AMIA Project Report

Namrata De
Masters, Data Science, Free University Berlin
Matrikelnummer: 5580793

Computer Vision for Biomedical Images
Summer Semester 2024

Abstract

The aim of the AMIA challenge [1] is to use chest radiographs to identify and categorise thoracic anomalies. This issue holds significance for medical imaging because it helps radiologists diagnose different lung disorders more efficiently. A total of 15,000 chest X-ray scans — 8,573 for training and 6,427 for testing — annotated by experienced radiologists make up the challenge dataset. In addition to a "No finding" class to account for photos with no visible abnormalities, the dataset comprises 14 different categories of thoracic abnormalities. To train our model for this project, however, we only used a subset of the total number of available images.

Contents

1 Introduction

1.1 Goal

The project's objective is to classify thoracic abnormalities from images of chest radiographs. In this case, we have used a YOLO-based deep learning framework [2] for the said task. We expect the model to identify abnormalities in images with a bounding box around each instance, and classify their respective types (including a "No Finding" class, where the model does not recognize any abnormality in a given image).

1.2 Motivation

This initiative seems motivated by the need to improve diagnostic efficiency in radiology. By saving analysis time and increasing diagnostic precision, radiologists can be benefited greatly by automating the identification and categorisation of thoracic anomalies from chest X-rays. Overall, the project intends to improve the treatment of patients by addressing the difficulties in detecting subtle and variable anomalies by utilising advanced object detection techniques such as YOLOv8 [3]. The widespread use of AI-driven medical imaging systems may be aided by the effectiveness of this automation.

2 Methodology

2.1 Original Dataset

Source: AMIA Public Challenge 2024 in 'kaggle' [1].

Content: The dataset consists of 15,000 postero-anterior (PA) chest X-ray scans in PNG format, annotated for the presence of 14 types of thoracic abnormalities and a "No finding" class. Each image has a unique ID, helping in anonymising the dataset. The images are organized into training and testing sets, with the following additional files provided:

Files:

- **train.csv:** Metadata for the training set, containing one row per object with class IDs and bounding box coordinates. Some images may have multiple objects annotated with several rows for the same image.
- **test.csv:** Metadata for the test set, listing filenames of the test images without bounding box information.
- **img_size.csv:** Contains the original dimensions of the images, where `dim0` denotes the height and `dim1` denotes the width.
- **sample_submission.csv:** A template for submissions, similar to `test.csv`, with placeholder results.
- Training and testing images are located in the `train` and `test` folders, respectively.

Sample Images:

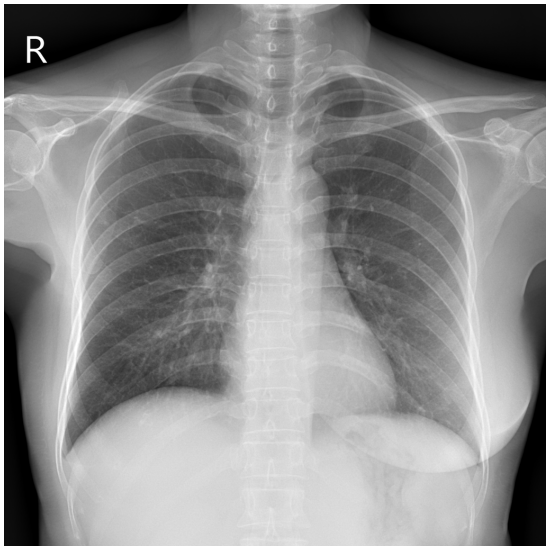

Figure 1: 0a2u28UKY8bZCWajCxzYjiRZfq32v7RM.png

Figure 2: 0aL2bmkIv3TILGLGXBqvOKiTrAaeSLcz.png

2.2 Anomaly Classification

The following are the types of thoracic abnormalities provided by the authors [1]:

- Aortic enlargement (Class 0)
- Atelectasis (Class 1)
- Calcification (Class 2)
- Cardiomegaly (Class 3)
- Consolidation (Class 4)
- ILD (Class 5)
- Infiltration (Class 6)
- Lung Opacity (Class 7)
- Nodule/Mass (Class 8)
- Other lesion (Class 9)
- Pleural effusion (Class 10)
- Pleural thickening (Class 11)
- Pneumothorax (Class 12)
- Pulmonary fibrosis (Class 13)
- No finding (Class 14)

From the distribution of classes (see Figure 3), it appears that some classes are less frequent than others, for example, "Atelectasis" and "Consolidation" have fewer instances compared to popular classes like "Aortic enlargement", "Pleural thickening" etc. Also, "No Finding", i.e. images without any thoracic abnormalities seemed to be frequent.

