Great Learning

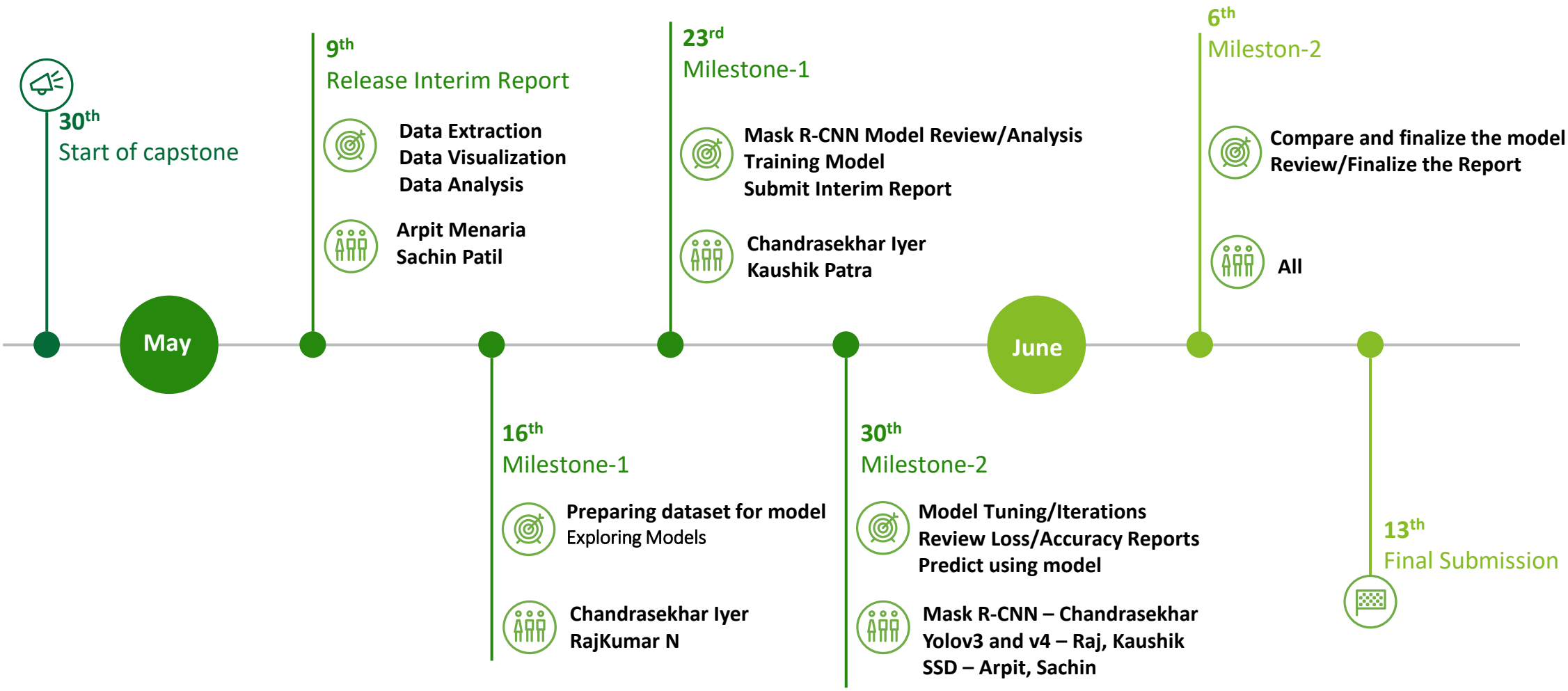
AIML Online Capstone

Pneumonia Detection Challenge

Arpit Menaria,
Chandrasekhar Iyer
Kaushik Patra
Rajkumar N
Sachin Patil

Objective, Approach and Timelines

Build a pneumonia detection system, to locate the position of inflammation in an image.



Summary of Problem statement, data and findings

Abstract

Pneumonia is a bacterial, viral, or fungal infection that causes swelling of the tissues in one or both lungs. Currently, diagnosis for pneumonia requires an expert to find anomalies in the X-ray images, as it is a widespread disease, and it is not always possible for doctors to review all the cases. Most of the caused deaths are due to late diagnosis or some human error, which leads to an incorrect diagnosis of the disease. Artificial intelligence models are becoming an integral part of modern computing systems and algorithm based on convolutional neural network has achieved breakthroughs in the field of target detection. As part of this project the objective is to make an automated diagnosis and identify infected regions in the lungs.

Summary of findings

As part of this experiment, we evaluated the following models

- Mask RCNN
- Yolo v4 TensorFlow
- SSD
- Yolo v3 Darknet

The experimental results indicate that Mask RCNN gets an IoU score of 0.55 and a dice co-efficient of 0.77, which is about 12.7% improvement over other methods, i.e., YOLOv3, YOLOv4 & SSD. We were able to achieve. When compared to our initial benchmark during interim state (0.35 Dice coeff), we were able to achieve a 100% improvement due to fine tuning.

Implication

However, the MASK-RCNN algorithm easily treats part of the background in the image as foreground, which results in false positives of target segmentation, which does not make it industry ready yet. It would require further training with appropriate up-sampling and higher quality imaging

Overview of the final process

1. Understand the use-case and research on the domain
2. Data extraction and EDA
3. Data visualization
4. Data Preprocessing
5. Model(s) Training & Tuning
6. Model(s) comparison & Evaluation

1
- Understand the problem statement and objective of the project experiment
- Research on the most appropriate solution for the problem at hand
- Decide on the sequence of activities to arrive at the final solution

2
- Extract subset of the data so that multiple team members could work on the same dataset in parallel
- Fix the imbalance data issue to avoid the model from learning from the negative cases
- Merge the labels to avoid duplicate information due to multiple bounding boxes
- Plot the data distribution, co-ordinate outlier analysis, duplicate data using MD5 checksum and centroid analysis

3
- Analyze the DICOM construct and visualize it
- Visualize the pixel array for patients with and without pneumonia
- Visualize masks based on bounding box co-ordinates

4
- Resize data based on target model requirements
- Convert images to JPG and format annotations based on model requirements
- Perform image augmentation using imgaug library
- Create the relevant directory and processing pipeline to conform with Darknet network requirements

5
- Train the following models – Mask RCNN, Yolo V3, Yolo V4 and SSD.
- Identify the LR range to work with using LR finder library
- Integrate callbacks like tensorboard for histogram analysis and Modelcheckpoint for saving weights
- Tune the model using various approaches like (Image augmentation, Hparams - batch size, LR, Steps per epoch, freeze CNN layers)
- Calculate metrics like classification MAP, regression loss using Dice coefficient, IoU etc.

6
- Compare the 4 models listed above on the basis of IoU
- Identify areas of weaknesses within each of the model and potential areas of improvements

Working with the Dataset

Understanding the Dataset

Exploratory Data Analysis

Understanding the raw dataset

Dataset comprises of 4 files required for model training and testing

File/Folder Name	Description	Details
stage_2_train_labels.csv	Contains patientIds and bounding box/ target information	Columns – patientId, x, y, width, height, Target (0,1) Rows – 30227
stage_2_detailed_class_info.csv	Provides detailed information about the type of positive or negative class for each image.	Columns – patientId, class (No Lung Opacity/Not Normal, Normal, Lung Opacity) Rows – 30227
Folder ../stage_2_train_images	Set of dicom Xray images to be used for model training and validation	26684 image files
Folder ../stage_2_test_images	Set of dicom Xray images to be used for model testing	3000 image files

Exploratory Data Analysis

Null check/Missing values

- In stage_2_train_labels.csv there are **20672 null values** for the bounding boxes associated with patientId with no Pneumonia, ie Target = 0. This is expected and need not be treated or populated.
- There are **no other missing values** in the dataset

Duplicate checks

- Both the csv files have 30227 rows, which comprises of repeating patientId where the same patient has more than one lung opacity in the DICOM image, which are identified with different set of bounding boxes
- On merging and grouping duplicate we get **26684 unique patiendIds**

Checksum validation on DICOM files

- To ensure there are no duplicate DICOM images in the dataset, we performed checksum.
- It came with **26684 unique checksum** values against 26684 unique patientIds and 26684 DICOM image files

EDA – Images with multiple bounding boxes

Distribution of DICOM images with multiple bounding boxes for lung opacity for unique patient

x	patientId	y	width	height	Target	class
0	0	20672	20672	20672	20672	20672
1	1	2614	2614	2614	2614	2614
2	2	3266	3266	3266	3266	3266
3	3	119	119	119	119	119
4	4	13	13	13	13	13

Inference

- **20672** images are without any bounding boxes for patients with Target = 0 (Class = Normal or No Lung opacity/Not Normal)
- **6012** images have one or more bounding boxes in a single DICOM file for Target =1 (Class = Lung Opacity)
- Most common cases of Pneumonia have 2 bounding boxes

