Pneumonia Detection Challenge

Project Report

Submitted By: Group 13

PRESENTED TO	Great Learning
PRESENTED BY	Group 13 – Aug C – Great Learning
DATE	17 August 2020
VERSION	1.4

Group 13 – Contacts

Name	Phone	Email
Sonal Sharma	+91 88102 64764	Sonal_1996@yahoo.com
Chandrashekar C	+91 98867 94586	chandrashekar.c@gmail.com
Susanta Mondal	+91 70440 55951	Susanta.221285@gmail.com
Gopinath Bailur	+91 99400 78996	gbailur@gmail.com

CHANGE LOG			
DATE	VERSION	AUTHOR	CHANGE DESCRIPTION
24th July 2020	1.0	Group 13	Initial Draft
27th July 2020	1.1	Group 13	Updated based on the feedback with the mentor
15th Aug 2020	1.2	Group 13	Final Report – First Draft
16th Aug 2020	1.3	Group 13	Updated Results from mask R-CNN and Yolo
17th Aug 2020	1.4	Group 13	Updated comments based on review

Table of Content

1. Overview

Pneumonia is a form of acute respiratory infection that affects the lungs. The lungs are made up of small sacs called alveoli, which fill with air when a healthy person breathes. When an individual has pneumonia, the alveoli are filled with pus and fluid, which makes breathing painful and limits oxygen intake.

Following are some of the key facts about Pneumonia which needs at most attention to address the problem proactively.

- Pneumonia accounts for 15% of all deaths of children under 5 years old, killing 808 694 children in 2017.
- Pneumonia can be caused by viruses, bacteria, or fungi.
- Pneumonia can be prevented by immunization, adequate nutrition, and by addressing environmental factors.
- Pneumonia caused by bacteria can be treated with antibiotics, but only one third of children with pneumonia receive the antibiotics they need.

Per WHO, Pneumonia is the single largest infectious cause of death in children worldwide. Pneumonia killed 808 694 children under the age of 5 in 2017, accounting for 15% of all deaths of children under five years old. Pneumonia affects children and families everywhere, but is most prevalent in South Asia and sub-Saharan Africa. Children can be protected from pneumonia, it can be prevented with simple interventions, and treated with low-cost, low-tech medication and care.

The WHO and UNICEF integrated Global action plan for pneumonia and diarrhoea (GAPPD) aims to accelerate pneumonia control with a combination of interventions to protect, prevent, and treat pneumonia in children with actions to:

- protect children from pneumonia including promoting exclusive breastfeeding and adequate complementary feeding.
- prevent pneumonia with vaccinations, hand washing with soap, reducing household air pollution, HIV prevention and cotrimoxazole prophylaxis for HIV-infected and exposed children.
- treat pneumonia focusing on making sure that every sick child has access to the right kind of care -- either from a community-based health worker, or in a health facility if the disease is severe -- and can get the antibiotics and oxygen they need to get well;

Based on research we have made an attempt to look at the Chest Radiography to identify the Lung Opacity and quickly help Clinical specialists to take right decisions to drive proactive measure to cure and help avoid the spread of this decease to larger extent.

2. Abstract

This project is aimed at detecting Pneumonia by locating the lung opacities on the Chest radiographs. This process can help identify the problem at an early stage as well as helps the Clinical analysis much faster and drives better decision making. This process of Pneumonia detection will be done by looking at several thousand images of Chest radiographs taken from past wherein the analysis and desired results were identified by the specialists. These past datapoints will become the indicators and these images will be processed through the Computer Vision Technology of deep learning to capture every details by which a Deep Learning Algorithm will be built.

The new patients data will be fed to this model which detects the Lung Opacity indication along with its location such that Clinical specialists will be able to confirm diagnosis quickly and can help in taking respective decisions quickly to move forward with the next steps of the treatment.

Deep neural networks models have conventionally been designed and experiments were performed upon them by human experts in a continuing trial and error method. This process demands enormous time, knowhow and resources. To overcome this problem, a novel but simple model is introduced to automatically perform optimal classification tasks with deep neural network architecture .

The Neural network architecture was specifically designed for Pneumonia image classification tasks. The proposed technique is based on the CNN algorithm, utilizing set of neurons to convolve on a given sample images to extract relevant features from them. This is demonstrated through validating the accuracy of the detection along with the objective to reduce the loss while the network is learning the details.

As part of this project, we will be demonstrating the outcome with 4 different model architecture along with their outcome in each of the model and provide the commentary for each of the models developed.