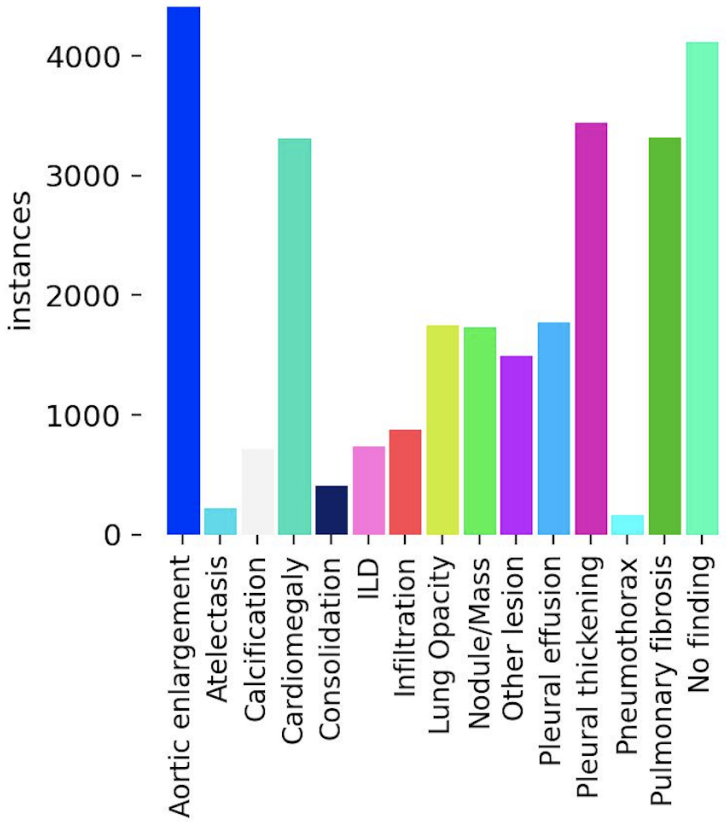

Figure 3: Distribution of Classes

2.3 YOLO Overview

YOLO (You Only Look Once) is a cutting-edge object detection algorithm, first introduced in 2016, [2] that processes images in a single pass, predicting bounding boxes and class probabilities simultaneously, allowing for real-time processing. We utilized YOLO v8 [3], released in January 2023. YOLOv8 offers five different scaled versions: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large). We have used YOLOv8n [3] with around 3M parameters, suitable for computational resource available to us.

YOLO Architecture:

- Input Layer: Receives a resized full image, often standardized to fixed dimensions like 448x448 pixels.
- Convolutional Layers: Extract features from the input through a series of convolutional layers, progressively downsampling spatial dimensions while increasing depth.

- Grid Division: The final feature map is divided into an S x S grid, with each grid cell responsible for predicting a certain number of bounding boxes.
- Bounding Box Predictions: Each grid cell predicts B bounding boxes, including coordinates (x, y), width and height (w, h), confidence score (P), and class probabilities (C).
- Anchor Boxes: Predefined bounding boxes with specific aspect ratios and sizes are used.
- Loss Function: Combines localization loss, confidence loss, and classification loss, ensuring a balance between precise localization, correct classification, and high confidence.
- Detection: During detection, the model outputs bounding box predictions, confidence scores, and class probabilities. Post-processing techniques like non-maximum suppression (NMS) are applied to refine predictions.

