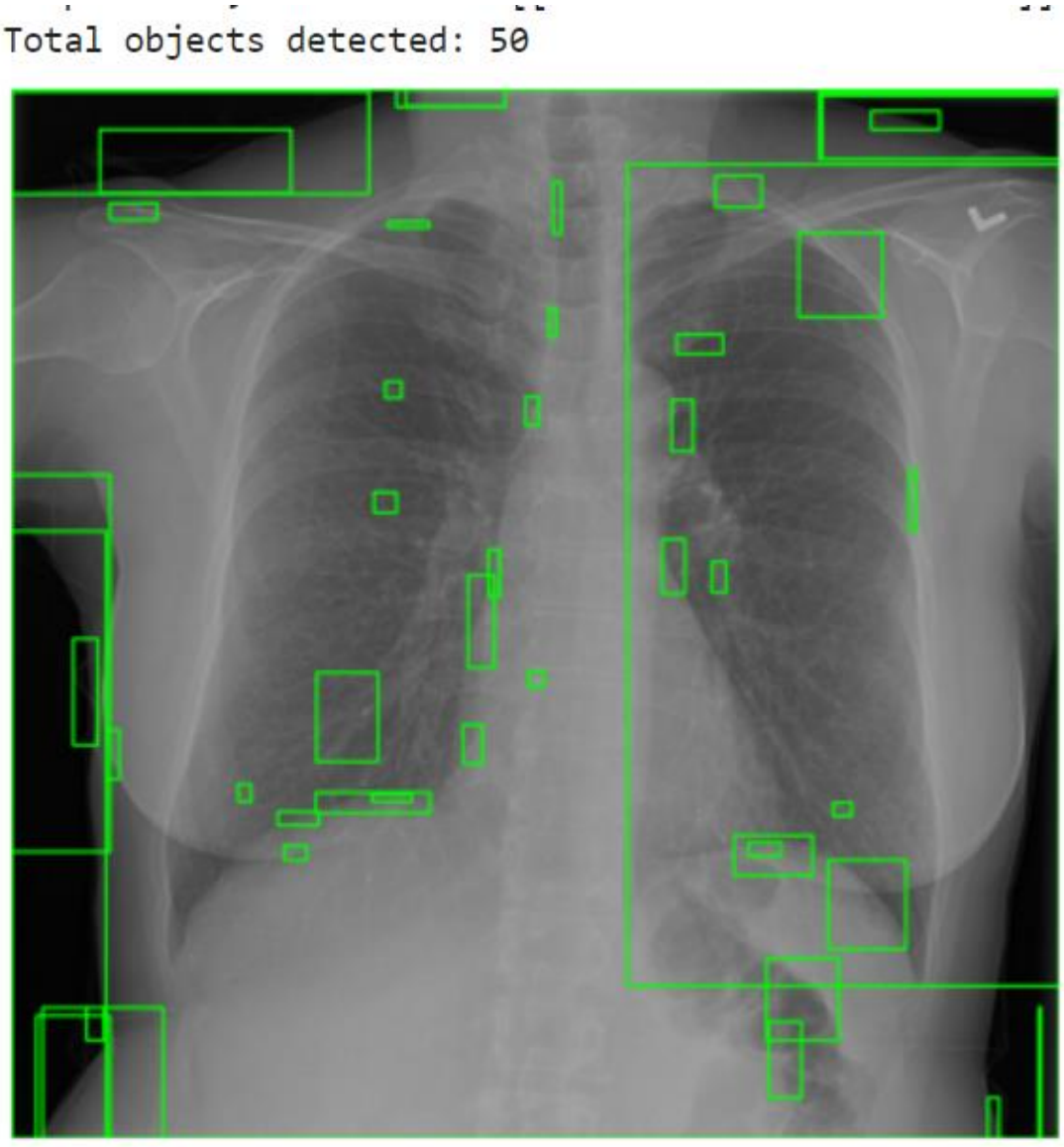


Intersection over union

Intersection over Union is simply an evaluation metric. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU. More formally, in order to apply Intersection over Union to evaluate an (arbitrary) object detector we need:

1.The ground-truth bounding boxes (i.e., the hand labeled bounding boxes from the testing set that specify where in the image our object is).

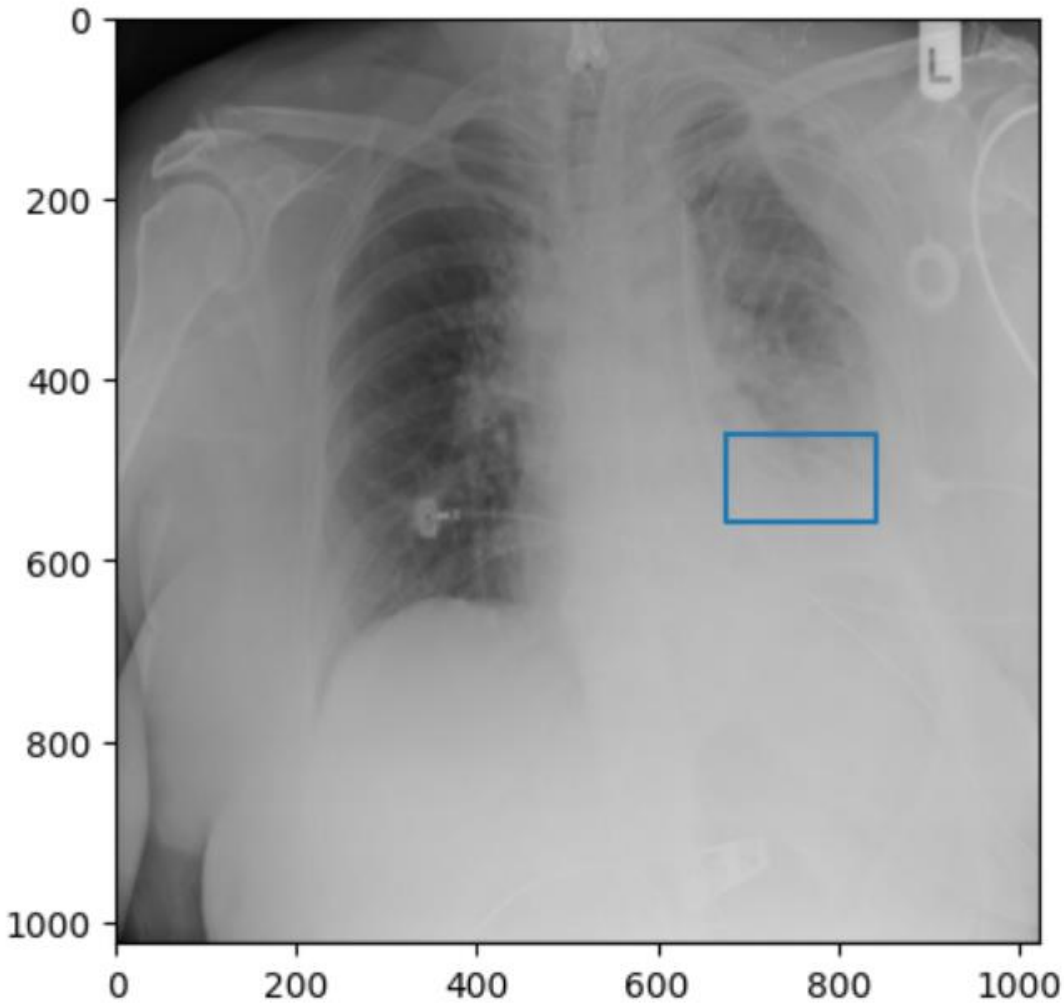
2.The predicted bounding boxes from our model. After getting these two we can find there intersection and union area,whih ultimately gives us our IoU.