Data Visualization

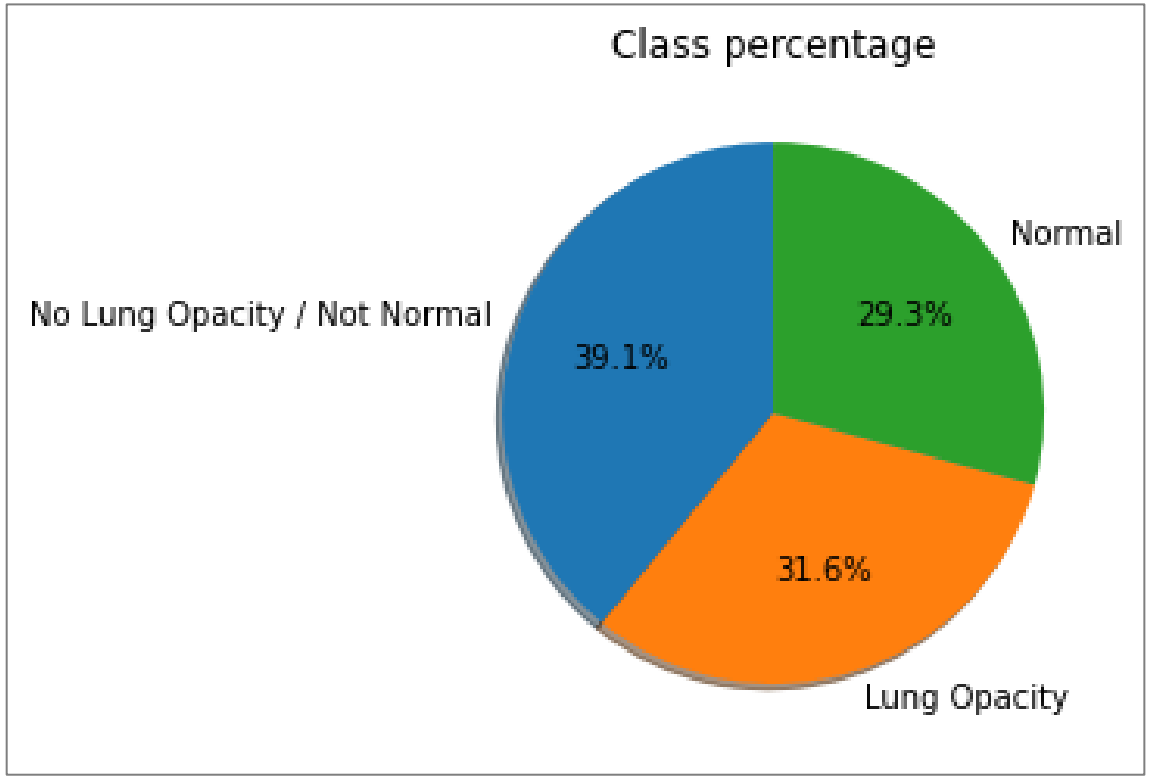
Class and Target distribution

Bounding box data distribution

Dicom images

Data Visualization – Distribution of Classes and Targets

Distribution across Class

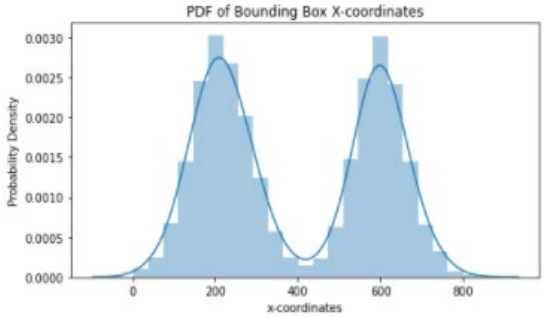
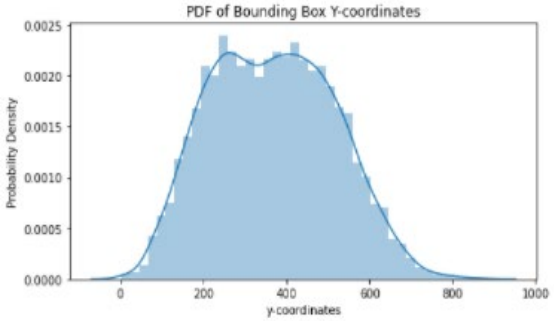
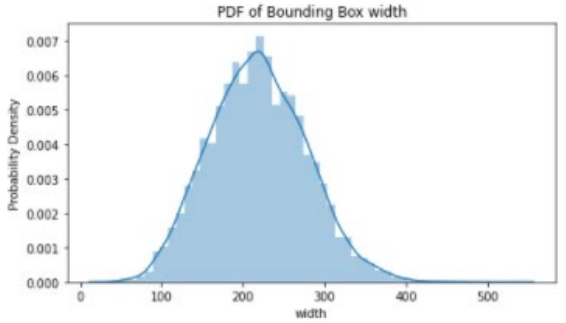
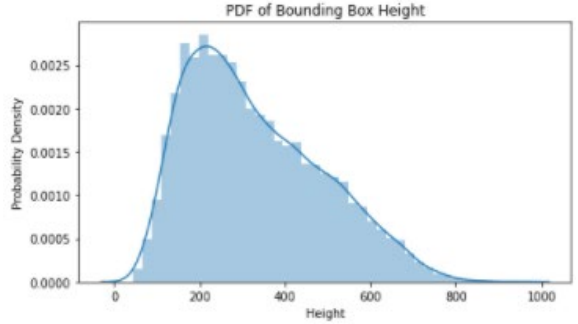
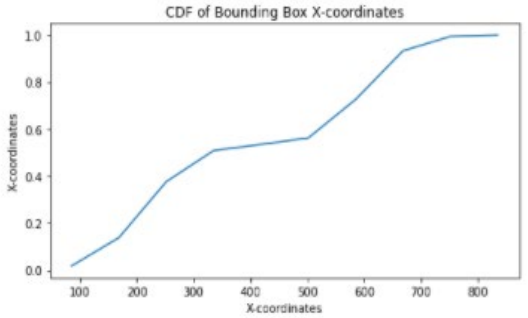
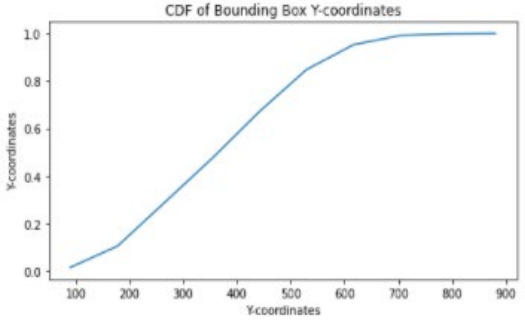
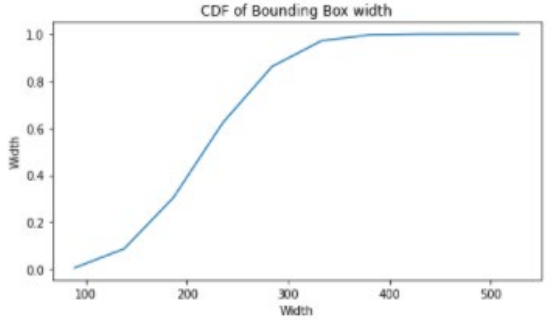
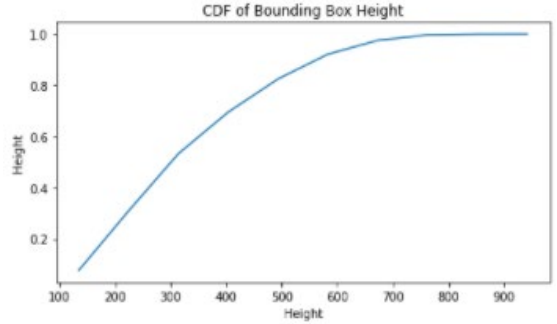
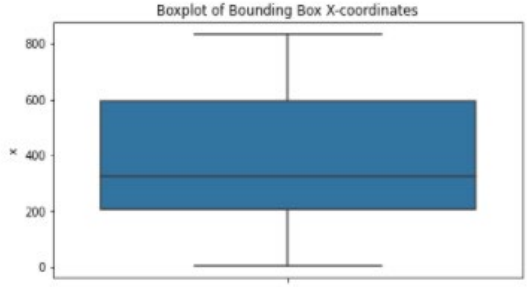
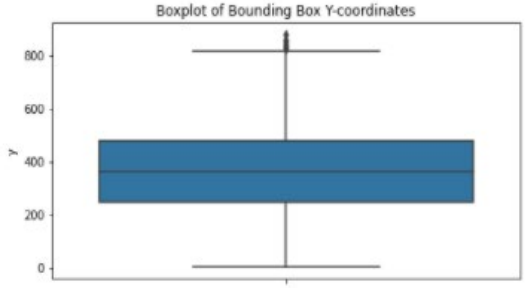
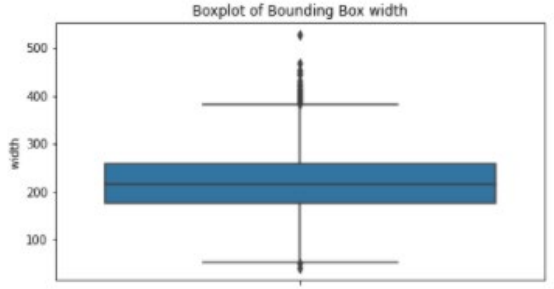
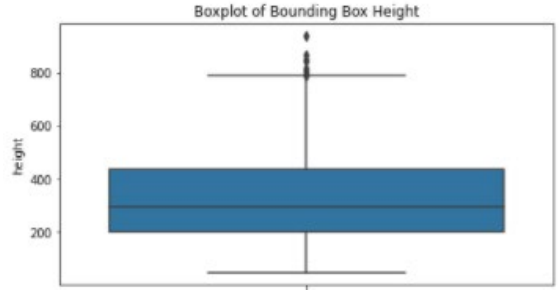


Distribution of Target

Inference

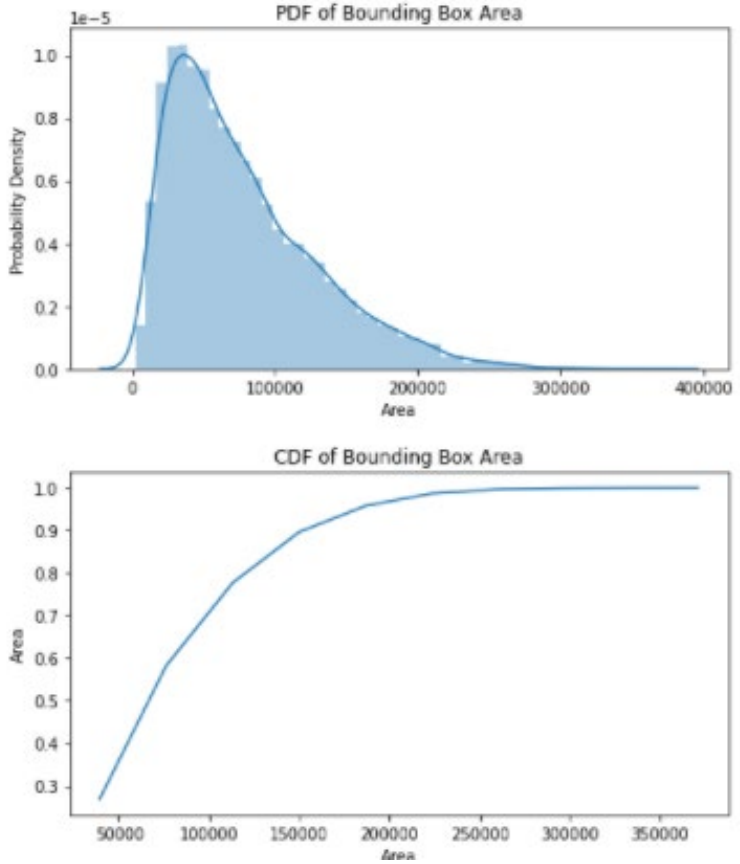
- All records with Target = 1 (pathology detected) are associated with class = Lung Opacity.
- The records with Target = 0 (no pathology detected) are either of class = Normal or No Lung Opacity/Not Normal
- The dataset is quite imbalance where Target = 0 (class = Normal or No Lung Opacity/Not Normal) comprises of 68% of the dataset, whereas Target = 1 (class = Lung Opacity) comprises of 32%

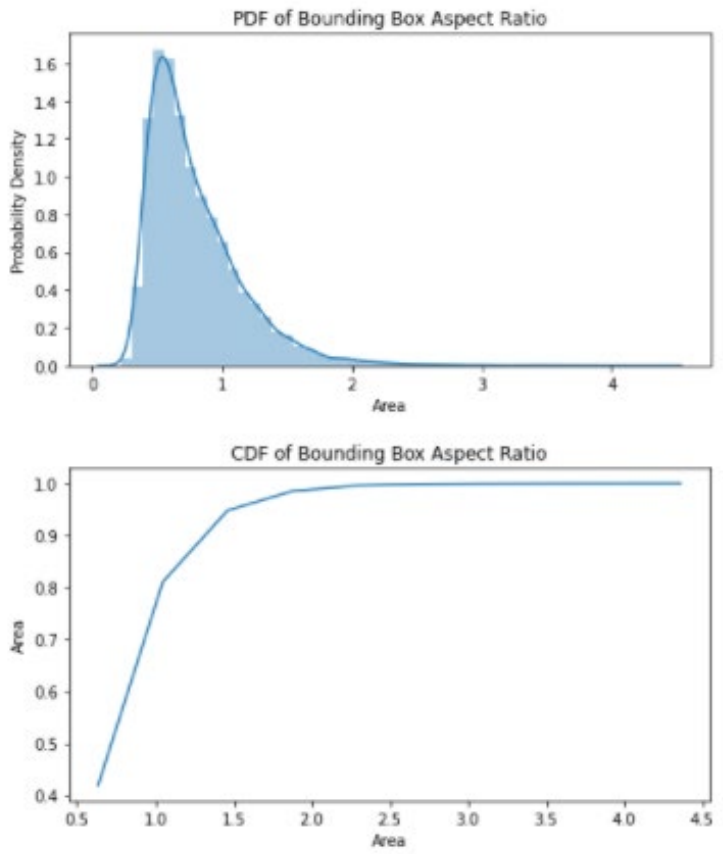
Data Visualization – Bounding Box Co-ordinates