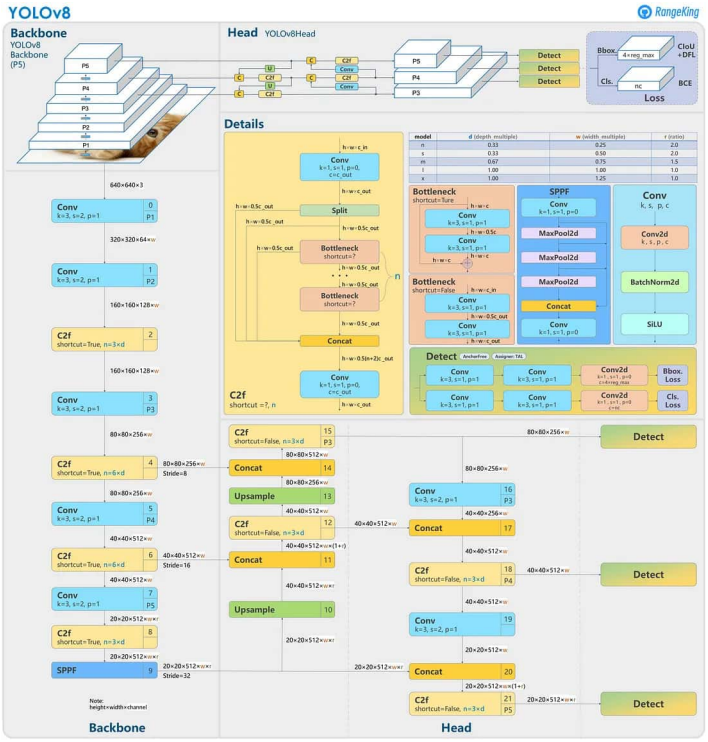

Figure 4: YOLOv8 Architecture, source: `https://github.com/ultralytics/ultralytics/issues/189`

Model Summary:

- **Total Layers:** 225
- **Trainable Parameters:** 3,013,773
- **Feature Extraction:** Convolutional layers, spatial pyramid pooling, upsampling layers

```
                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    752482  ultralytics.nn.modules.head.Detect           [6, [64, 128, 256]]
Model summary: 225 layers, 3012018 parameters, 3012002 gradients, 8.2 GFLOPs
```

Figure 5: YOLOv8 Model Summary

2.4 Dataset Preparation

Training Data:
We used `train.csv` and `img_size.csv` to create the training data for YOLOv8. The YOLO algorithm expects a label (`.txt`) file for each image, containing the following information for each identified class in an image:

- `class_id`
- `x_center`
- `y_center`
- `bbox_width`
- `bbox_height`

From `train.csv`, we obtain `x_min`, `y_min`, `x_max`, and `y_max` for each class in a particular image. From `img_size.csv`, we get `dim0` (height) and `dim1` (width) for each image.

Using these details, we compute the YOLO format information as follows:

```
x_center = (x_min + x_max) / 2 / width
y_center = (y_min + y_max) / 2 / height
bbox_width = (x_max - x_min) / width
bbox_height = (y_max - y_min) / height
```

However, if `class_id` is 14 (representing "No finding"), we write a label with `0.5 0.5 0.0 0.0`, indicating the absence of any findings.

Finally, we write the formatted label to a `.txt` file named after the `image_id`.

image_id	class_name	class_id	rad_id	x_min	y_min	x_max	y_max
bM8C97htulC9fHKlDurJHquCXr1KZuug	No finding	14	R5				
0FDQVdLgDKl1sRnPL94LzVh9EvXDVM9m	Aortic enlargement	0	R10	1148.0	503.0	1466.0	823.0
Dwk2TnGJFaMhyi3OfCrhdZG9ppGgIC5w	Consolidation	4	R8	264.0	732.0	550.0	1119.0
vqw6mWifHgCf8jmTotrMAS3qCk5eJuc4	No finding	14	R13				
EzfCkMwi4E5bAtZZo4brqt9dNbm7sF9z	No finding	14	R5				
SfZftHo1WlammJ8tb4rBMSzDaTtWYNsM	Nodule/Mass	8	R8	1624.0	1115.0	1741.0	1242.0
G9EcgYlokmElcAJefVC8olCAs84Kfeis	No finding	14	R5				
qvPw02EvPFN4H42FlkPlsUjEhQZkgqJe	Cardiomegaly	3	R9	786.0	1620.0	1618.0	2132.0
w6fPyFCOHnKXcXhD5m1XxEbfTa5nWDwx	Nodule/Mass	8	R8	1594.0	1222.0	1743.0	1381.0
XzjNuEQlogUWcTj2s8CFbtE4NcRimCtE	No finding	14	R4				
c9puPdGiltllvMlpl9WouxsSfltlw8fJ	No finding	14	R10				
BByLB5f2rCtoThi66OTBQjbSkd22xqce	Lung Opacity	7	R10	1359.0	1005.0	1521.0	1287.0