YOLO FOR OBJECT DETECTION

```
demo_patient_id = val_series.values[8]
demo_img_path = DATASET_DIR + 'images/val/' + demo_patient_id + '.jpg'
demo_label_path = DATASET_DIR + 'labels/val/' + demo_patient_id + '.txt'

plt.imshow(cv2.imread(demo_img_path))

with open(demo_label_path, "r") as f:
    for line in f:
        print(line)
        class_id, rx, ry, rw, rh = list(map(float, line.strip().split()))

        x, y, w, h = revert_bbox(rx, ry, rw, rh)
        plt.plot([x, x, x+w, x+w, x], [y, y+h, y+h, y, y])
```

```
0 0.7412109375 0.49658203125 0.162109375 0.0966796875
```



```
# Access and print metrics from the validation results
print(f"Precision: {results.box.p.mean():.2f}")
print(f"Recall: {results.box.r.mean():.2f}")
print(f"F1 Score: {results.box.f1.mean():.2f}")
```

```
Precision: 0.51
Recall: 0.58
F1 Score: 0.54
```

Step 8: Build the GUI

1. **User Interface**:
   - Design a simple GUI using a framework like Tkinter.
   - Features:
     - Upload chest radiographs.
     - Display prediction (Pneumonia detected or not).
     - Highlight regions of interest (bounding box).
2. **Integrate Model**:
   - Embed the trained model into the GUI for real-time inference.

Step 9: Analyze Results

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967
3	VGG16 Model	0.842660	0.700992	0.685487
4	Resnet Model	0.865488	0.681147	0.674680

SECTION 4: MODEL EVALUATION:

- VGG16 and ResNet50 outperformed other models, with VGG16 slightly better.
- Fine-tuned models showed improved generalization compared to base models that overfitted the training data.
- For object detection, RCNN and YOLO are evaluated for bounding box accuracy.

Model Architecture:

Step 1: Fine tune the trained basic CNN models for classification.

CNN Base Model:
Total params: 3,305,027 (12.61 MB)
Trainable params: 3,305,027 (12.61 MB)
Non-trainable params: 0 (0.00 B)

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218

Fine tune Model:
Total params: 3,305,027 (12.61 MB)
Trainable params: 3,305,027 (12.61 MB)
Non-trainable params: 0 (0.00 B)

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967

Step 2: Apply Transfer Learning model for classification

VGG 16 Model:

Total params: 15,763,779 (60.13 MB)

Trainable params: 1,049,091 (4.00 MB)
Non-trainable params: 14,714,688 (56.13 MB)

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967
3	VGG16 Model	0.842660	0.700992	0.685487

Res Net Model:

Total params: 24,638,339 (93.99 MB)
Trainable params: 1,050,627 (4.01 MB)
Non-trainable params: 23,587,712 (89.98 MB)

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967
3	VGG16 Model	0.842660	0.700992	0.685487
4	Resnet Model	0.865488	0.681147	0.674680

Insights :

- **VGG16** and **ResNet** are the best-performing models in terms of generalization to both validation and test sets, with **VGG16** slightly outperforming **ResNet**.
- **Finetuned Model** is a good intermediate model, with decent performance, especially considering it has lower training accuracy than the **Base Model**, which suggests better generalization.
- **Base Model** is overfitting the training data, as evidenced by its significant drop in validation/testing accuracy.
- **Milestone 1 Model** shows lower performance across the board, but this could be due to its simpler architecture compared to the more complex models (like **VGG16** and **ResNet**).

Recommendations:

1. **VGG16 and ResNet** are the best models to consider for deployment, with **VGG16** showing a slight edge.
2. **Finetuned Model** is a good balance between performance and generalization, and its performance on the validation/test sets is relatively stable.
3. For **Base Model**:
    - Regularization techniques like dropout or L2 regularization can help reduce overfitting and improve generalization.

SECTION 5: BENCHMARK

The **benchmark** at the outset of this project was based on evaluating the performance of existing models, especially in the context of pneumonia detection from chest radiographs. The goal was to identify how well the model could detect pneumonia with both **classification** and **object detection** capabilities, while also assessing its ability to generalize across new, unseen data.

Test Results Comparison

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.74317	0.61742	0.604543

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967

	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967
3	VGG16 Model	0.842660	0.700992	0.685487

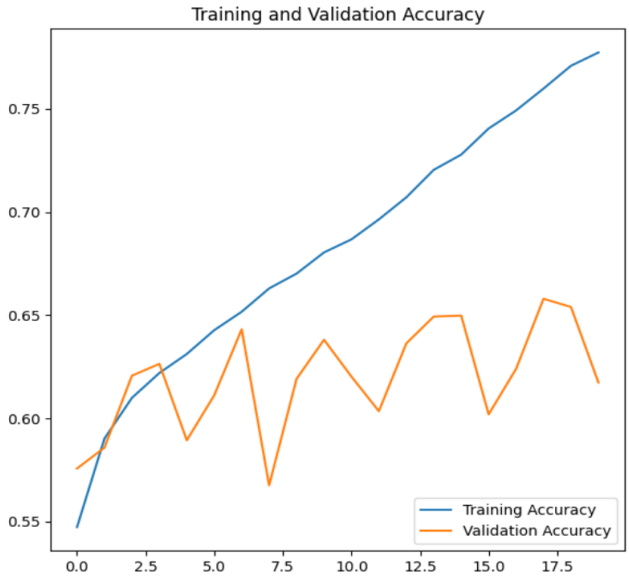
	Model	Training Accuracy	Validation Accuracy	Testing Accuracy
0	Milestone 1 Model	0.743170	0.617420	0.604543
1	Base Model	0.960913	0.632856	0.618218
2	Finetuned Model	0.903063	0.627784	0.621967
3	VGG16 Model	0.842660	0.700992	0.685487
4	Resnet Model	0.865488	0.681147	0.674680