3.Project Objective

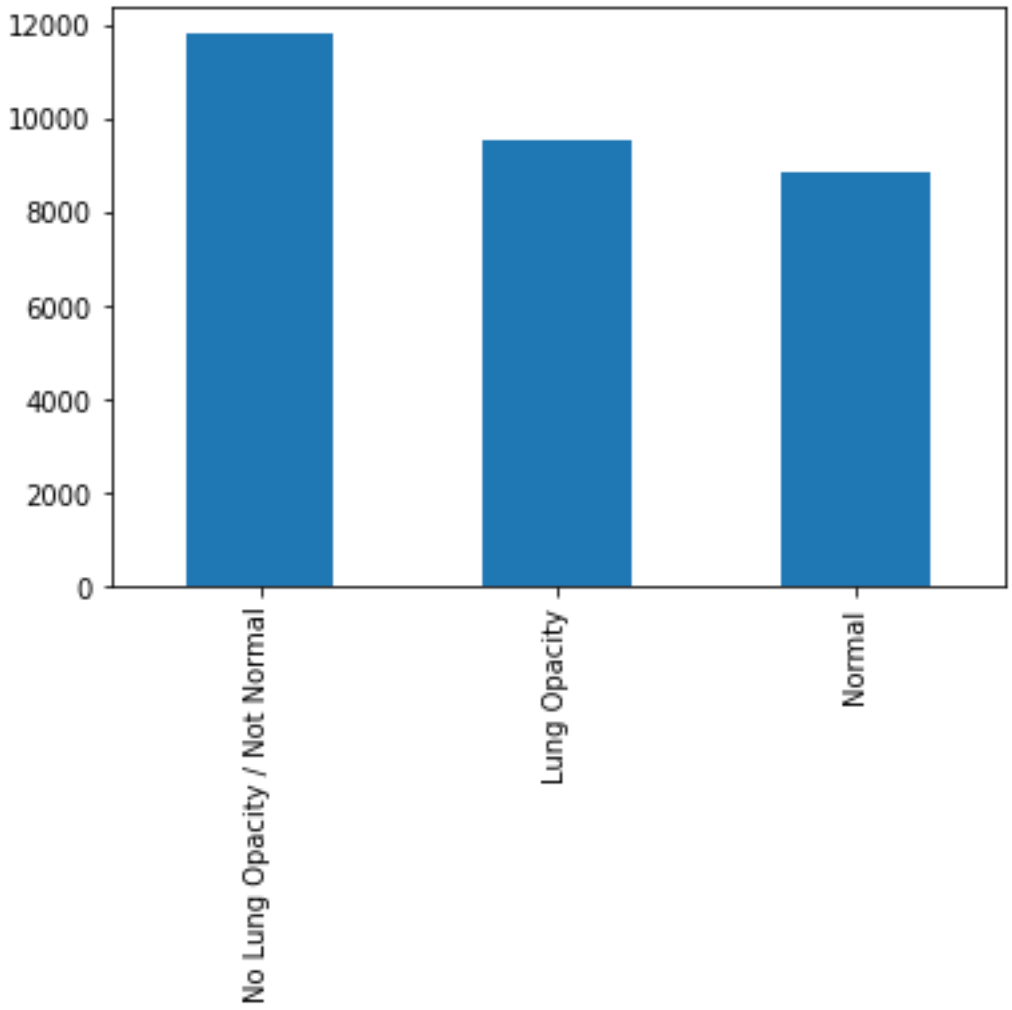
The objective of this capstone project is to build a Pneumonia detection system to locate the position of the inflammation in a Chest Radiography.

Based on the dataset provided, we will have to do the following steps to work towards building the final model and validate and results.

- Validate the images and respective bounding box coordinates.
- Extract all the features from the images and build the csv file for processing further.
- Perform Exploratory Data Analysis to validate all the data points and build insights
- Based on the project objective – isolate the dataset and accordingly images as well which are fully unique by removing the duplicate records in the dataset based on target variable.
- Build the model with different architectures and showcase the accuracy and showcase the right model to approach the problem description

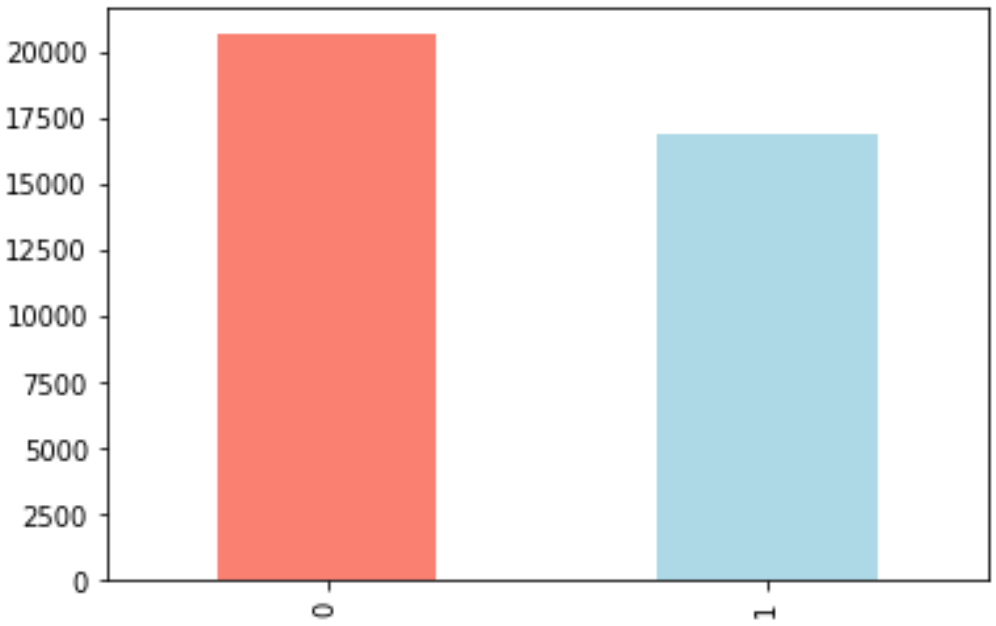
4. EDA Inference and Data pre-processing

1) Based on the sample dataset provided – we see that there are 3 class values present..