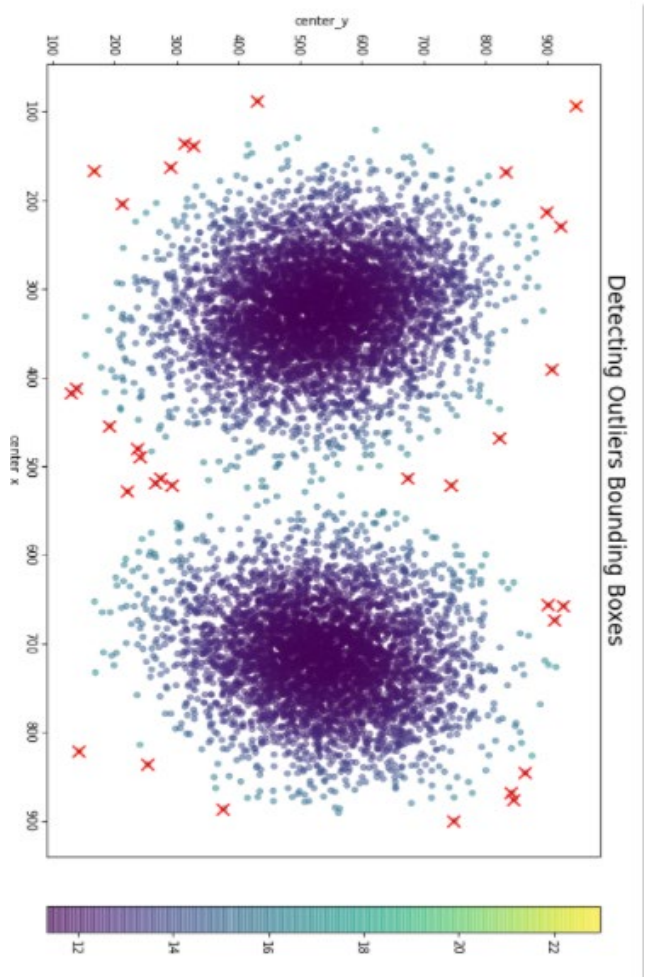
X-coordinate	Y-coordinates	Width	Height
			
			
			

Data Visualization – Bounding Box properties

Area	Aspect Ratio	Centroid







Data Visualization – DICOM images

DICOM images with Target = 0

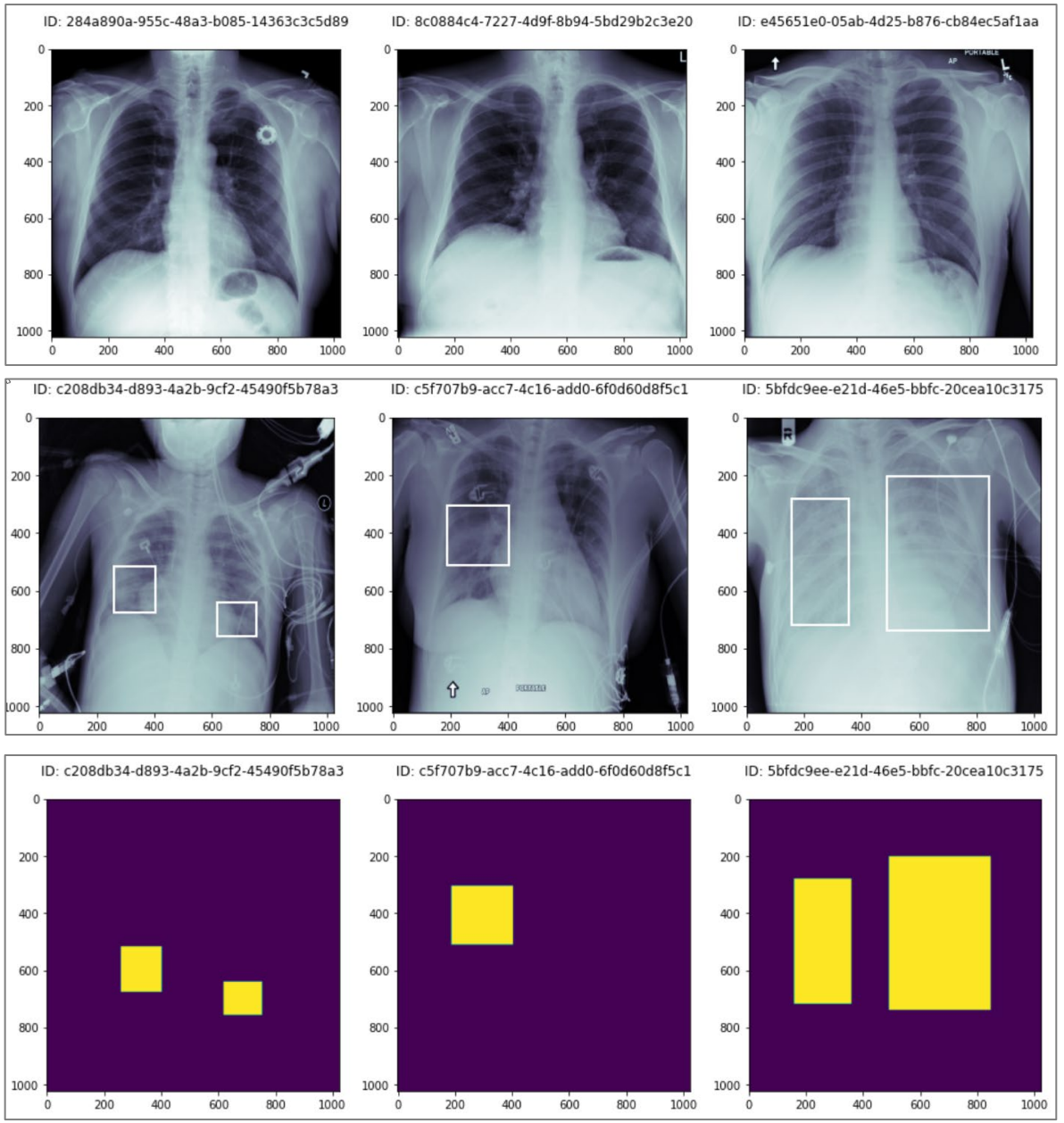
These images don’t have any bounding boxes and hence no relevant lung opacity for patients without pneumonia

DICOM images with Target = 1

The bounding boxes in these images show the location of lung opacity for patients with pneumonia

Masks for images with Target = 1

Masking the images for lung opacity



Data Pre-processing

Filtering

Image Augmentation

Resizing

Data Pre-processing for Model Training

Filtering Dataset

We have taken a subset of the original dataset due to ***GPU/CPU limitations***. 800 samples with Target = 0 is used for '***Hard –ve sample training***'

- Lung Opacity – 3000
- No Lung Opacity/Not Normal – 400
- Normal – 400

Image and mask resizing

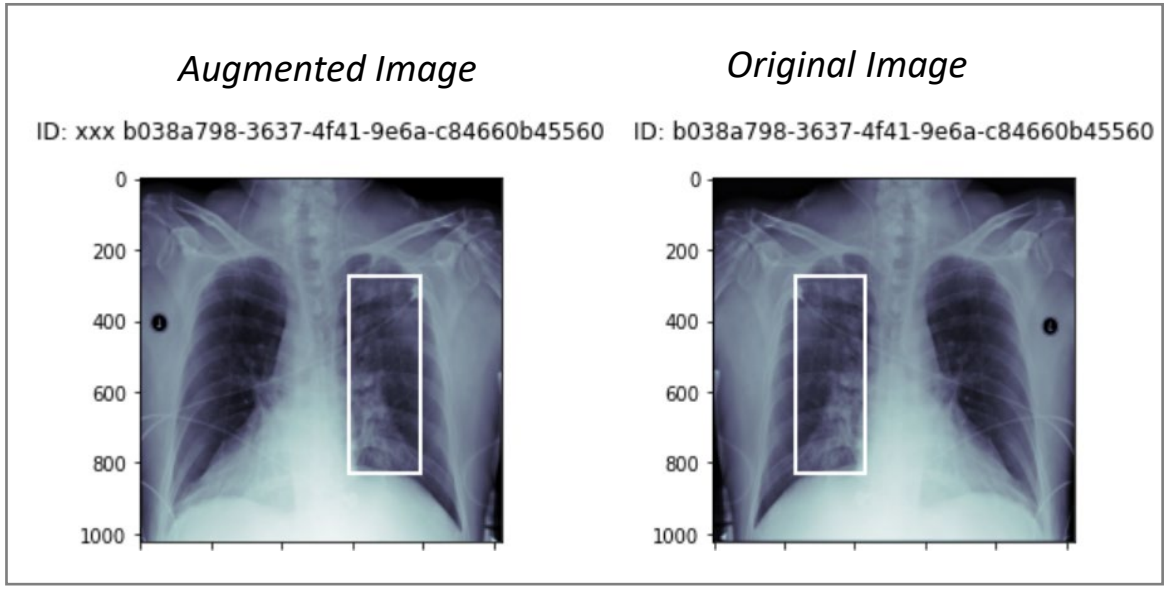
As a part of model training following resizing is performed on the dataset for different models

- SSD – 300 x 300
- RCNN - 256 x 256
- Yolov3 - 608 x 608
- Yolov4 – 416 x 416

Image Augmentation

Following augmentation techniques have been applied to 3000 image files randomly to handle…

- ***Different orientation and size*** - scale, translate, shear, horizontal flip
- ***Different image quality*** - brightness, contrast, blur, sharpen
- Example depicted below



SSD

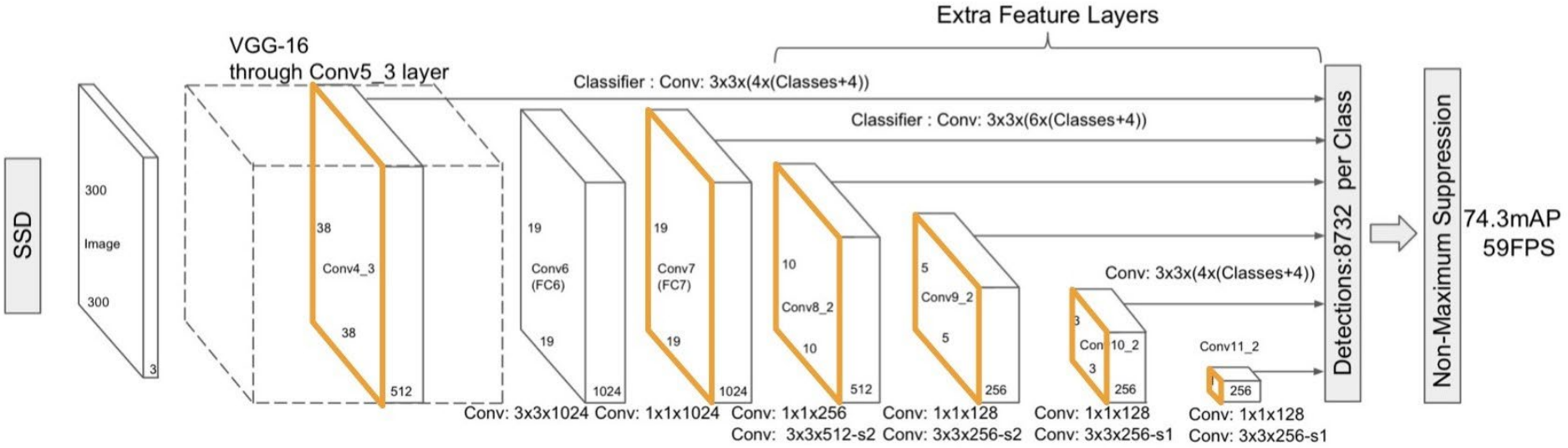
Model Overview

Model Advantages

Training Iterations

Testing and Evaluation

SSD Model Overview



Architecture overview

- SSD has a base VGG-16 network followed by multibox conv layers
- Base neural network extracts features - VGG-16 base network for SDD is standard CNN architecture for high quality image classification but without the final classification layers. VGG-16 is used for feature extraction.
- Additional Conv Layers detect objects - To the base VGG network, we add additional convolutional layers for detection. Convolutional layers at the end of the base network decrease in size progressively. This helps with detection of objects at multiple scales. The convolutional model for detection is different for each feature layer.