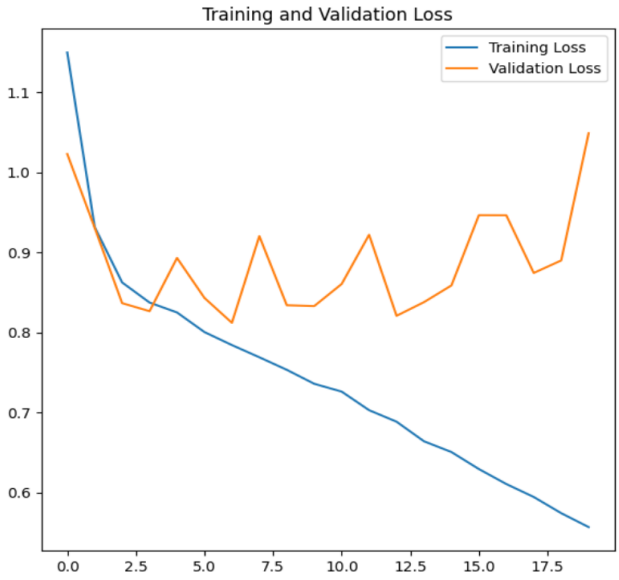
```
# Access and print metrics from the validation results
print(f"Precision: {results.box.p.mean():.2f}")
print(f"Recall: {results.box.r.mean():.2f}")
print(f"F1 Score: {results.box.f1.mean():.2f}")
```

```
Precision: 0.51
Recall: 0.58
F1 Score: 0.54
```

SECTION 6: VISUALISATIONS

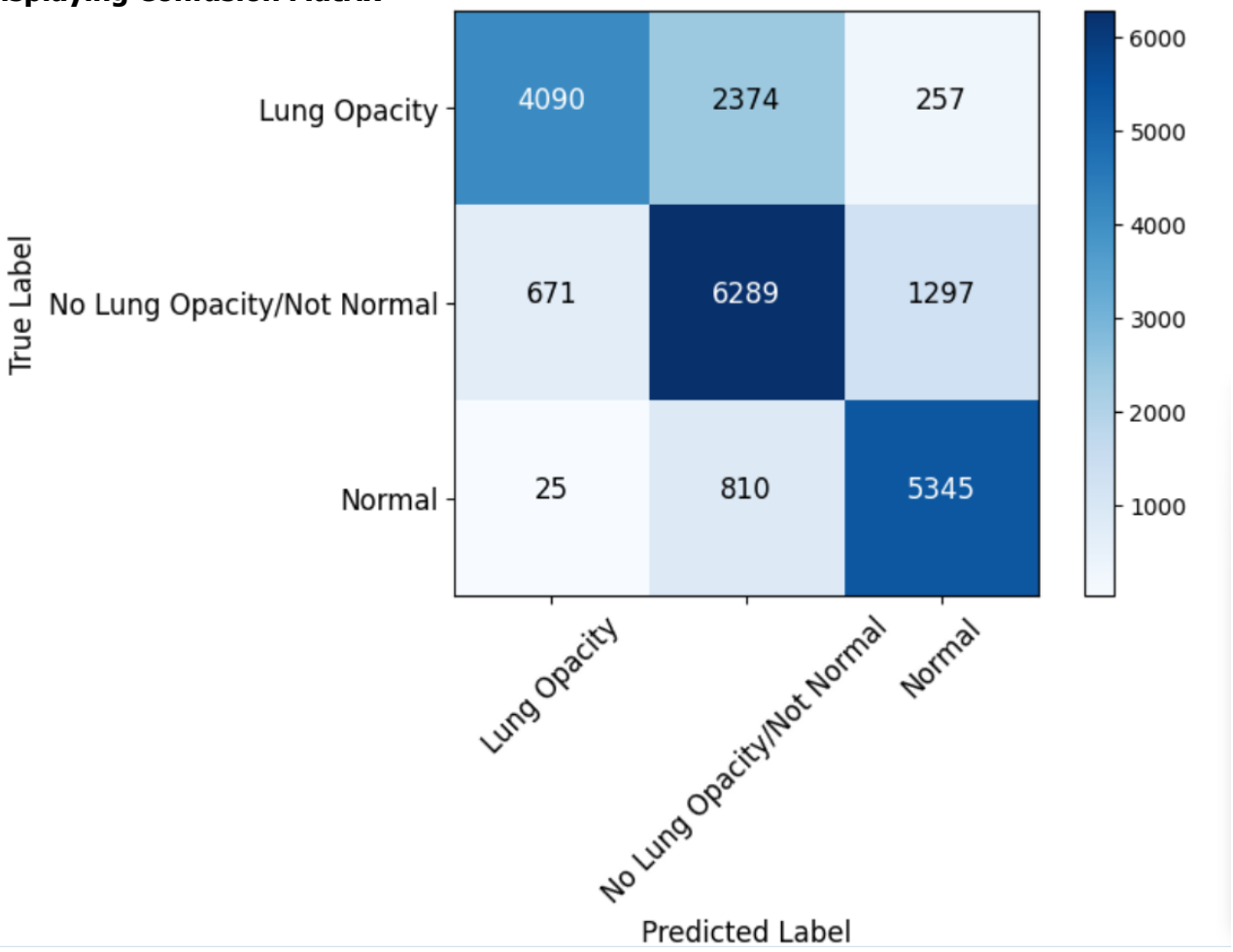
Displaying Training and Validation Accuracy as well as Loss.





Generating predictions on training set...