Figure 6: Sample from `train.csv`

img_size

image_id	dim0	dim1
j8ucb4pF6s210AWzYbWtWDjkHDfElvqh	3000	3000
QX1b3KBwsnmxGJzmdUwwH0leQtNlvYAU	3072	3072
kXObj3UtUTmZDgpBy3UdxAi2hef5ODEx	2836	2336
byLdYlU27P9zWLSYefZjEVRYBZPuwlW2	2952	2744
hTBiatGXktRwkGmEKdsLu7QSUYaH8XBa	2880	2304
m1mkU7C9WF4SlMnFfJ2x0cVeiaLjfbel	3170	2642
llXsB94LSKfiFDTmVqQ8qK5dYtQEyJdN	2819	2545
bo9m9fWglG3Bvzrw65zjPXSvHgC98UYN	3408	3320

Figure 7: Sample from `img_size.csv`

0Du41mCEDwbeOj9KzuSnSp0UPBQON808.txt

```
13 0.7565977175463623 0.3261543229005268 0.08523537803138374 0.0554694762937713
12 0.13391583452211128 0.7169197396963124 0.1587018544935806 0.23520297489928726
10 0.8660841654778887 0.6980167338084908 0.07453637660485021 0.03935543848775953
12 0.7908345221112696 0.6769445305237062 0.21647646219686162 0.0362565850635265
12 0.8396932952924394 0.6639293461419274 0.11019971469329529 0.049891540130151846
12 0.2640870185449358 0.23706228695382708 0.2892296718972896 0.24480942051440968
6 0.2710413694721826 0.47923768205763867 0.3074179743223966 0.19801673380849086
12 0.8457560627674751 0.6770994731949179 0.09165477888730385 0.02850945150294391
8 0.7337731811697575 0.5608924697861791 0.04386590584878745 0.05577936163619461
9 0.2710413694721826 0.47923768205763867 0.3074179743223966 0.19801673380849086
12 0.23858773181169757 0.4656027269910335 0.3751783166904422 0.6969321351100093
10 0.8445078459343794 0.6780291292221878 0.11412268188302425 0.0638363805392005
9 0.04547075606276747 0.5241710567090176 0.055991440798858774 0.2953207313294081
12 0.13855206847360912 0.7172296250387357 0.1522824536376605 0.22838549736597458
7 0.7348430813124108 0.5602726991013325 0.05099857346647646 0.04400371862410908
12 0.27014978601997147 0.3650449333746514 0.2606990014265335 0.2094824914781531
12 0.12660485021398002 0.7204834211341804 0.12125534950071326 0.20824295010845986
10 0.8170470756062768 0.7074682367524016 0.17760342368045648 0.02850945150294391
9 0.03940798858773181 0.5357917570498916 0.06098430813124108 0.30926557173845676
9 0.04368758915834522 0.5294391075302138 0.031027104136947217 0.31515339324449954
```

Figure 8: Sample .txt File Generated for an Image

3 Model Training and Evaluation

3.1 Training Process

We split all the training images (and corresponding labels(.txt) files) into three parts, train, test and validation. We used 80 percent of all available data for training (6858 images and labels), and split the rest equally between validation and test (858 images and labels each).

We used the pre-trained yolov8n.pt model and trained it on our dataset for 20 epochs (we kept the number of training epochs comparatively low due to time and space complexity issues) with batch size as 16 and image size as 640*640.