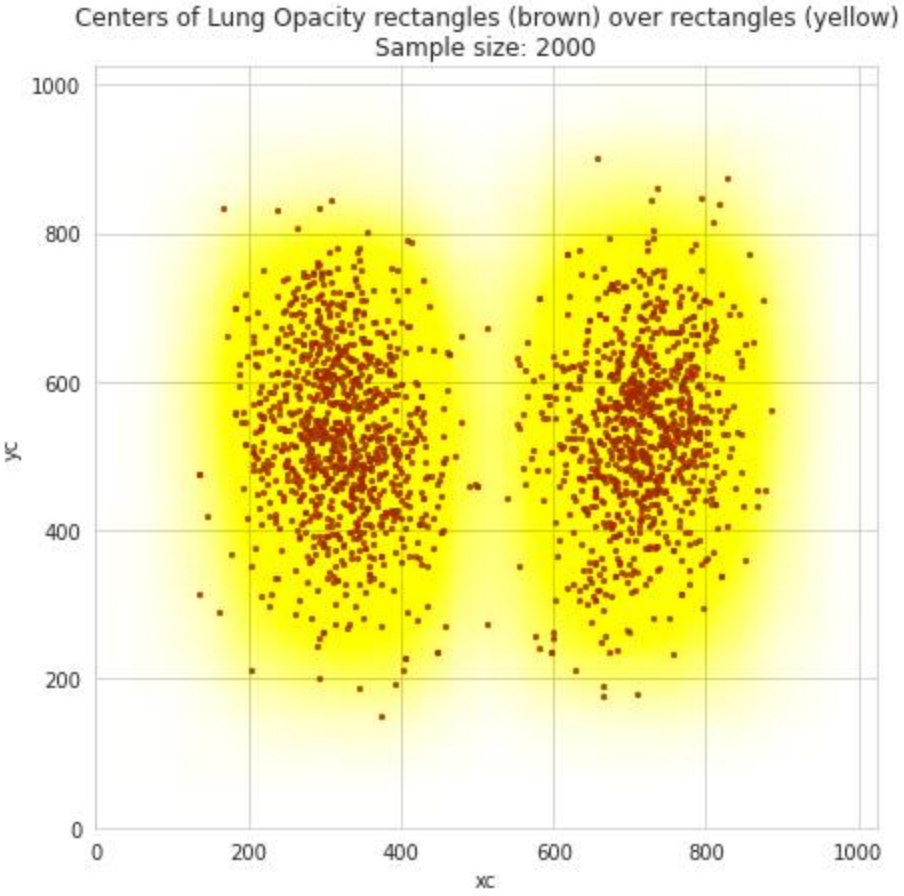
Our objective would be look at images having Lung Opacity and identify the right bounding box to locate the area of inflammation, hence we may need to club them into Lung Opacity and others as another single category.

2) Post merging both the documents – class information with the labels document we see the following distribution on the "Target" variable.



We observe there are more records not having lung opacity issues, which may lead to bias.

3) We observe the following heat map making the visualization where in the Target = 1



The scatter plot with brown dots shows the areas of inflammation highlighted which needs to be learnt to identify the issues.

4) Following are set of Metadata that we are able to notice from each of the chest X-ray's provided.

```
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
```

5) Of the above metadata – basis the observation of key fields names and its values, we decided to consider following metadata to form the CSV file for validating and processing further.
    - 'patientId'
    - 'Modality'
    - 'PatientAge'
    - 'PatientSex'
    - 'BodyPartExamined'
    - 'ViewPosition'
    - 'ConversionType'

- 'Rows'
- 'Columns'
- 'PixelSpacing'

6) Processing the dicom images, we see the following

ID: dc4d3a01-94f5-4e5a-a843-1c60dfd8b352
Modality: CR Age: 40 Sex: M Target: 1
Class: Lung Opacity
Window: 206.0:553.0:211.0:170.0

ID: 958f6fd1-f377-4a55-bb38-7a9db2fd79dc
Modality: CR Age: 59 Sex: F Target: 1
Class: Lung Opacity
Window: 260.0:114.0:269.0:596.0

ID: 6f988c8c-ad47-4694-b5c9-bf572cc7c23a
Modality: CR Age: 51 Sex: F Target: 1
Class: Lung Opacity
Window: 611.0:413.0:197.0:378.0

ID: bf6071e6-0a92-4896-83bf-f0f24cd1b4d6
Modality: CR Age: 44 Sex: M Target: 1
Class: Lung Opacity
Window: 600.0:481.0:192.0:289.0

ID: f40ba5da-1ef0-49d4-bb19-492892b41704
Modality: CR Age: 38 Sex: M Target: 1
Class: Lung Opacity
Window: 239.0:534.0:237.0:230.0

ID: be2a8801-3cc0-4c24-a73a-a1e13ff94948
Modality: CR Age: 68 Sex: F Target: 1
Class: Lung Opacity
Window: 550.0:434.0:240.0:432.0

ID: 57835e6c-04e7-4e8f-9570-da0a6bef6b31
Modality: CR Age: 28 Sex: M Target: 1
Class: Lung Opacity
Window: 573.0:358.0:281.0:410.0

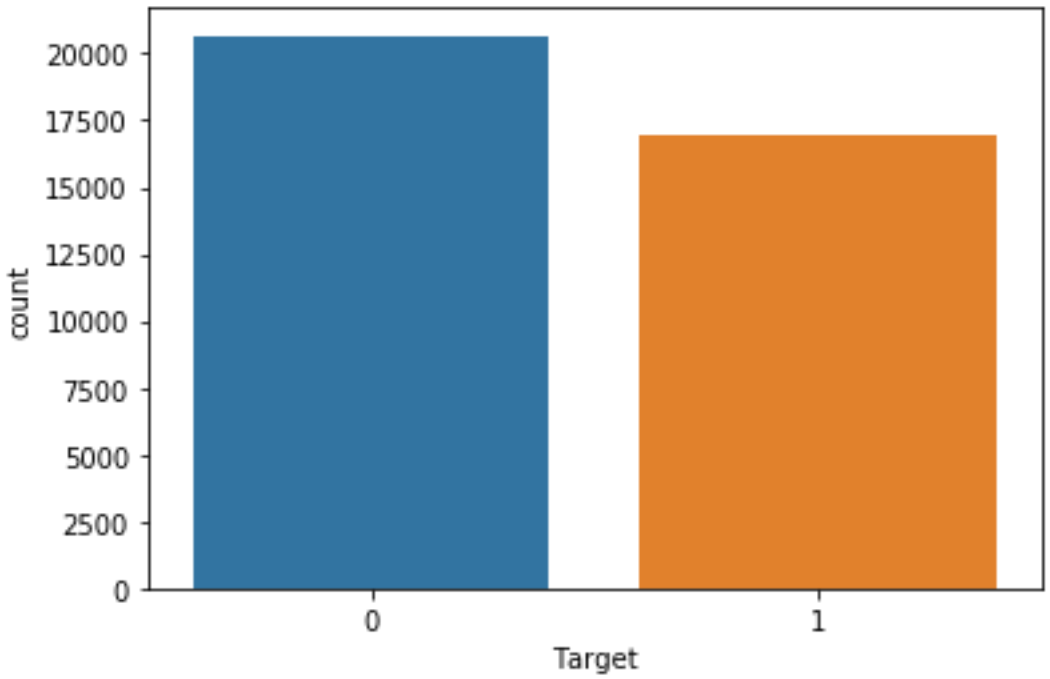
ID: 3d413032-b091-4f90-a131-17bc1eeb0647
Modality: CR Age: 43 Sex: M Target: 1
Class: Lung Opacity
Window: 398.0:512.0:147.0:177.0