SSD Model Advantages

Advantages of SSD

- Accuracy increases with the number of default boundary boxes(but at the cost of speed). Designing better default boundary boxes will help accuracy.

- Multi-scale feature maps improve the detection of objects at a different scale.

- SSD has lower localization error comparing with R-CNN but more classification error dealing with similar categories. The higher classification errors are likely because we use the same boundary box to make multiple class predictions.

3e-3

SSD Training Iteration

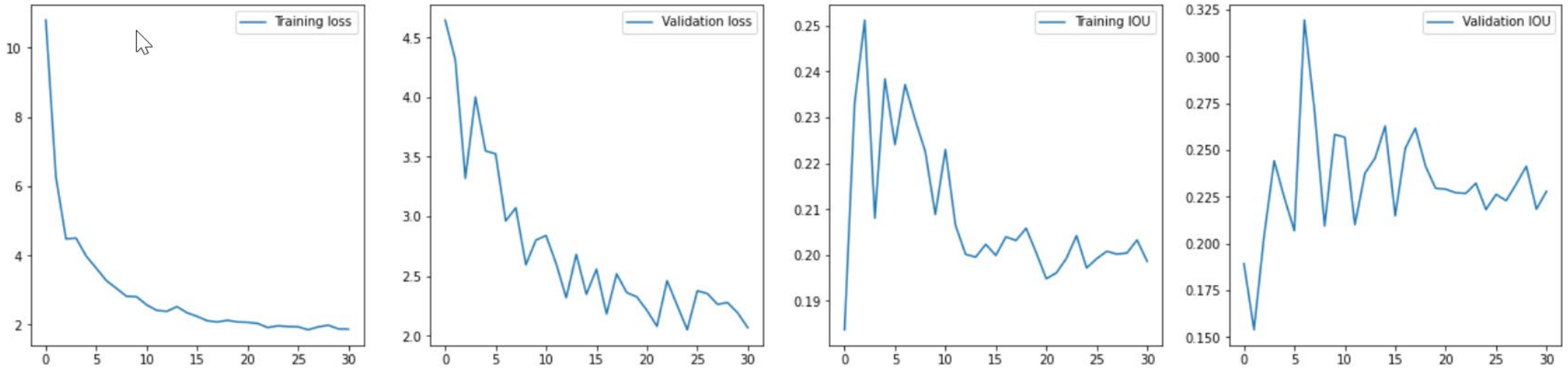
Objective – Train using Lr 3e-3 and Adam optimizer

Total params: 23,497,767
Trainable params: 1,210,471
Non-trainable params: 22,287,296

Summary

F1 score = 0.384 (train), 0.381 (val)
IoU = 0.197 (train), 0.194 (val)
Loss = 2.180 (train), 2.223 (val)

Loss vs IoU

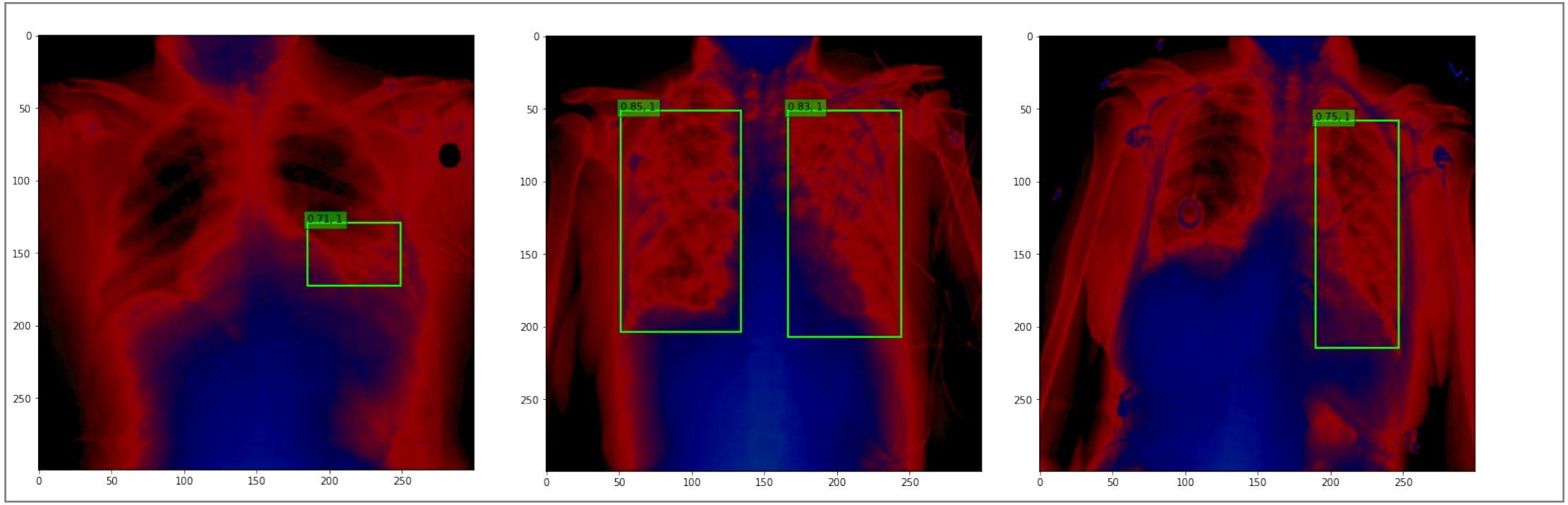


SSD Testing and Evaluation

Summary of model testing:

- Model was able to detect lung opacity successfully in images with confidence set to 0.7
- IoU: 31%

Model Predictions



Yolov4 TensorFlow

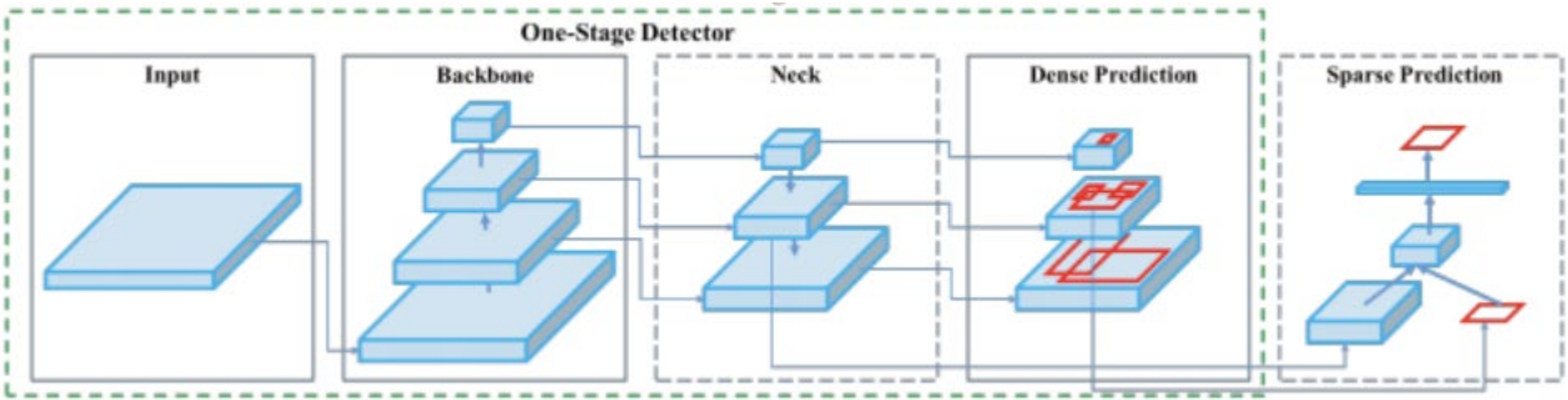
Model Overview

Model Advantages

Training Iterations

Testing and Evaluation

Yolov4 TensorFlow Model Overview



Architecture overview

- YOLOv4 was a real-time object detection model published in April.
- It works by breaking the object detection task into two pieces, regression to identify object positioning via bounding boxes and classification to determine the object's class.
- This implementation of YoloV4 uses the Darknet framework, **which is converted in TensorFlow**
- By using YOLOv4, you are implementing many of the past research contributions in the YOLO family along with a series of new contributions unique to YOLOv4 including new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss. In short, with YOLOv4, you're using a better object detection network architecture and new data augmentation techniques.

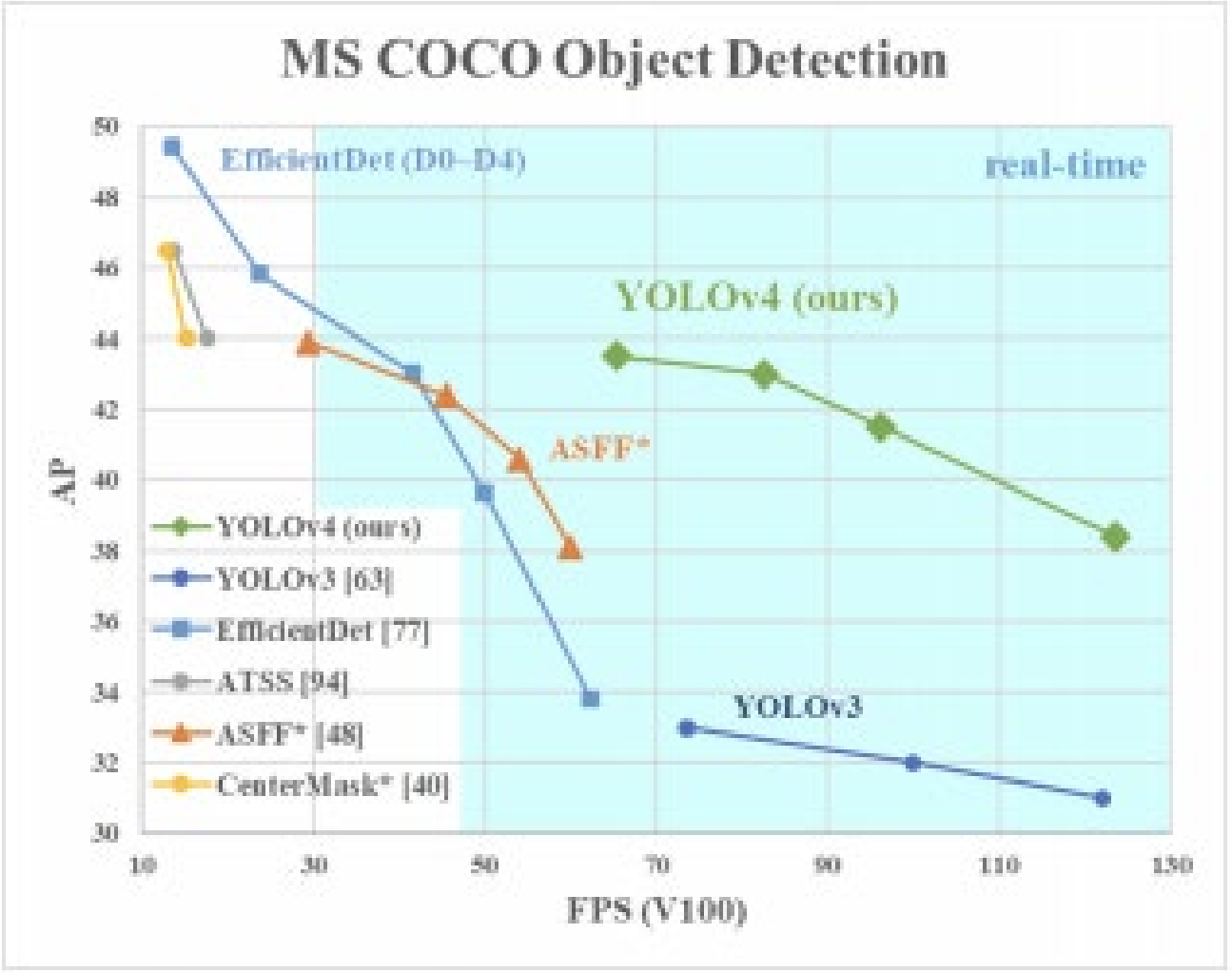
Yolov4 TensorFlow Model Advantages

Advantages for Mask R-CNN

- YOLOv4 has an incredibly high performance for a very high FPS; this was a major improvement from previous object detection models which only had either high performance or high inference speeds.