After each epoch, we have the model checkpoint saved.

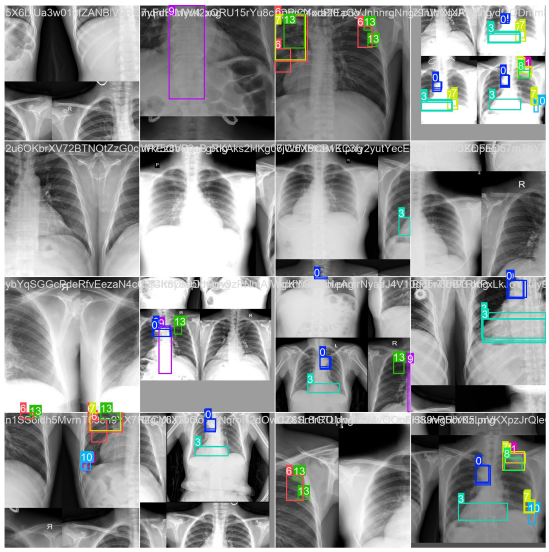

Figure 9: Sample Training Batch 0

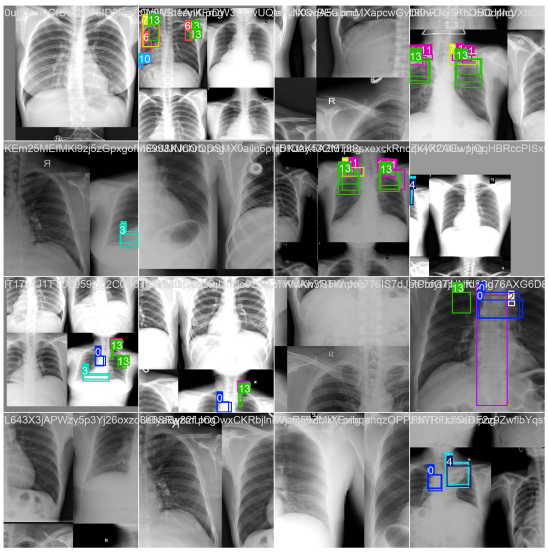

Figure 10: Sample Training Batch 1

3.2 Performance Metrics

Overall Performance:

- **Precision (P): 0.501** This is a measure of the accuracy of a binary classification model, representing the proportion of true positive predictions (correct positive predictions) out of all positive predictions made by the model.
- **Recall (R): 0.507** This measures the ability of a model to correctly identify all positive instances, indicating the proportion of true positive predictions out of all actual positive instances in the data.
- **mAP50: 0.546** It calculates the average precision (AP) across multiple classes and then takes the mean of these average precisions. The "50" indicates that it considers precision at 50% recall.

3.3 Hyperparameter Tuning

To prevent overfitting, we adjusted the model's hyperparameters the following way:

- `batch=16`: Batch size
- `imgsz=640`: Input image size
- `augment=True`: Enabling Data augmentation
- `fliplr=0.5`: Horizontal flip augmentation
- `hsv_h=0.015, hsv_s=0.7, hsv_v=0.4`: Color jitter augmentation
- `weight_decay=0.0005`: Weight decay regularization
- `lr0=0.001`: Initial learning rate
- `dropout=0.3`: Dropout rate