ID: 22ce653c-e7f4-45d5-9c5e-4abdc753b1b0
Modality: CR Age: 27 Sex: M Target: 1
Class: Lung Opacity
Window: 647.0:421.0:142.0:223.0

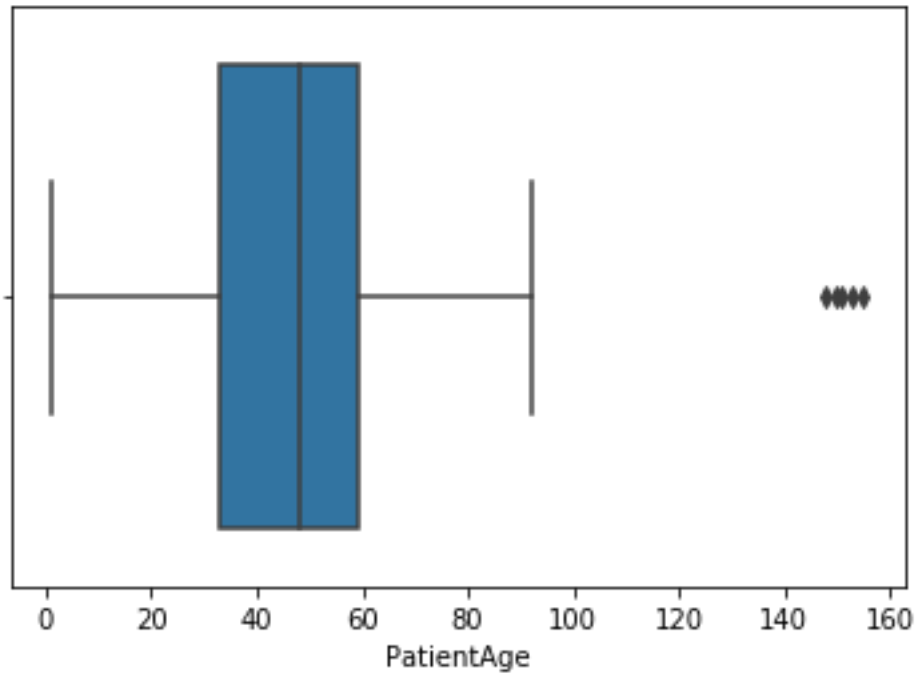
We observe all the images are 1024/1024 shape and it needs to be reshaped before feeding them into the model.

Based on the processed CSV file from the metadata – EDA was performed on this and below are the findings and also pre-processing steps taken to perform the data analysis

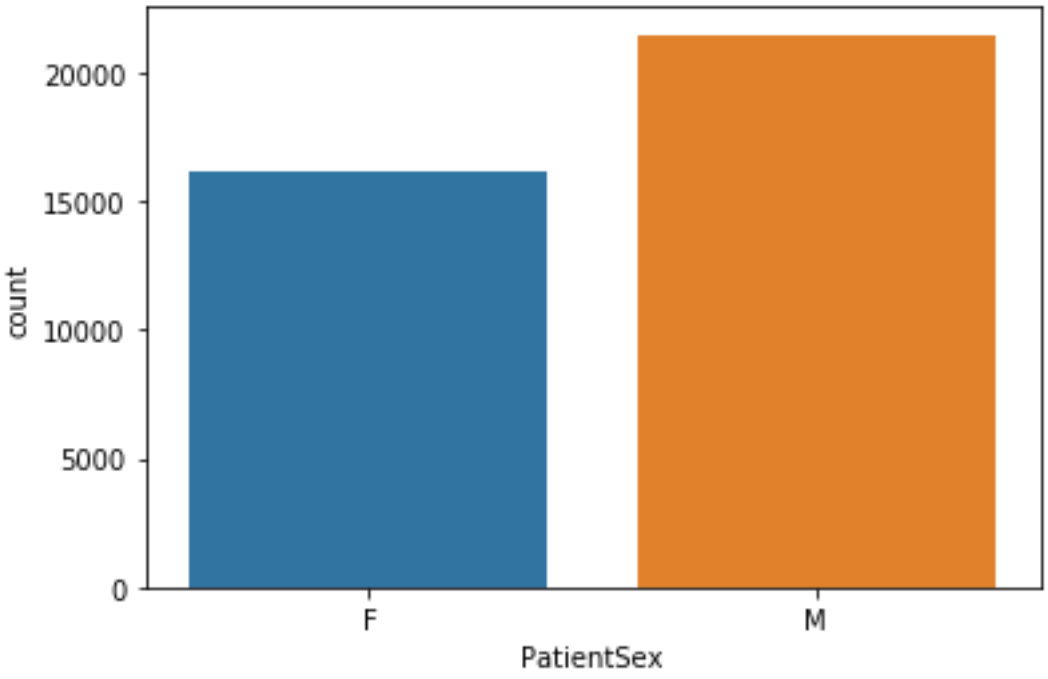
- EDA performed on the data taken out from each of parameters extracted from the images and clubbed against their patient ID's to validate them for its accuracy, impact based on availability of information overall. Below are some of the key findings.
- There are totally 37,629 records available in the dataset where in 16 fields were extracted for validation
- For Pneumonia detection - "Target" Field provides the classification. 0 : No Lung Opacity, 1 : Lung Opacity