Displaying Confusion Matrix

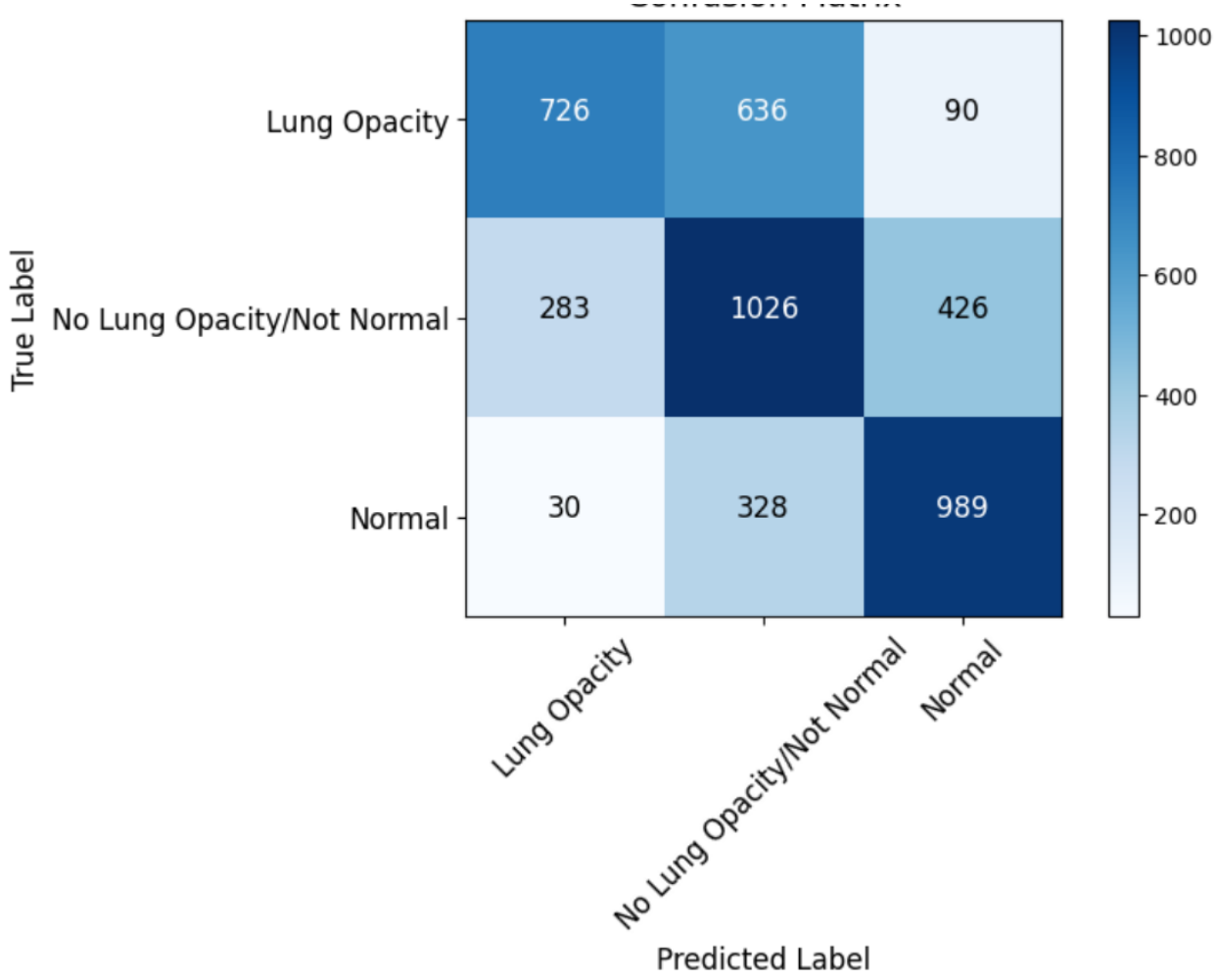


Displaying Classification Report:

```
Classification Report:

                           precision    recall  f1-score   support

              Lung Opacity       0.85      0.61      0.71      6721
No Lung Opacity/Not Normal       0.66      0.76      0.71      8257
                    Normal       0.77      0.86      0.82      6180

                  accuracy                           0.74     21158
                 macro avg       0.76      0.75      0.75     21158
              weighted avg       0.76      0.74      0.74     21158
```

Generating predictions on test set...

Displaying Confusion Matrix

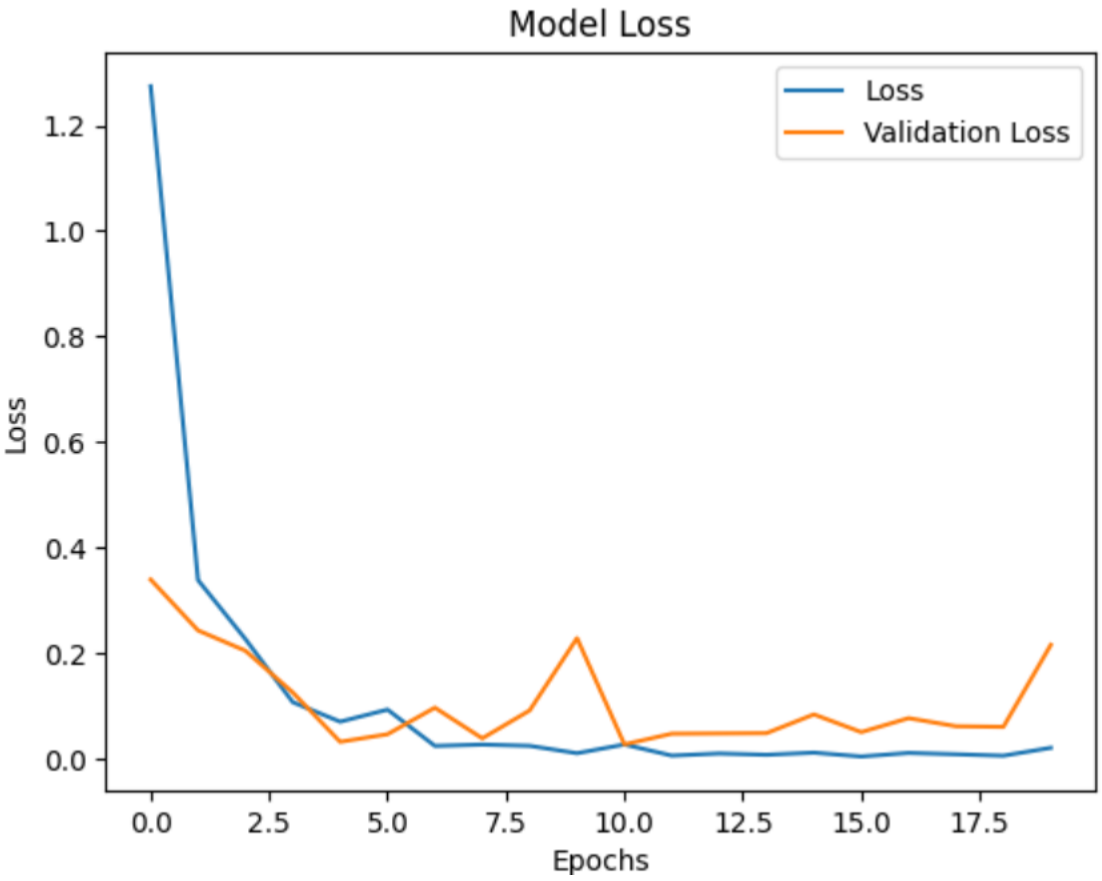


Displaying Classification Report:

Classification Report:

	precision	recall	f1-score	support
Lung Opacity	0.70	0.50	0.58	1452
No Lung Opacity/Not Normal	0.52	0.59	0.55	1735
Normal	0.66	0.73	0.69	1347
accuracy			0.60	4534
macro avg	0.62	0.61	0.61	4534
weighted avg	0.62	0.60	0.60	4534