- The influence of state-of-the-art “Bag-of-Freebies” and “Bag-of-Specials” object detection methods during detector training has been verified.

- The modified state-of-the-art methods, including CBN (Cross-iteration batch normalization), PAN (Path aggregation network), etc., are now more efficient and suitable for single GPU training.



Yolov4 TensorFlow Training Iteration

Objective – Trained on Darknet and converted weights to Tensorflow

.cfg file
batch = 64
subdivisions = 32
max_batches = 6000
steps = 4800, 5400
classes = 1
filters = 18
width = 416
height = 416

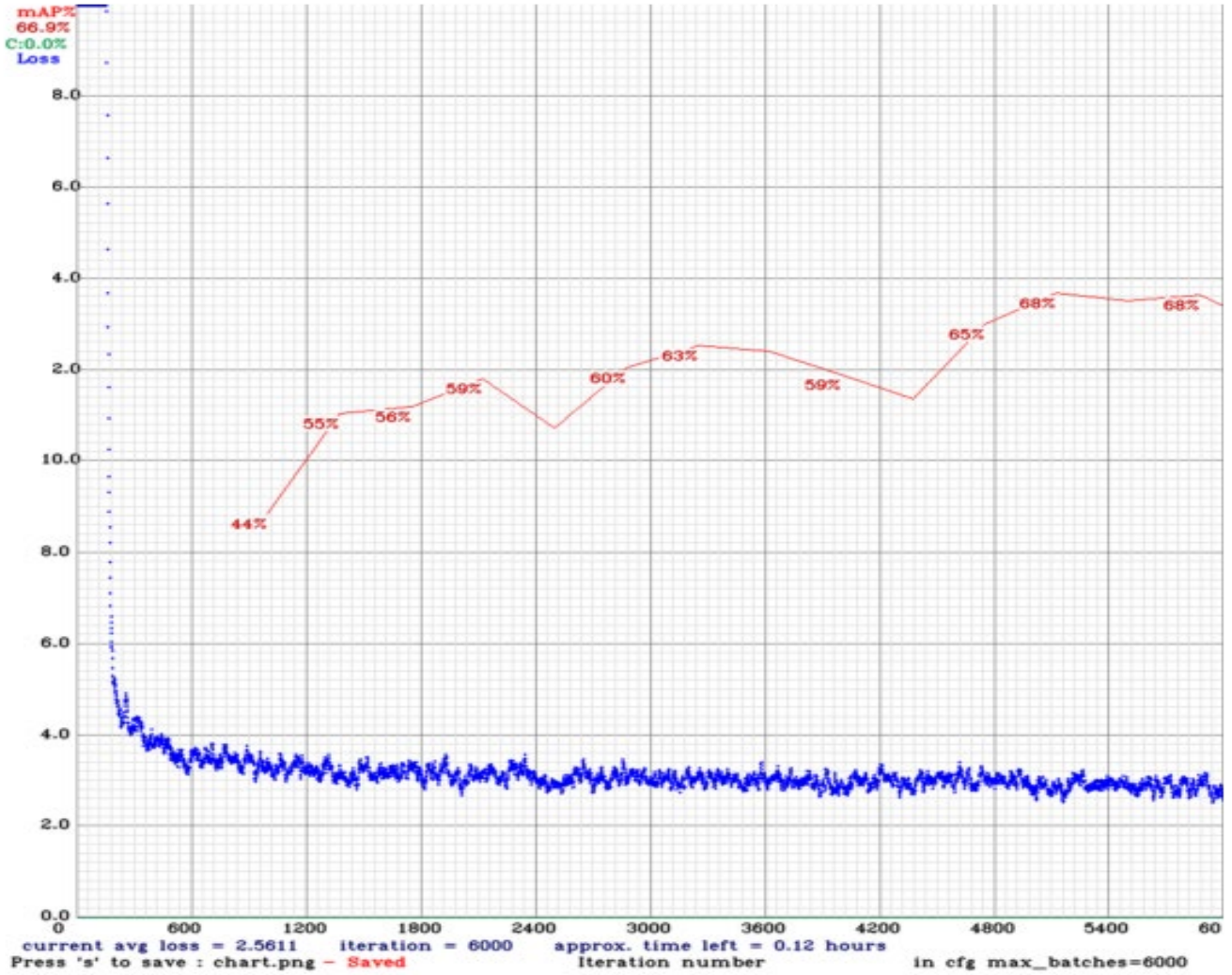
obj.names
Lung_Opacity

obj.data
classes = 1
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /mydrive/yolov4/backup

Summary

Precision = 0.69
Recall = 0.70
Average IoU = 48.32
mAP@0.50 = 0.683289, or 68.33 %

Loss vs mAP

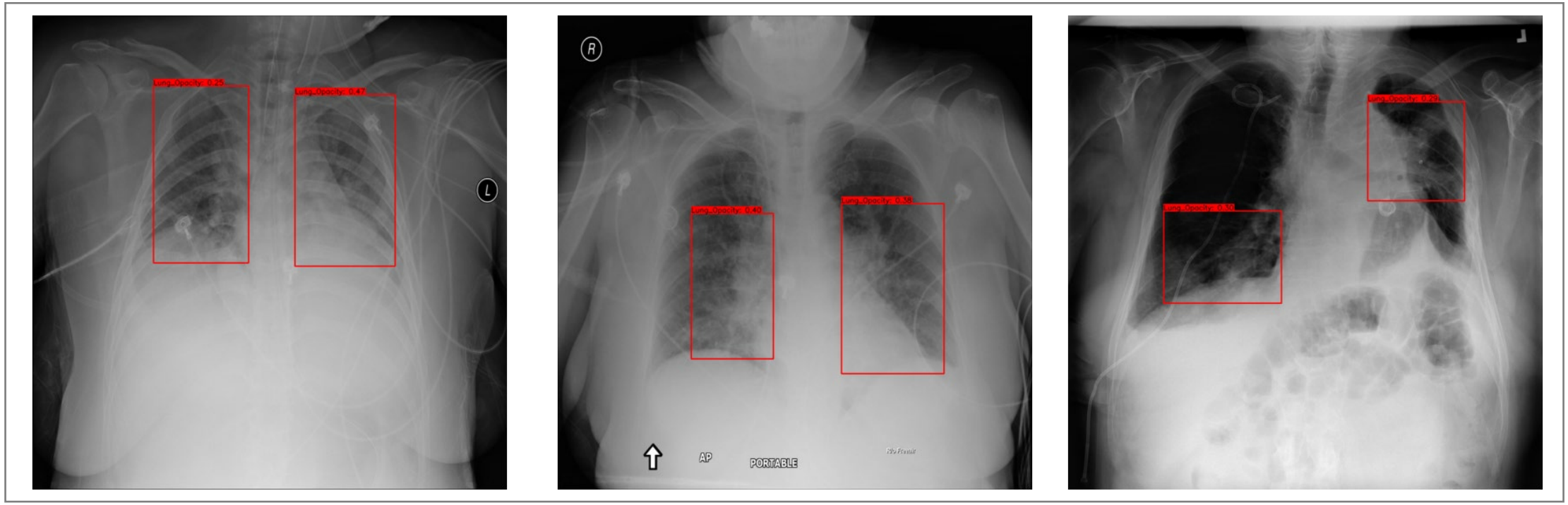


Yolov4 TensorFlow Testing and Evaluation

Summary of model testing:

- Model was able to detect lung opacity successfully in images when the threshold was set to 0.2
- Model average IoU = 48.32 % and mAP@0.50 = 68.33 %

Model Predictions



Yolov3 Darknet

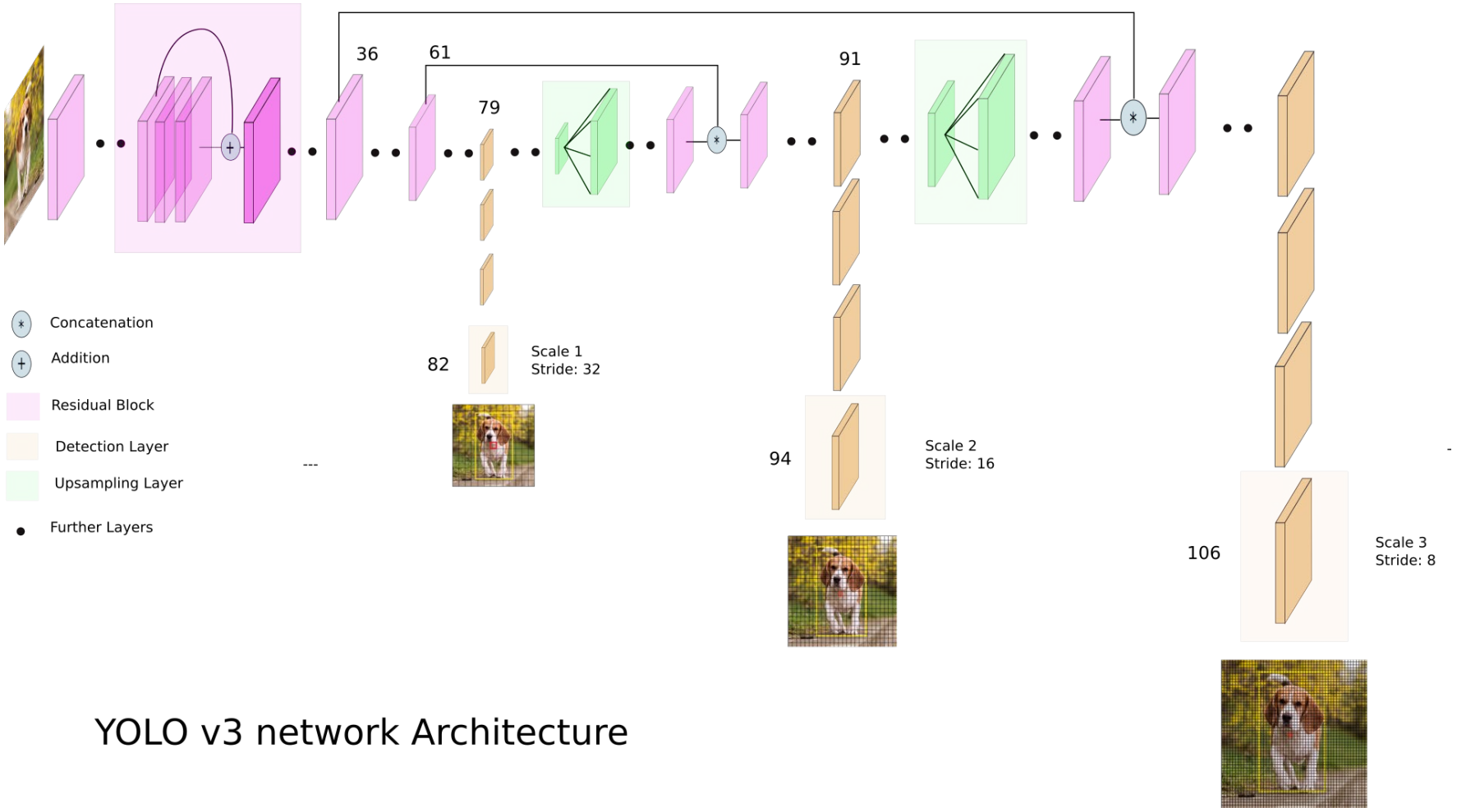
Model Overview

Model Advantages

Training Iterations

Testing and Evaluation

Yolov3 Darknet Model Overview



YOLO v3 network Architecture

Architecture overview

- YOLO v3 uses a variant of Darknet, which originally has 53 layer network trained on Imagenet. For the task of detection, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for YOLO v3.
- The architecture boasts of residual skip connections, and upsampling.
- The most salient feature of v3 is that it makes detections at three different scales.