- "class" Field provides 3 groups. However when its seen against Target classification and respective coordinates availability - it doesn't seems to isolate between Normal and Not Normal cases. There are may be other problems, for the purpose of this exercise this observation will be ignored
- "Target" has 20,672 records/images which are Not having Lung Opacity and 16,957 having lung Opacity. We observe 55% of the images doesn't have lung opacity and only 45% having Lung Opacity - This may create imbalance in prediction tending towards not having Lung Opacity - Need to observe this further for duplicate records and plan for data augmentation.
- "PatientAge" - We observe 5 records having ages above 100 - which should be dropped

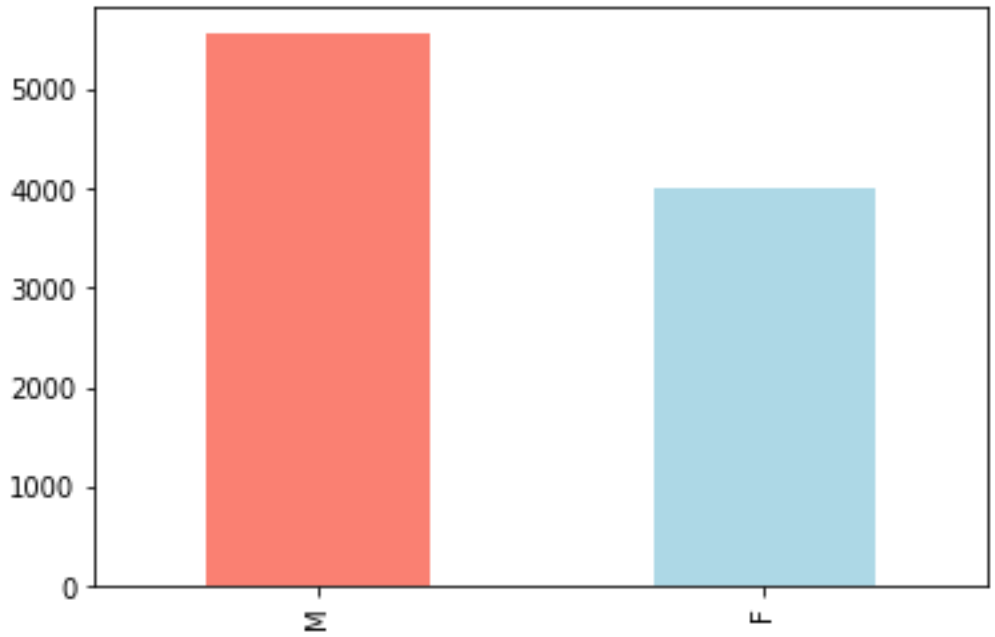


- "PatientSex" - We observe 55% of male records in the total count. This may not be a factor for recognition hence decided to continue even though there is imbalance..

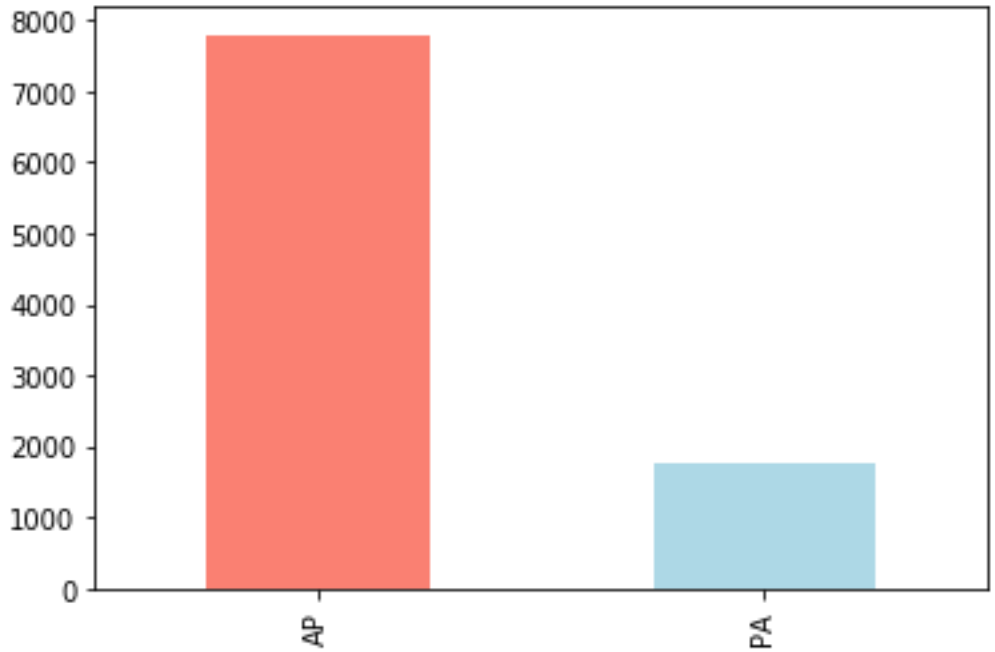


- We observe 9555 records are unique - which needs to be considered for model building
- For processing further – we have taken the ROWS having Target = 1, we found that there are 16K records

- Post processing for Duplicate records – we see that the record count comes to 9555 records..
  1 9555
  Name: Target, dtype: int64
- Of the 9555 records – below is the distribution on M / F



- Since Male records are more – this may create data imbalance even though it may not be crucial at this time this needs to be validated.
- With the column "ViewPosition" we notice the below findings..