Displaying Model Loss and Model Accuracy



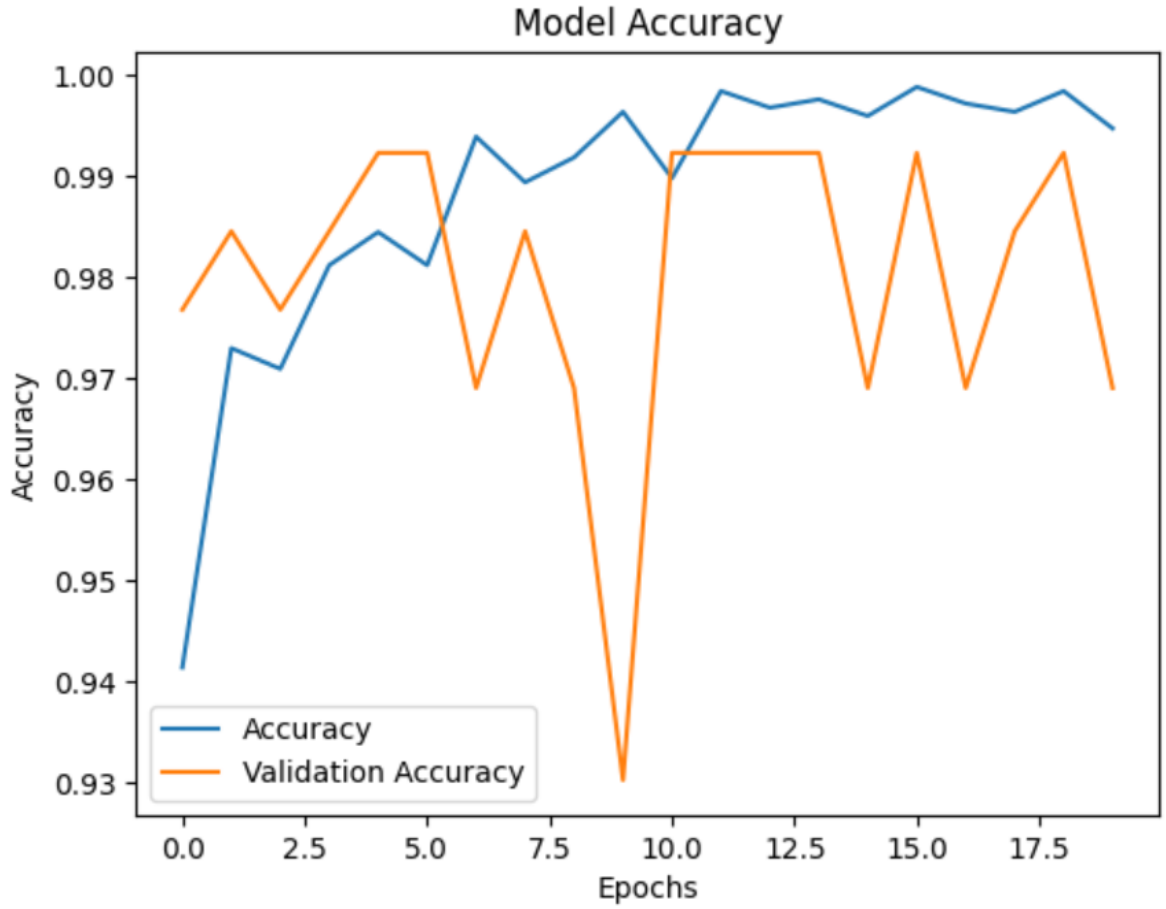


Image 1: PR_curve.png

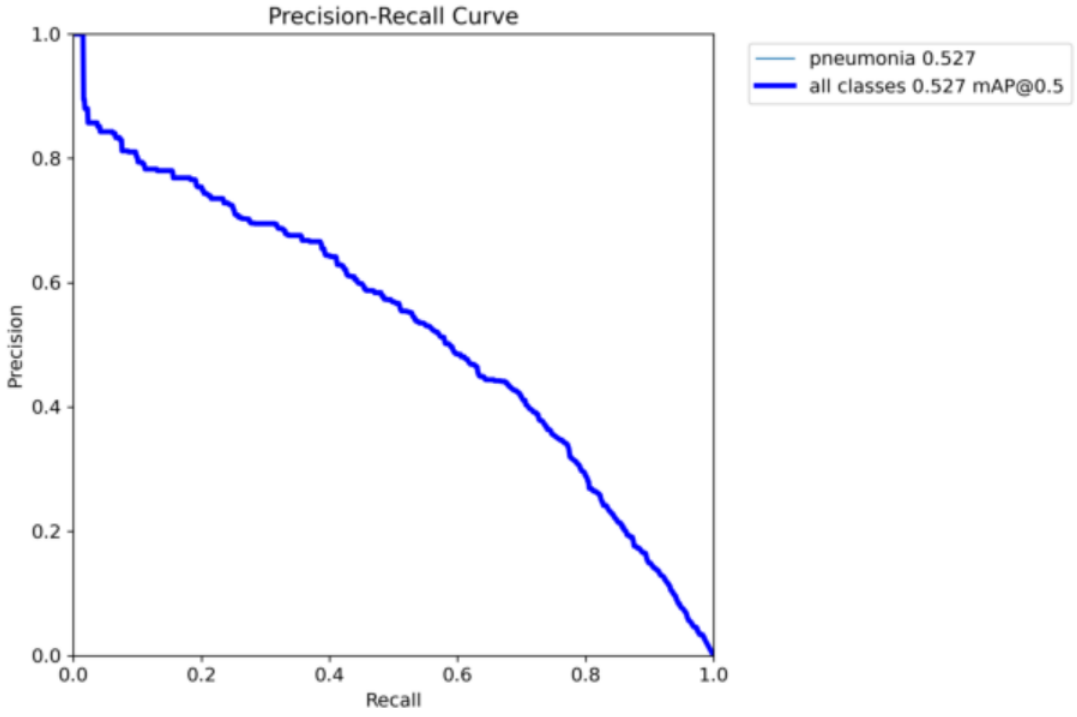


Image 2: R_curve.png

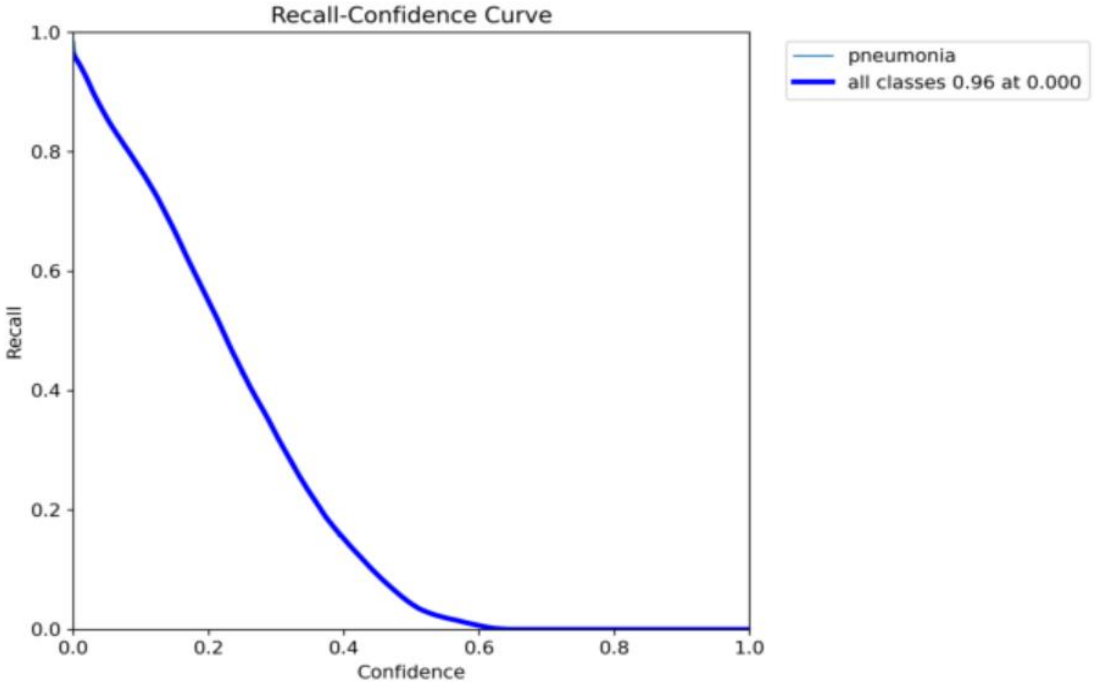


Image 3: P_curve.png

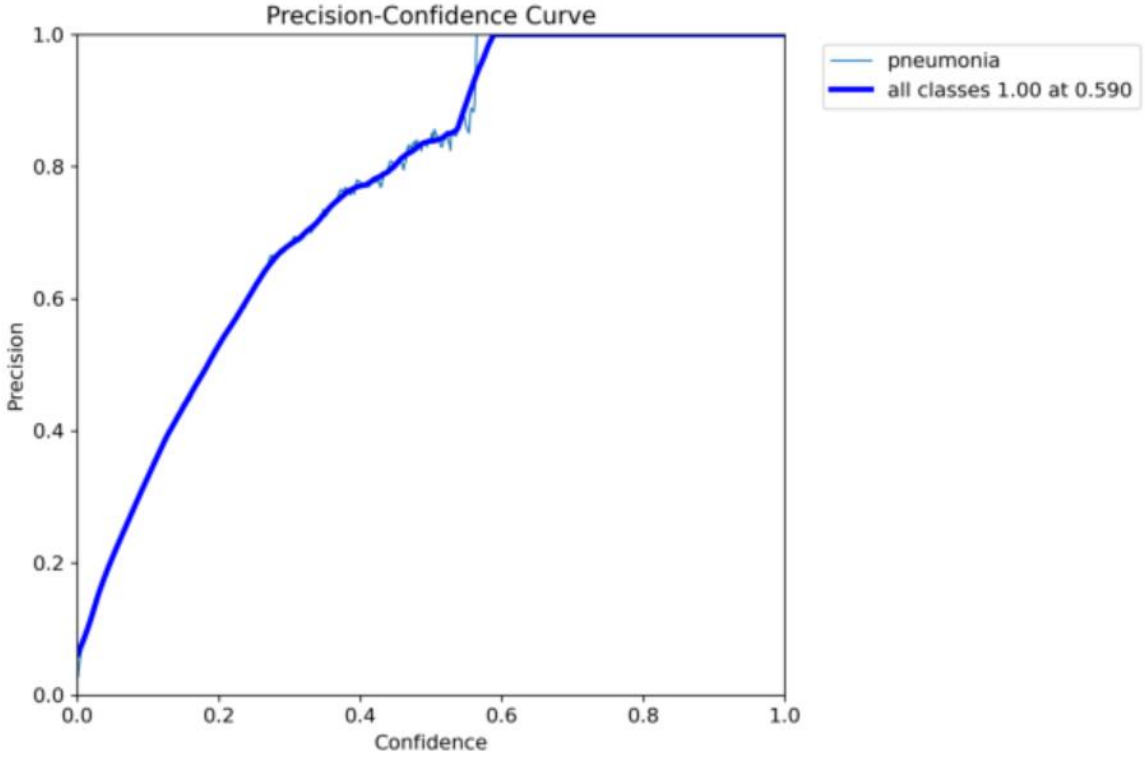


Image 4: confusion_matrix_normalized.png

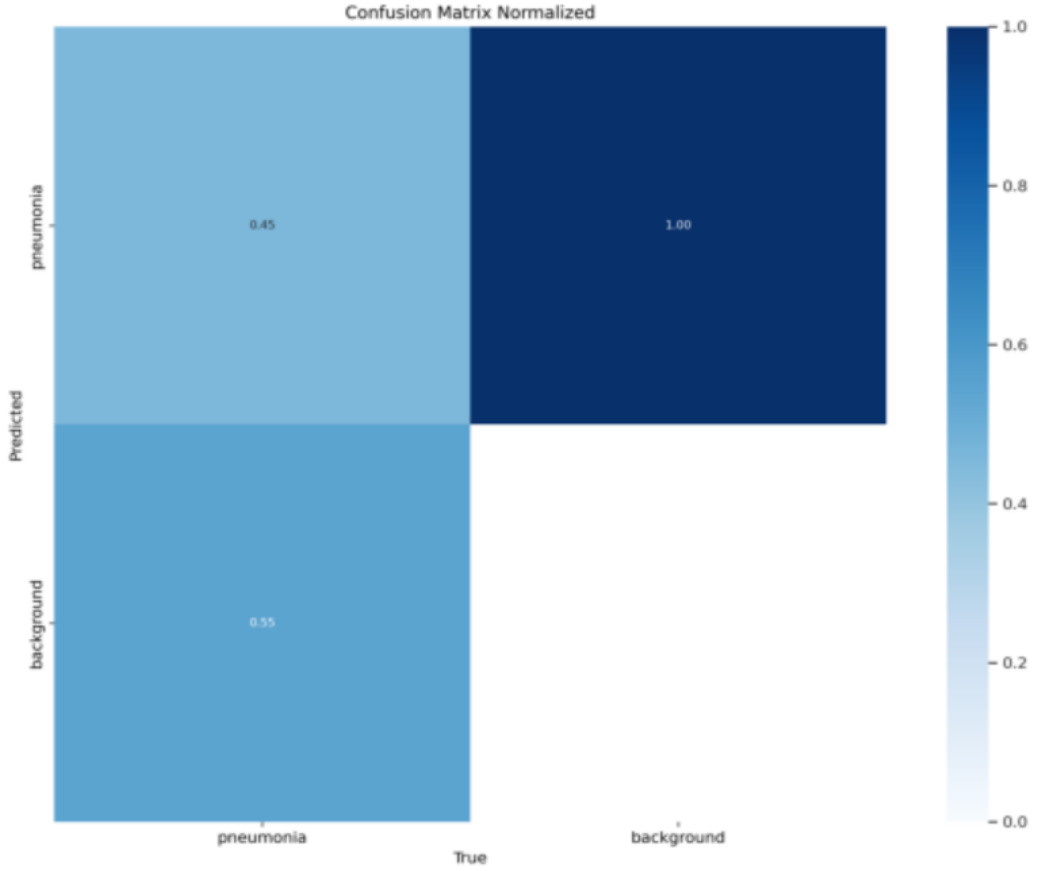