Yolov3 Darknet Model Advantages

Advantages for Yolov3

- YOLO v3 makes prediction at three scales, which are precisely given by downsampling the dimensions of the input image by 32, 16 and 8 respectively.

- YOLO v3, in total uses 9 anchor boxes. Three for each scale. If you're training YOLO on your own dataset, you should go about using K-Means clustering to generate 9 anchors.

- For an input image of same size, YOLO v3 predicts more bounding boxes than YOLO v2. For instance, at it's native resolution of 416 x 416, YOLO v2 predicted 13 x 13 x 5 = 845 boxes. At each grid cell, 5 boxes were detected using 5 anchors.

- On the other hand YOLO v3 predicts boxes at 3 different scales. For the same image of 416 x 416, the number of predicted boxes are 10,647. This means that **YOLO v3 predicts 10x the number of boxes predicted by YOLO v2.** You could easily imagine why it's slower than YOLO v2.

Yolov3 Darknet Training Iteration

Objective – Design custom object detector for RSNA dataset using Darknet yolov3 and pre-trained weights

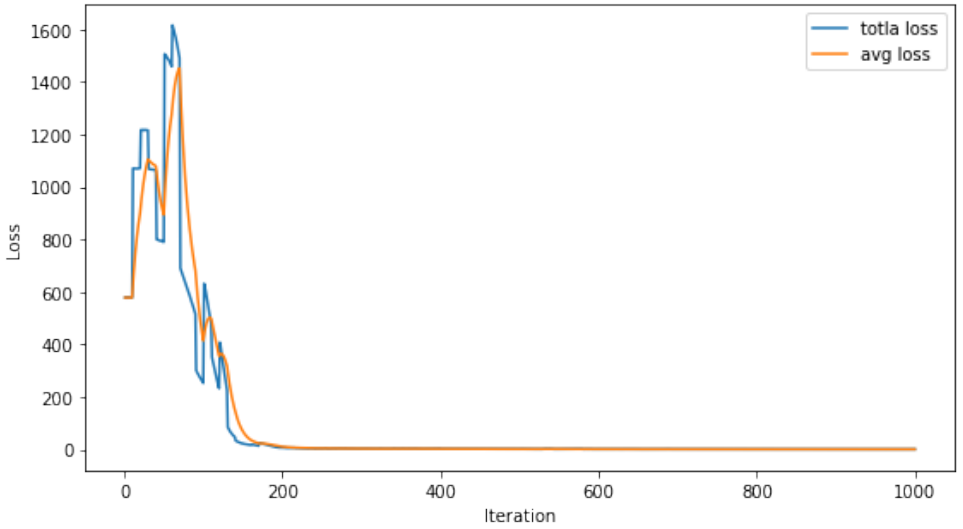
Configuration file

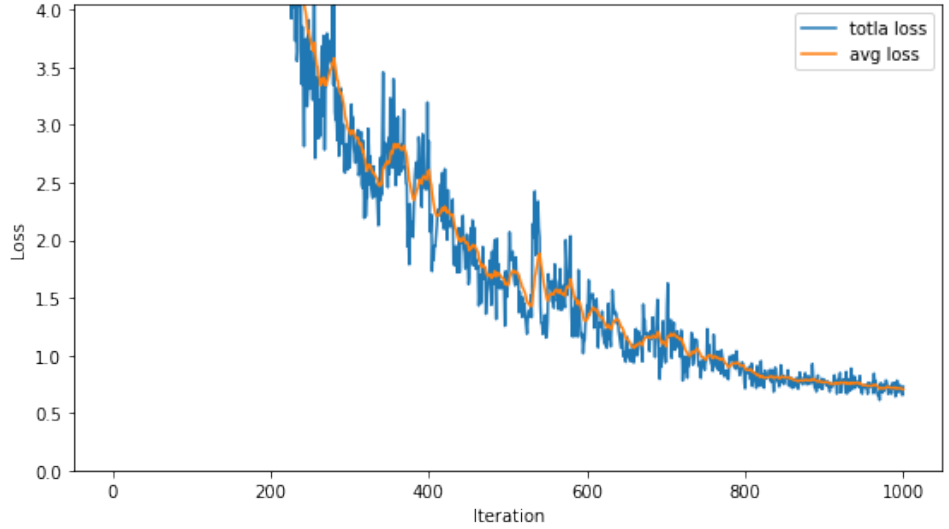
batch=64, subdivisions=32
width=608, height=608
channels=3
momentum=0.9
decay=0.0005
learning_rate=0.001
max_batches = 1000

Summary

Across iterations IoU ranges between 0.55 to 0.75

Loss vs Accuracy



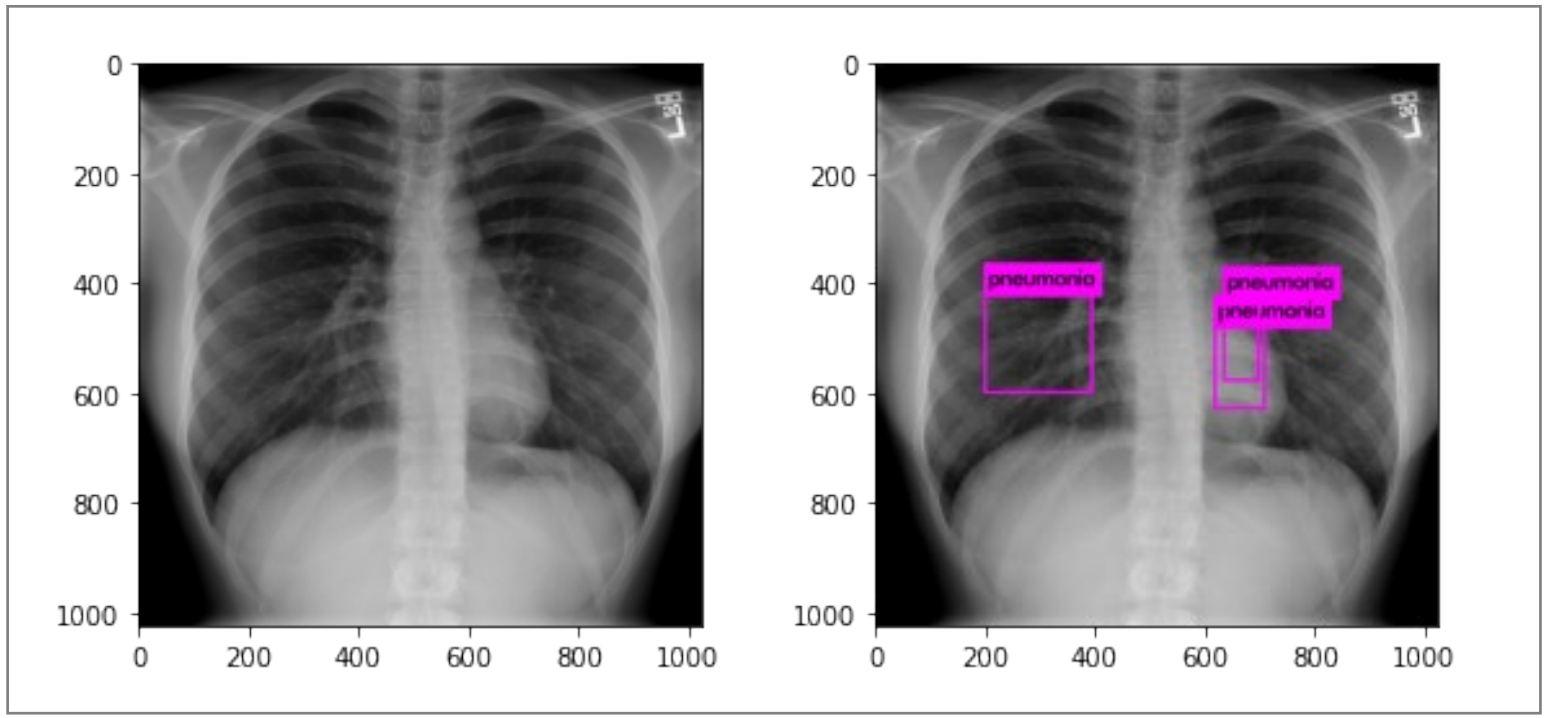


Yolov3 Darknet Testing and Evaluation

Summary of model testing:

- Model was able to detect lung opacity successfully in images when the threshold was set to thresh 0.005
- Across iterations IoU ranges between 0.45