With respect to Column "ViewPosition" correlated with the Target value of 1 - There is high imbalance that we observe with the above information having AP (Anterior-Posterior). One way to interpret this target unbalance is that patients that are imaged in an AP position are those that are more ill, and therefore more likely to have contracted pneumonia. Note that the absolute split between AP and PA images is about 50-50, so the above consideration is extremely significant

- Validating the target with age group – we observe the below pattern.

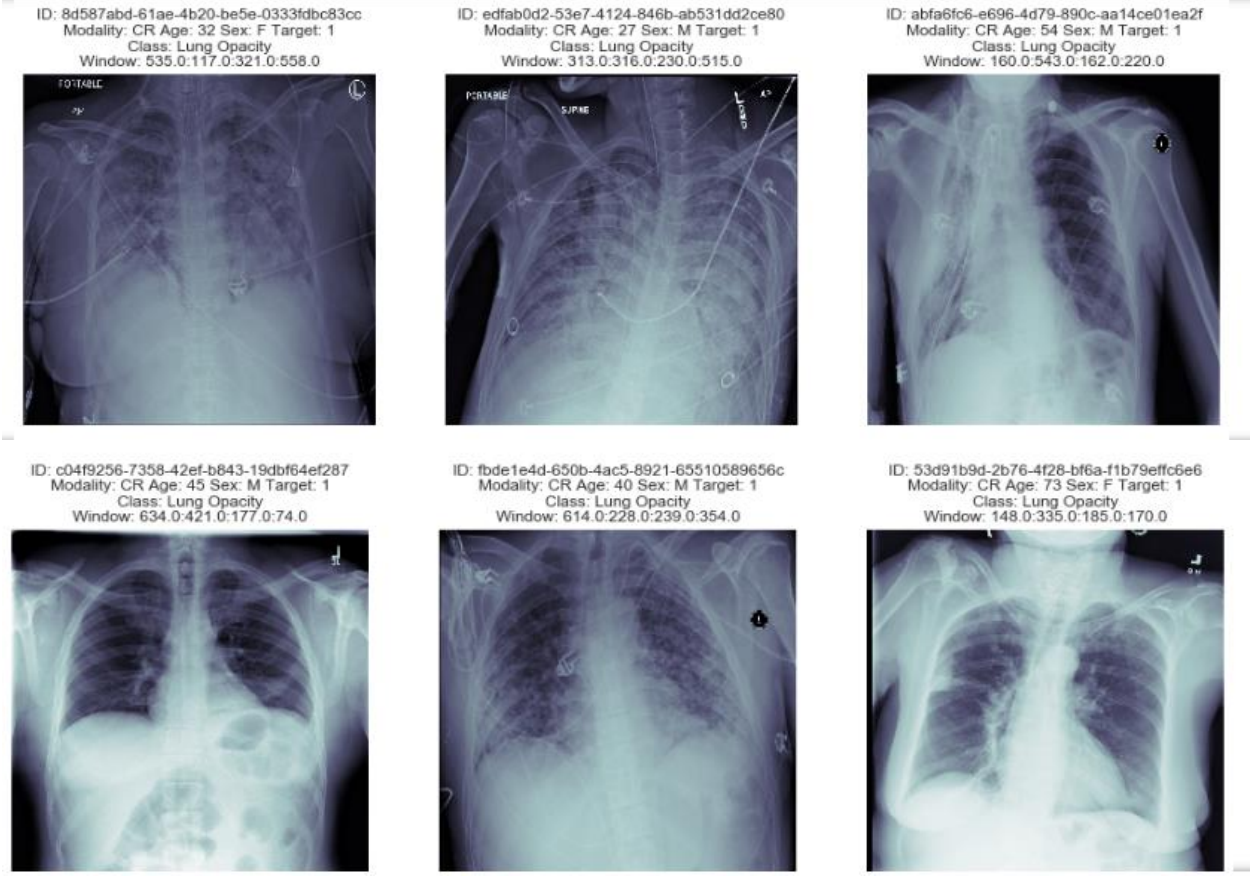
	PatientAge	count
0	0-10	219
1	11-20	602
2	21-30	1308
3	31-40	1591
4	41-50	1675
5	51-60	2226
6	61-70	1313
7	71-80	518
8	81-90	100
9	91-100	3
10	more than 100	0

Between age from 21 to 70 – The problem is highly detected and specifically 51 to 60 has m ore occurrence. We also observe some of the outlier which may needs to be dropped.

- We were able to extract and visualize the dicom images provided and visual representation are detailed below.