Image 5: F1_curve.png

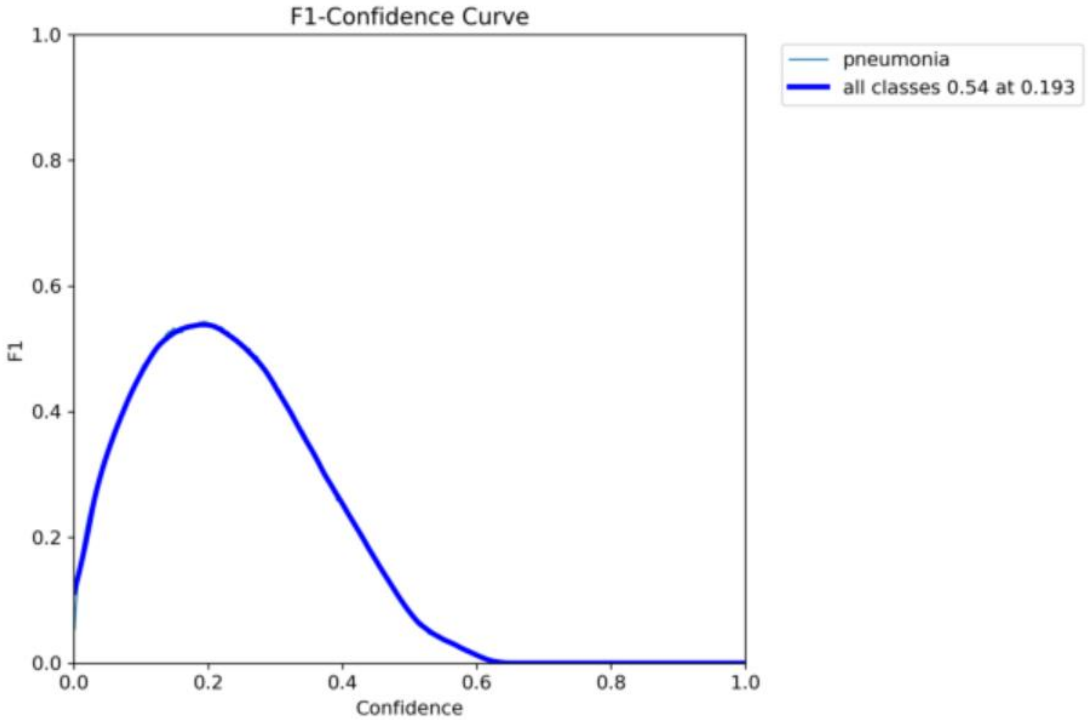
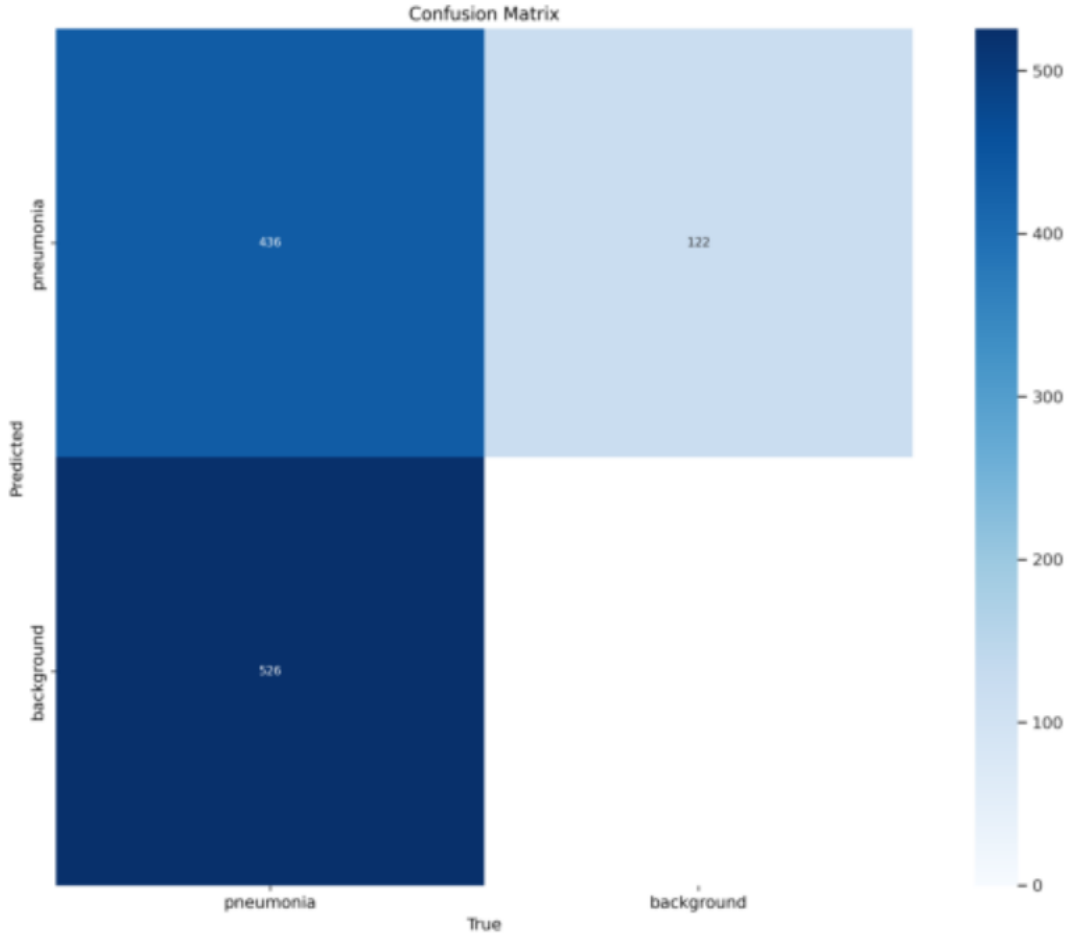


Image 6: confusion_matrix.png



SECTION 7: IMPLICATIONS

Implications of the Solution

The solution developed for pneumonia detection using deep learning techniques has significant implications in both the healthcare domain and business contexts.

Impact on Healthcare:

1.Early Detection:

Automation of Diagnosis: The proposed model automates the detection of pneumonia from chest radiographs, significantly reducing the workload of radiologists. By accurately identifying lung opacities, it aids in the early detection of pneumonia, which is crucial for improving patient outcomes.