Model Prediction in a Test image



Mask R-CNN Model

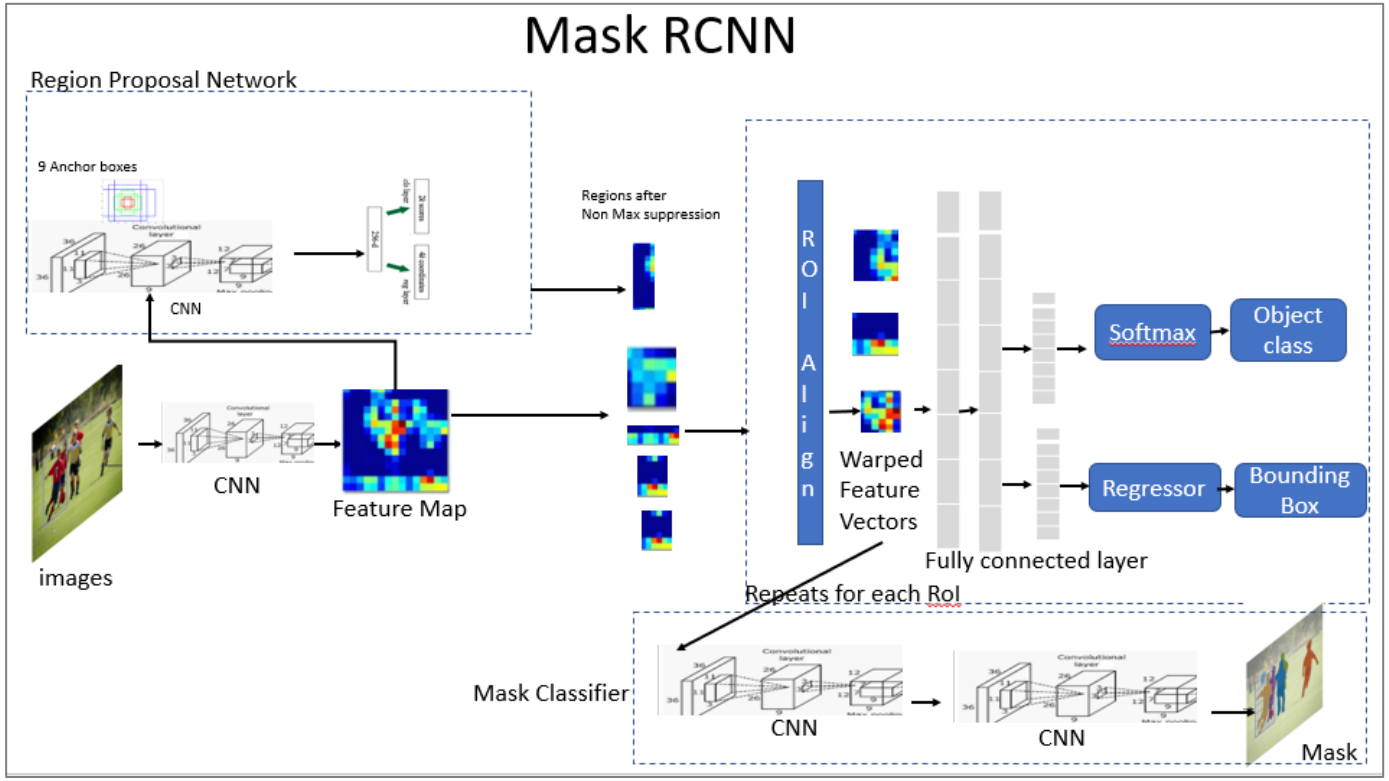
Model Overview

Model Advantages

Training Iterations

Testing and Evaluation

Mask RCNN Model Overview



Mask R-CNN model is divided into two parts

- Region proposal network (RPN) to proposes candidate object bounding boxes.
- Binary Mask Classifier to generate mask for every class

Architecture component overview

- Image is run through the CNN to generate the feature maps.
- RPN uses a CNN to generate the multiple Region of Interest (RoI) using a lightweight binary classifier. It does this using 9 anchors boxes over the image. The classifier returns object/no-object scores. Non Max suppression is applied to Anchors with high objectness score
- The RoI Align network outputs multiple bounding boxes rather than a single definite one and warp them into a fixed dimension.
- Warped features are then fed into fully connected layers to make classification using softmax and boundary box prediction is further refined using the regression model
- Warped features are also fed into Mask classifier, which consists of two CNN's to output a binary mask for each RoI. Mask Classifier allows the network to generate masks for every class without competition among classes

Mask RCNN Model Advantages

Advantages for Mask R-CNN

- Mask R-CNN was built using Faster R-CNN.
  - Faster R-CNN has 2 outputs for each candidate object, a class label, and a bounding-box offset.
  - Mask R-CNN is the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring extraction of a much finer spatial layout of an object.

- Mask R-CNN adds a branch for predicting an object mask (Region of Interest) in parallel with the existing branch for bounding box recognition.

- The key element of Mask R-CNN, is the pixel-to-pixel alignment, which is the main missing piece of Fast/Faster R-CNN. Mask R-CNN adopts the same two-stage procedure with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. This is in contrast to most recent systems, where classification depends on mask predictions.

- Mask R-CNN is simple to implement and train given the Faster R-CNN framework which facilitates a wide range of flexible architecture designs.

- Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

Mask RCNN Model Training Iteration # 1

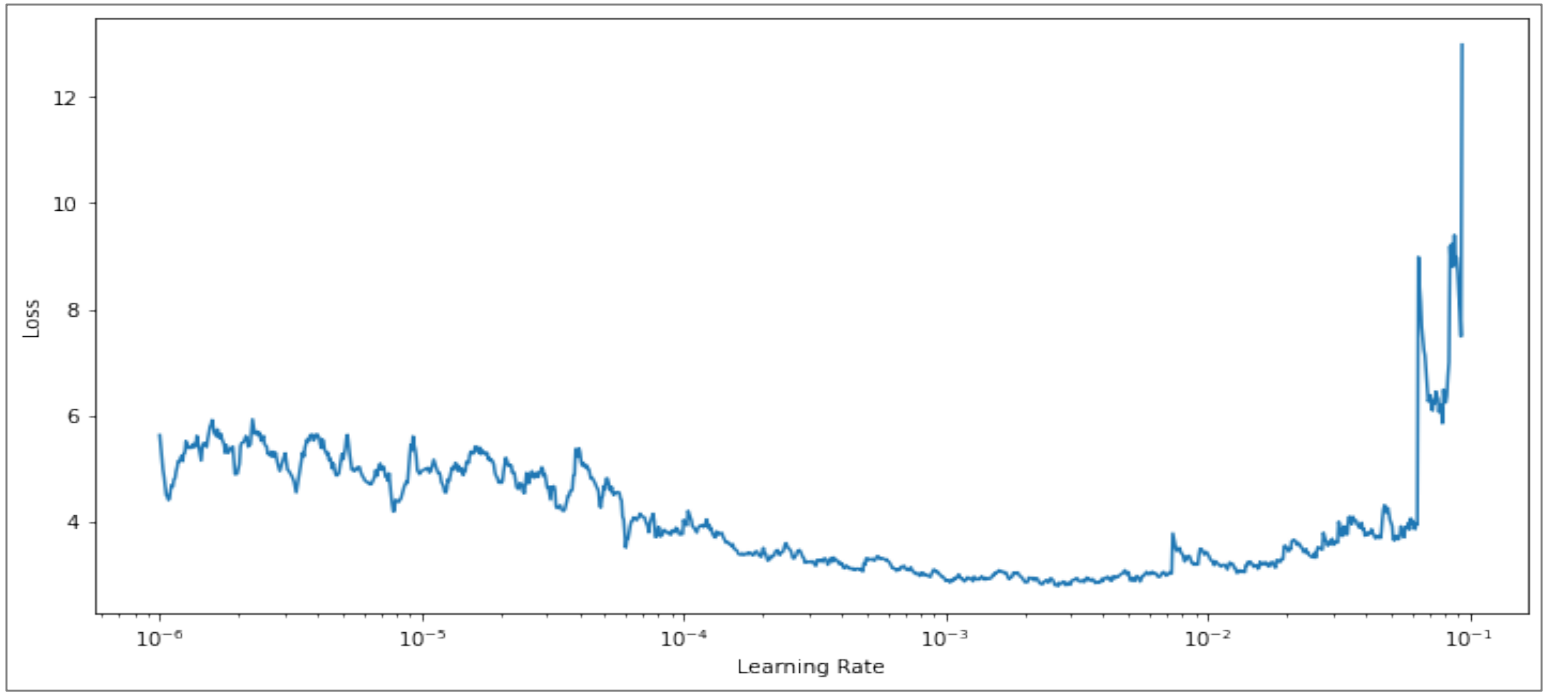
Objective – To determine best learning rate

Total params: 44,662,942
Trainable params: 44,603,678
Non-trainable params: 59,264

Summary

Learning rate of 1e-3 to 1e-2 seems to be the best learning rate

Learning Rate vs Loss



Mask RCNN Model Training Iteration # 5

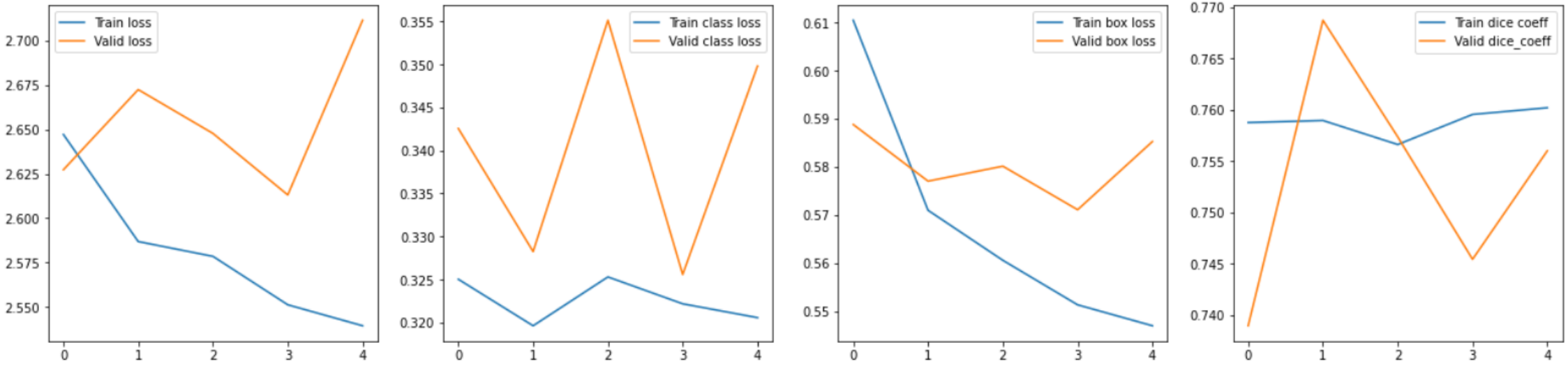
Objective – SGD Optimizer, LR = 0.001

Total params: 45,185,182
Trainable params: 21,069,086
Non-trainable params: 24,116,096

Summary

With learning rate 0.001, best epoch with minimum validation loss generated a dice co-efficient of 0.76

Training vs Validation loss, classification loss, bbox loss and dice-coeff



** Iterations 2, 3, 7 are captured in the notebook **

Mask RCNN Model Training Iteration # 6

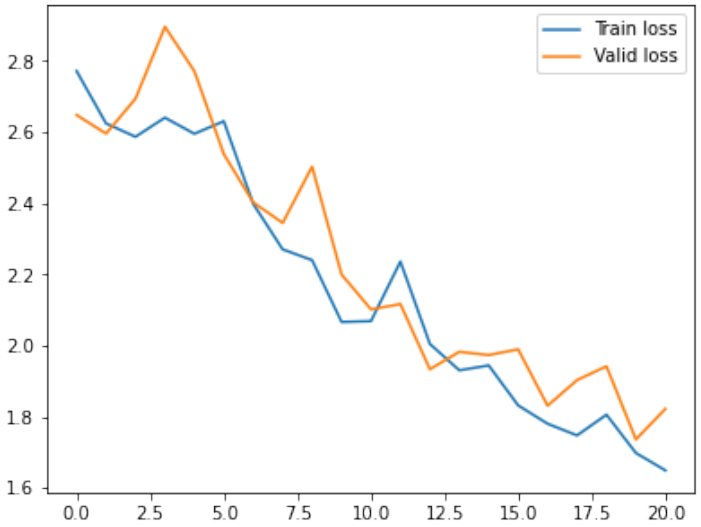
Objective – LAMB Optimizer, LR 0.002

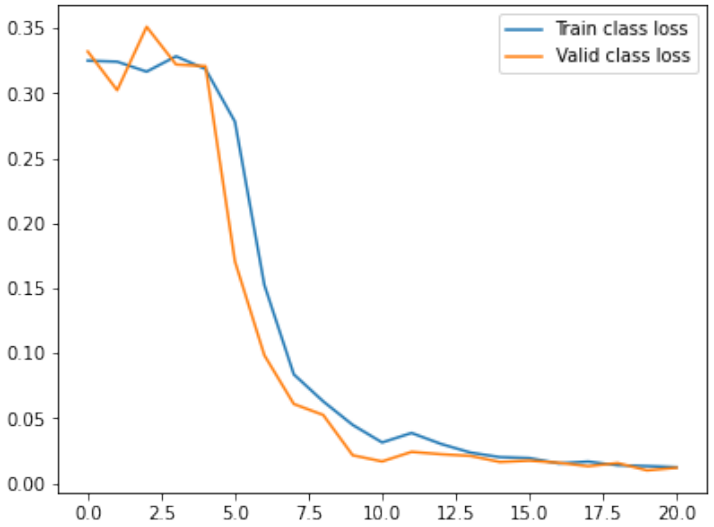
Total params: 45,185,182
Trainable params: 45,125,918
Non-trainable params: 59,264