4 Results

4.1 Output Results:

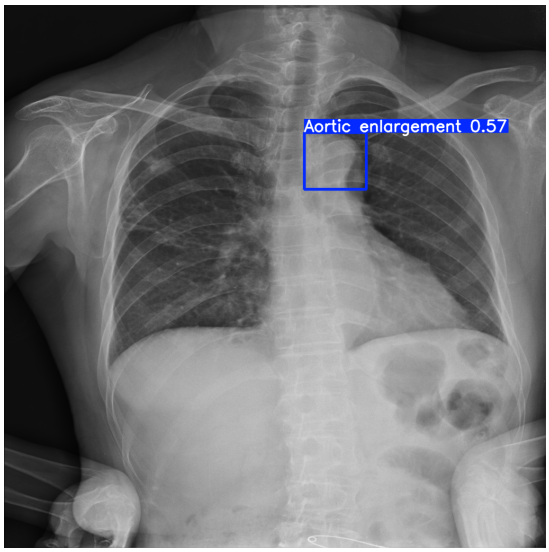

Figure 11: Sample Result 1

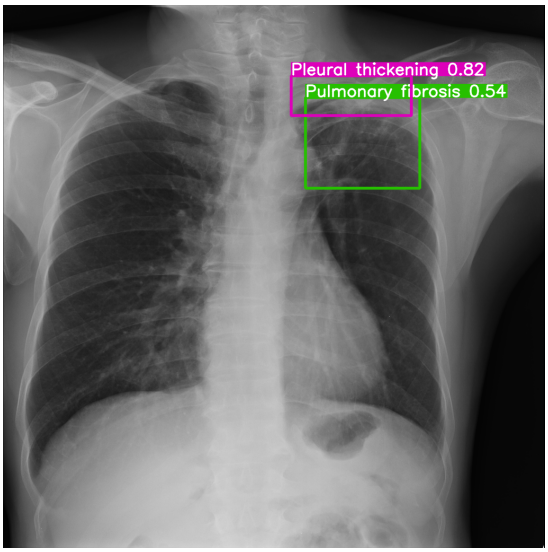

Figure 12: Sample Result 2

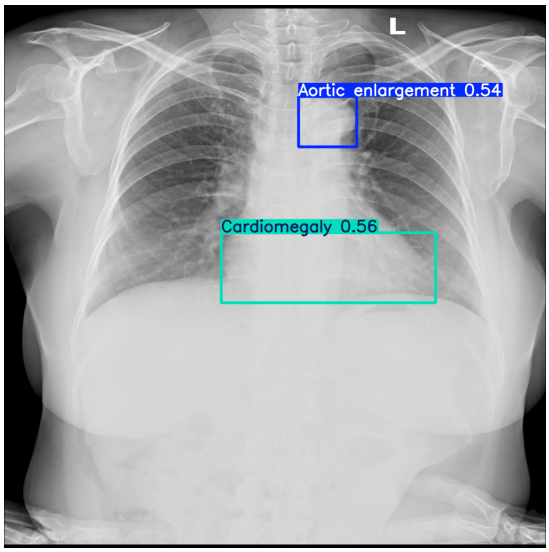

Figure 13: Sample Result 3

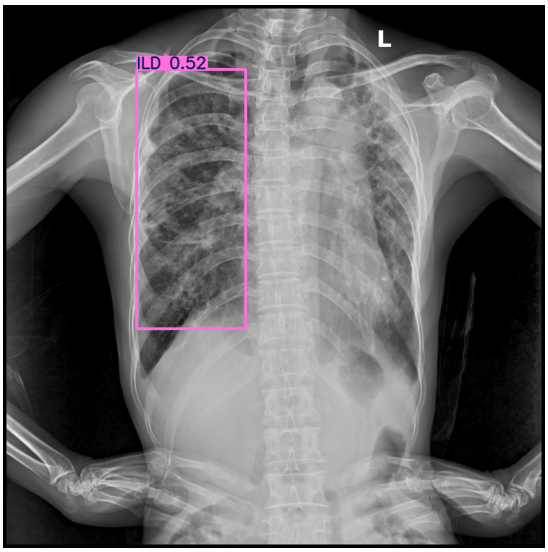

Figure 14: Sample Result 4

4.2 Comparison: Original vs. Tuned Model

Original Model (without Hyperparameter tuning as mentioned in section 3.3):

- Exhibited higher overfitting during validation.
- Demonstrated lower overall recall.