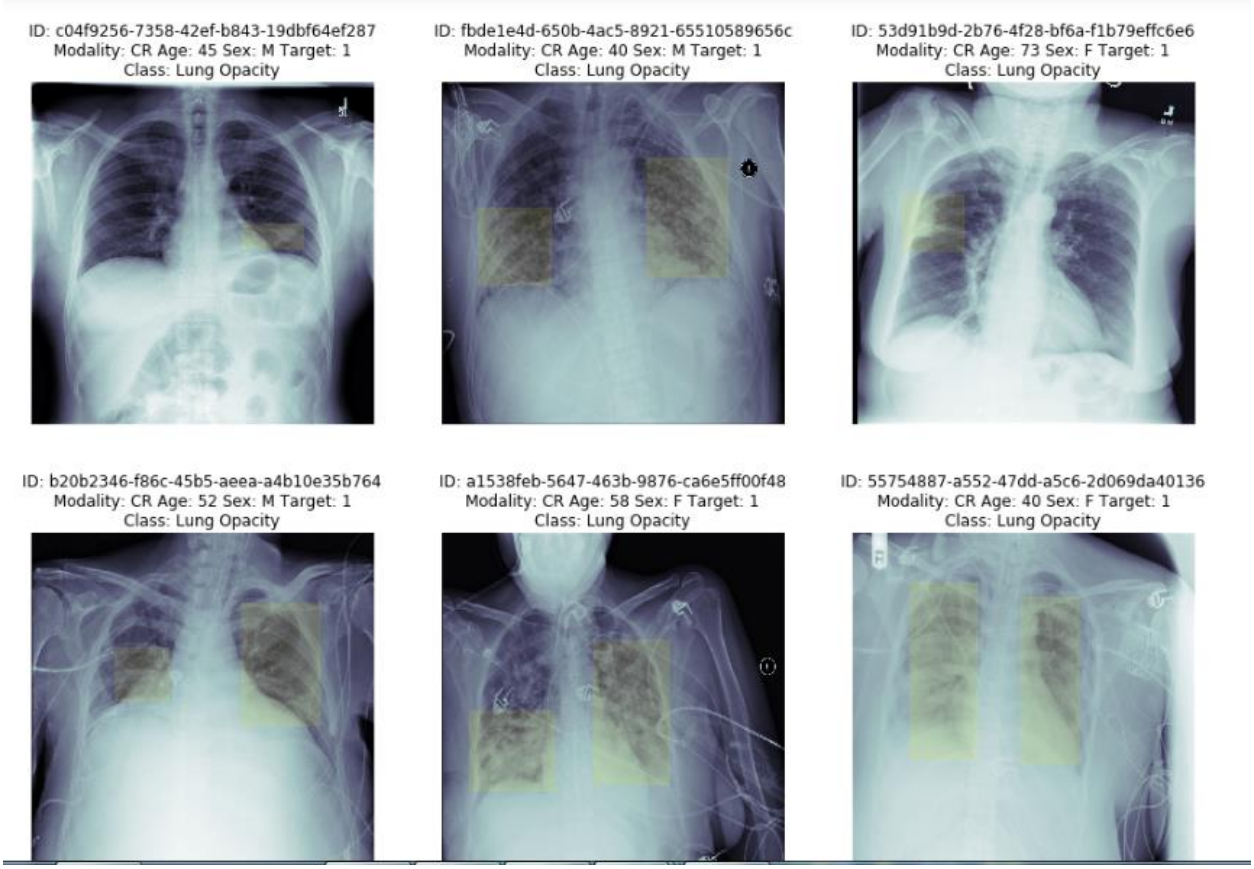
RAW Images:

```
1 show_dicom_images(train_class_df[train_class_df['Target']==1].sample(9))
```



RAW Images along with its bounding boxes (for the same samples above):

```
1 show_dicom_images_with_boxes(train_class_df[train_class_df['Target']==1].sample(9))
```



- Post this process as part of pre-processing we also associated the proper labelling for the respective images identified for processing further, we also validated this against the total records of 9555 to make sure that images along with associated labels are properly matched.

5. Interim Approach on Model building and Model Selection

Deep neural networks models have conventionally been designed and experiments were performed upon them by human experts in a continuing trial and error method. This process demands enormous time, knowhow and resources. To overcome this problem, a novel but simple model is introduced to automatically through CNN models.

We also noted that constructing and training a complex deep learning model from scratch is mostly infeasible due to the lack of hardware infrastructure. Therefore, we decided to exploits the idea of transfer learning which is the improvement of learning in a new prediction task through the transfer of knowledge from a related prediction task that has already been learned. This will improve the current computer vision methods based on the use of deep learning to diagnose X-rays images more effectively. By utilizing convolutional neural networks re-trained with our obtained data, we would like to experiment them to achieve the greater classification accuracy.

We decided to develop the following models as part of our experiment and present our findings with details.

- MobileNet
- YOLO
- Mask R-CNN
- SSD

Following section details the MobileNet Interim Model implementation details:

As we begin our MobileNet implementation by adopting to the Transfer learning operation, we selected the model architecture through Tensor Hub which can be referenced in this URL https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4

TensorFlow Hub 2.0 allows the easier way to select the appropriate model based on our requirement and provides us the directional inputs on using that further in our model building exercise. We wanted to attempt this as a new feature to explore during this project cycle and selected Version 4 for our development purposes.

Some of the key steps / decisions taken as part of model development is detailed as below.

Pre-processing data:

1) Pre-process the image by converting them from Dicom to JPG images

2) Convert the images into Tensors
3) Resize the image from 1024/1024 to 224 / 224 as expected by MobileNet architecture

The above steps are defined as functions so that it returns the appropriate images.

Tuning our data into batches:

To make the model development efficient and faster to visualise we set the batch size to 32. Say, we are process 1000+ images they all may not fit into the memory, hence decided to set this to 32 per batch and this can be modified based on trials.

Also, in order to use TensorFlow effectively, we need our data in the form for Tensor Tuples which looks like (image, label). For which we have defined a function "get_image_label" which does all the steps of image conversion, processing, reshaping and returns the images and corresponding labels.

Building the model:

Below steps were followed to build the model..

```
# Setup input shape to the model
INPUT_SHAPE = [None, IMG_SIZE, IMG_SIZE, 3] # batch, height, width, color channelsl

# Setup output shape of our model
OUTPUT_SHAPE = 4

# Setup model URL from TensorFlow Hub
MODEL_URL = "https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4"#
@param ["https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4"]
```

NOTE: Model URL was provided to direct to the mobilenet architecture that we decided to use going forward..

The model summary is defined below:

Initialize the model and print summary

```
[ ] model = create_model()
    model.summary()

Building model with: https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer_17 (KerasLayer)  multiple                  4363712
_________________________________________________________________
dense_16 (Dense)             multiple                  7172
=================================================================
Total params: 4,370,884
Trainable params: 7,172
Non-trainable params: 4,363,712
_________________________________________________________________
```

Defining callback:

We have setup the Tensorboard feature to define the call back and below steps are taken to address them.

- Load the tensorboard notebook extension
- Create tensorboard callback – by which we will able to save logs to a directory and pass it to our model during fit
- To visualize our training logs we will use %tensorboard magic function

Defining Early stopping:

We have also defined early stopping to ensure we stop the process if there is a overfit scenario and/or loss value is not improving much. This way we will able to validate and re-run the model by changing right hyper parameter for better accuracy outcome.