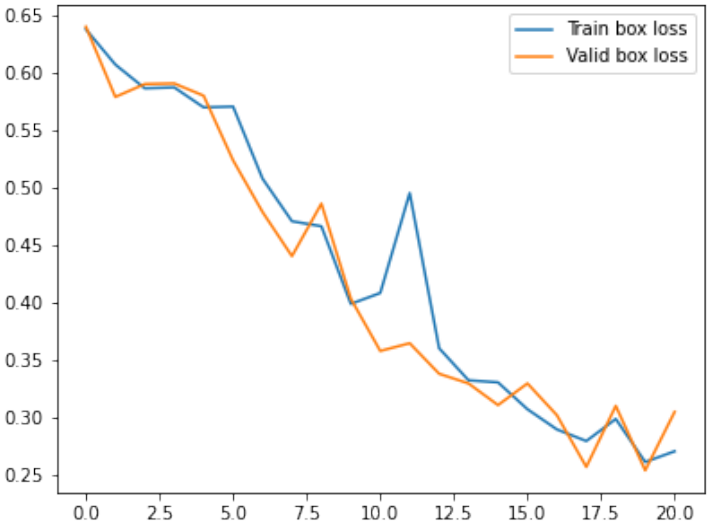
Summary

With learning rate 0.002, best epoch with minimum validation loss generated a dice co-efficient of 0.60

Training vs Validation loss, classification loss, bbox loss and dice-coeff





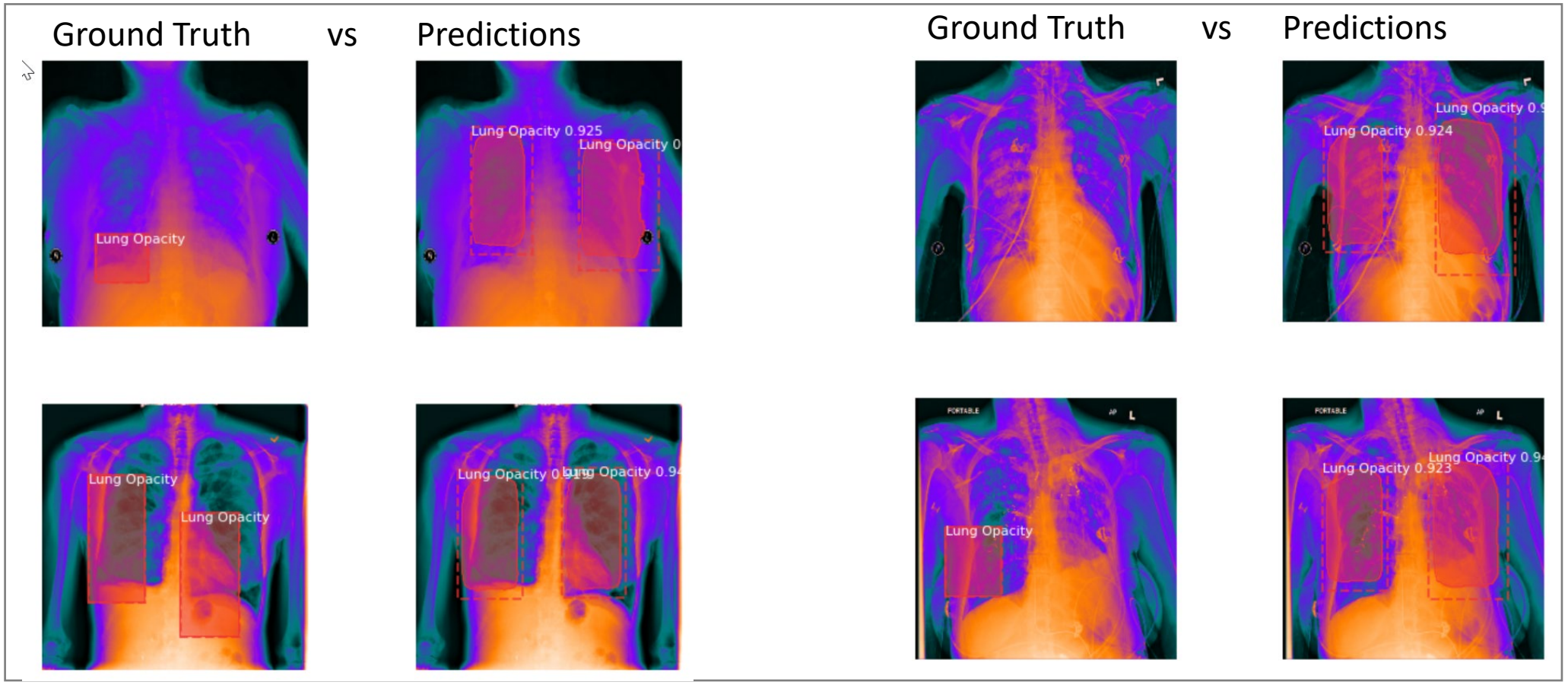


Mask RCNN Model Testing and Evaluation

Summary of Initial model testing:

- Model was able to detect lung opacity in test images with confidence set at 85%
- Model gave a dice co-efficient of 0.77 in the test run

Model Prediction



Mask RCNN Model – Important parameters

Parameter	Description	What did we evaluate	Final Observation
Batch Size	Batch size indicates the number of samples that the model goes through before it update the internal parameters.	Batch size of 64, 32, 16, 8	Batch size of 8 enabled the best learning in case of Mask RCNN
Epochs	Defines the number of times that model will iterate through the entire dataset	Epochs 5,10,15,30	Given the number of samples the number of epochs between 5 and 10 performed the best.
ROIs per Image	Defines the maximum number of ROIs the RPN will generate for a given image to feed classifier/mask	100, 200, 512	100 had a negative impact. 512 didn't have any positive impact. Hence 200 was kept as optimal
Loss_weights	Loss weights for more precise optimization of the model. These weights influence the weightage for each of the specific loss functions.	Class loss, bbox_loss and mask loss range from 0.1 to 1	Most optimal performance was derived when all weights were set to 1
Optimizer	Is an algorithm that updates the various parameters that can reduce the loss in much less effort.	SGD, Adam, LAMB	SGD appeared very erratic. Adam was resulting in vanishing gradients within the first few epochs. LAMB provided the best results.
Learning Rate	Is a hyper-parameter that determines the step size at each iteration while attempting to minimize the loss function	0.01 – 0.0001	Tried various ranges and an initial learning rate of 0.001 and decreasing it to 0.0001 provided the best result
Freeze Layers	Freeze the layers to disable training of layers as we were using pre-trained models.	Heads vs All	Overall only training the head provided the best result. But as we evaluated various optimizers, a combination of Head and All in subsequent iterations provided better result

Model comparison

Comparing various models and metrics

Final solution

Models	Metrics	Key Parameters
Mask RCNN (Initial result)	Dice Coefficient = 35%	• Learning Rate – 0.01 • Batch size - 32 • Optimizer - SGD
SSD	F1 score = 33.4% IoU = 31%	• Learning Rate - 3e-3, 3e-4, 3e-2 • width=300, height=300 • LR Decay - epoch^^(0.9) • Optimizers - Adam, RMSProp, SGD • Freeze different layers - Trainable params from 21M to 1.2M • Callbacks - LR Scheduler, Modelcheckpoints, Early stopping
Yolo v4 (Tensorflow)	IoU = 48.32 % mAP@0.50 = 68.33 %	• batch=64, subdivisions=32 • width=416, height=416 • learning_rate=0.001, momentum=0.9, decay=0.0005 • max_batches = 6000
Yolo v3 (Darknet)	IoU = 45%	• batch=64, subdivisions=32 • width=608, height=608 • learning_rate=0.001, momentum=0.9, decay=0.0005 • max_batches = 1000
Mask RCNN	Dice co-eff = 77% IoU = 55%	• Learning Rate – 0.001, 0.0001, 0.0002, Momentum - 0.9 • width=256, height=256 • Optimizers – SGD, Lamb • Freeze different layers – Heads only vs all. • Callbacks – Tensorboard, Modelcheckpoint, WandbCallback

Key Takeaways…

Implications

Limitations

Closing Reflections

Implications

Solution affecting the problem in the domain or business.

- The Solution outcome is highly dependent on the choice object detection model with use of transfer learning to fine-tune the models. Solution was based on Learning to set the optimizers, loss functions, epochs, learning rate, batch size, checkpointing, early stopping etc.
- The real business value can be derived
    - By Automating Pneumonia screening in chest radiographs and providing affected area details through bounding box, The Solution aims to Assist physicians to make better clinical decision making in certain functional areas of healthcare (eg. radiology).
    - By accelerating the diagnosis of positive and negative cases, it enables optimal utilization of clinical resources towards complex cases. The key dependency is on accuracy of classification.

List of Recommendations and with a certain level of confidence

- For accuracy, use Mask R CNN Model. Model was able to detect Lung Opacity in test images with 85% Confidence.
- If speed is the driver, YOLO and SSD models can be used for making predictions.

Limitations

Identify limitations of the solution:

- Data limitation - The data is highly imbalanced with more number of negative cases vs positive. This results in learning more biased toward the negative pneumonia cases.
- Image resolution - The data is in monochrome and lower clarity leading to limitation in learning. A higher fidelity image would result in better detection as edges wouldn't be blurry.
- Radiology specific pre-trained model - Currently the solution used pre-trained models like imagenet or coco dataset. A radiology specific trained weights would enhance the detection.

Does the model fall short in the real world?:

- The choice of models fits the object detection problem. But for the domain of clinical & healthcare domain, there would be a need to fine tune the models to provide better accuracy.
- Currently the model is not ready for real world yet because there are high number of false positives, thereby having a low efficacy in the real world. The precision is low because of the high number of false positives. It would require further tuning and evaluation of alternative backbone.

Potential Ideas/Steps to enhance the solution:

- Explore Resnet101 instead of Resnet50 for Mask RCNN solution
- Explore Chexnet instead of Darknet backbone since it is specifically trained on chest radiographs
- Enhance pre-processing to eliminate outliers that may influence the learning
- Additional augmentation of images to increase positive samples
- Potentially improve training performance with better hardware capabilities and use all training samples

Closing Reflections

Key Lessons Learned from the Process:

- Working with Dicom images and converting them to jpeg files
- Various EDA and data visualization techniques
- Various augmentation techniques using imgaug libraries and internal to models like SSD/YOLO
- Hands on experience on models like Mask-RCNN, YOLO with Darknet backbone and SSD models
- Work with libraries like Tensorboard and WandB.ai
- Techniques to identify LR ranges using LR Finder
- Work with various optimizers like SGD, Adam, Lamb etc.
- Fine tuning of Hyper-parameters like learning rate, steps per epoch, batch size, epochs etc. and understood the impact of them
- Working with pre-trained models and weights, and create custom object detectors for RSNA dataset using YOLO v3/v4

What Could Have Been Done Different:

- Explored more models supporting radiology image processing (CheXnet)
- Handled outlier and imbalances in the dataset
- Explored more augmentation techniques

Thank You!!!