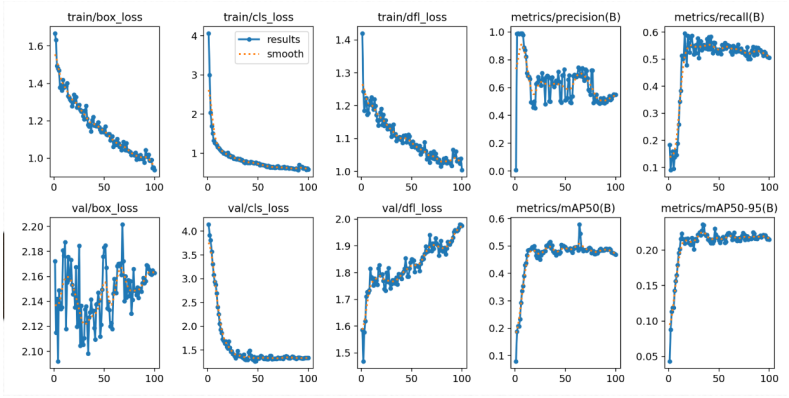

Figure 15: Original Model

Tuned Model:

- Showed reduced overfitting.
- Achieved a slight increase in overall performance and precision values.

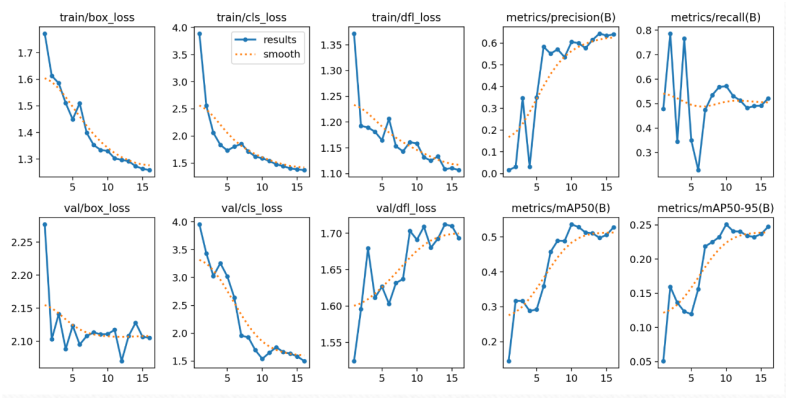

Figure 16: Tuned Model

5 Reproducibility

5.1 System Specifications:

- Mac OS version and the package versions: System Version: `macOS 12.6 (21G115)` Kernel Version: `Darwin 21.6.0`

5.2 Software Specifications:

- Jupyter Notebook, version 6.5.4

5.3 Python Packages used:

- python 3.11.5
- yolo 8.1.45
- numpy 1.24.3
- torch 2.1.1
- cv2 4.9.0
- ultralytics 8.1.45
- matplotlib 3.8.4
- scipy 1.11.1
- requests 2.31.0
- pandas 2.2.2
- tqdm 4.65.0
- PIL 10.0.1

6 Issues and Challenges

- **Computationally Expense:** Training the model with more epochs and images is computationally intensive.

7 Improvement Scope and Future Work

1. **More Training Instances:** Increase the training data to enhance model robustness.
2. **Advanced Augmentation:** Introduce additional synthetic variations to diversify the training dataset.
3. **Hyperparameter Tuning:** Conduct systematic experiments to optimize key parameters.

4. **Architectural Modifications:** Experiment with different optimizers, activation functions, and batch normalization techniques.

5. **Class Assignment Algorithm:** Implement algorithms to give more weight to under represented classes in training dataset and check how that affects the overall model performance.

8 Conclusion

The project has shown that by utilising deep learnning networks like YOLO, it is possible to detect and categorise thoracic anomalies from chest radiographs. The accuracy and robustness of the model should be improved by suggested changes in data collection, augmentation, model architecture, and hyper-parameter tuning. These efforts have the potential to greatly improve our understanding and ability to optimise automated diagnostic tools in the medical industry, which will eventually improve patient outcomes and facilitate healthcare processes.

References

[1] AMIA Public Challenge 2024 `https://www.kaggle.com/competitions/amia-public-challenge-2024`.

[2] Ross Girshick Ali Farhadi `http://pjreddie.com/yolo/` Joseph Redmon, Santosh Divvala. Yolo: Real-time object detection, 2024.

[3] Ultralytics "Ultralytics YOLO v8" GitHub repository `https://github.com/ultralytics/ultralytics.git`.