Faster Diagnosis: With real-time or faster processing capabilities, this model allows healthcare providers to make quicker decisions, particularly in high-demand areas where radiologist availability may be limited.

Reduced Human Error: By leveraging the precision of deep learning models, this system reduces human error, ensuring more consistent diagnoses across different healthcare settings.

2.Scalability:

Wide Adoption in Healthcare Systems: The model can be integrated into healthcare systems worldwide, especially in areas with limited access to qualified medical professionals. This helps in ensuring that even remote and underserved populations receive high-quality medical attention.

Cost Efficiency: With the ability to process thousands of images quickly and consistently, this solution can significantly lower costs associated with manual image analysis and diagnosis.

Business Implications:

1.Productization of the Solution:

This solution could be turned into a commercial product for hospitals, clinics, or diagnostic centers. By integrating this deep learning model into existing medical imaging software, businesses can offer a cutting-edge tool to the healthcare industry.

Revenue Opportunities: Companies providing medical imaging solutions could enhance their product offerings by integrating automated pneumonia detection, making them more competitive.

2.Competitive Advantage:

Hospitals adopting this technology can offer faster diagnostic services, providing a competitive advantage in patient care. The ability to make faster and more accurate diagnoses could improve hospital ratings and patient satisfaction.

SECTION 8. LIMITATIONS OF THE SOLUTION

While the model provides promising results, there are inherent limitations that need to be addressed to ensure its success in real-world applications.

Limitations:

1.Data Imbalance:

Class Imbalance: The dataset may be skewed toward certain types of pneumonia or other conditions, which can lead to biased predictions. If the model has not been exposed to a representative distribution of cases, it might not generalize well to unseen cases.

1.Generalization:

Dataset Bias: The model is trained on specific datasets and might not generalize well to other populations or geographical areas. Chest radiographs from regions with different pneumonia types or from different equipment may cause the model to underperform.

Age and Gender Factors: If the dataset is not diverse in terms of patient age and gender, the model might fail to recognize pneumonia effectively in certain subpopulations, impacting diagnostic equity.

2.False Positives and False Negatives:.

Bounding Box Precision: In the case of object detection (bounding boxes), the model might not always locate pneumonia regions accurately, particularly in cases where the opacities are small or difficult to distinguish from surrounding tissues.

3.Interpretability of Results:

Black Box Nature: Deep learning models, particularly CNNs, can often be considered "black boxes," meaning their decision-making process is not easily interpretable by healthcare professionals. This makes it harder to trust or explain the model’s predictions, especially in critical clinical situations.

SECTION 9: REFLECTIONS

Model classification:

The Base Model has the highest training accuracy but suffers from overfitting, as seen in the drop in validation and testing accuracy.
The Finetuned Model shows slight improvements over the Base Model, suggesting that finetuning mitigates overfitting to some extent.
The VGG16 Model achieves the best validation and testing accuracy, indicating it generalizes better than the others and is the most promising for deployment.

R-CNN object detection
R-CNN object detection system consists of three modules. The first generates category-independent region proposals(through selective search here). These proposals define the set of candidate detections available to our detector. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region(here we use VGG 16 whereas original paper works on AlexNet). The third module is a set of class specific linear SVMs. So, let's implient them step by step.

1. Selective search

Selective Search is a region proposal algorithm used in object detection. It is designed to be fast with a very high recall. It is based on computing hierarchical grouping of similar regions based on color, texture, size and shape compatibility.

Intersection over union

Intersection over Union is simply an evaluation metric. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU.
More formally, in order to apply Intersection over Union to evaluate an (arbitrary) object detector we need:
1.The ground-truth bounding boxes (i.e., the hand labeled bounding boxes from the testing set that specify where in the image our object is).
2.The predicted bounding boxes from our model. After getting these two we

can find there intersection and union area,which ultimately gives us our IoU.

Preparing training data
Training data here is prepared according to following scheme:-
a) Those boxes which have an IoU greater than 0.7 are considered as a positive example.
b) And boxes with relative low IoU 0.3 are taken to be negative examples.
Number of regions taken here is 10 positive and 10 negative of size 224x224. (Trying out with less number of regions datas due to large time and memory consumption)

Pre-training
We perform tansfer-learning as pre-training here as we are using VGG16 net which is already trained on Imagenet. We are going to train a binary classifier by making all layers as untrainble except two last. And then we are going to make a new model which will be stacked on last 2nd output layer.

Yolo Object Detection :

Metric Analysis
Precision (0.51):

Indicates the proportion of true positive detections among all positive predictions.
A precision of 0.51 suggests that 51% of the detections are correct, while 49% are false positives.
Recall (0.58):

Represents the proportion of true positives detected among all actual positives.
A recall of 0.58 means the model identifies 58% of the actual objects but misses 42%.
F1 Score (0.54):

The harmonic mean of precision and recall.
An F1 score of 0.54 indicates a trade-off between precision and recall but points to room for improvement in overall model performance.
Potential Issues
False Positives: Precision being relatively low indicates that the model is over-

predicting objects.
Missed Detections: Recall suggests the model struggles to detect all objects in the dataset.

Class Imbalance: If the dataset is imbalanced across object classes, some classes may dominate training, affecting minority classes.
Dataset Quality: The dataset may contain noisy labels, occlusions, or insufficient examples for certain object types.

Recommendations for Improvement

1. Data Enhancements
Increase Dataset Size: Add more labeled examples, especially for underrepresented classes.
Data Augmentation:
Use transformations like scaling, flipping, rotation, and random cropping to increase diversity.
Adjust brightness, contrast, and color for better robustness in varied conditions.
Synthetic Data: Generate synthetic images using tools like GANs or data synthesis frameworks.

2. Model Improvements
Anchor Box Tuning: Optimize the anchor box sizes and aspect ratios for the object sizes in your dataset.
Loss Function: Experiment with focal loss or other loss functions to better handle class imbalance.
Higher-Resolution Inputs: Use higher-resolution images to capture finer details, if computationally feasible.

3. Training Strategies
Hard Negative Mining: Focus on challenging samples where the model performs poorly.
Class-Balanced Sampling: Ensure each class is equally represented during training to mitigate bias.
Transfer Learning: Use pre-trained weights fine-tuned on your dataset to leverage prior knowledge.

4. Evaluation Adjustments

Confidence Threshold: Lower the confidence threshold for detecting objects to balance precision and recall.
Non-Maximum Suppression (NMS): Fine-tune the IoU threshold in NMS to

reduce false positives without missing true positives.

The ResNet Model performs better than the Milestone 1, Base, and Finetuned models, with strong validation and testing performance but slightly behind the VGG16 Model.
The VGG16 Model remains the top performer in terms of validation and testing accuracy, indicating the best generalization to unseen data.