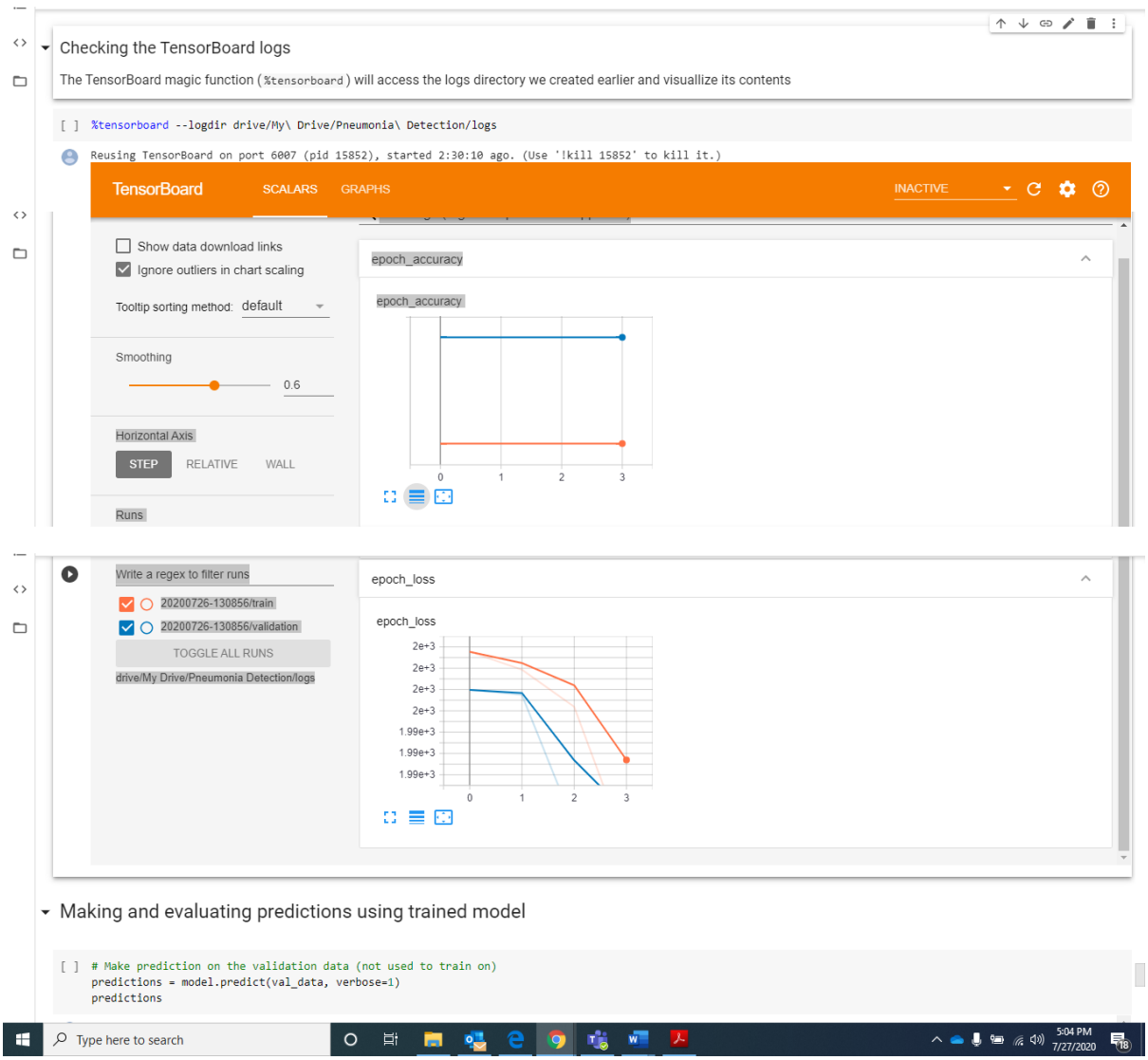
Running the model:

We have set the epochs to 100 to start with and called the fit function to execute the model.. Below is our INTERIM Observation..

```
[ ] # Fit the model to the data
    model = train_model()

Building model with: https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4
Epoch 1/100
25/25 [==============================] - 44s 2s/step - loss: 2000.5381 - accuracy: 0.2700 - val_loss: 1996.9641 - val_accuracy: 0.2950
Epoch 2/100
25/25 [==============================] - 44s 2s/step - loss: 1998.8617 - accuracy: 0.2700 - val_loss: 1996.5079 - val_accuracy: 0.2950
Epoch 3/100
25/25 [==============================] - 44s 2s/step - loss: 1995.3879 - accuracy: 0.2700 - val_loss: 1984.2837 - val_accuracy: 0.2950
Epoch 4/100
25/25 [==============================] - 43s 2s/step - loss: 1982.1591 - accuracy: 0.2700 - val_loss: 1979.6708 - val_accuracy: 0.2950
```

Validating the logs:



We did get into some of the challenges in progressing with the image translation and hence we observe high loss and very low accuracy to start with. We will continue further to address the challenges with the above approach to drive a very high accuracy when we finally conclude on this.

6. Model Development

We continued the effort to build the model further and this section details model building process and associated outcome.

6.1 MobileNet

MobileNet represent the efficient models for embedded computer vision applications. MobileNet' s are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks. MobileNet are built primarily from depthwise separable convolutions and subsequently used in inception models to reduce the computation in first few layers.

6.1.1 MobileNet Application reference

The code implementation path:
https://github.com/julycapstone2020/Pneumonia-Detection-Challenge/blob/master/notebook/Pneumonia_Detection_Challenge_5000.ipynb

Model was trained on 5000 images for this purpose with following parameters

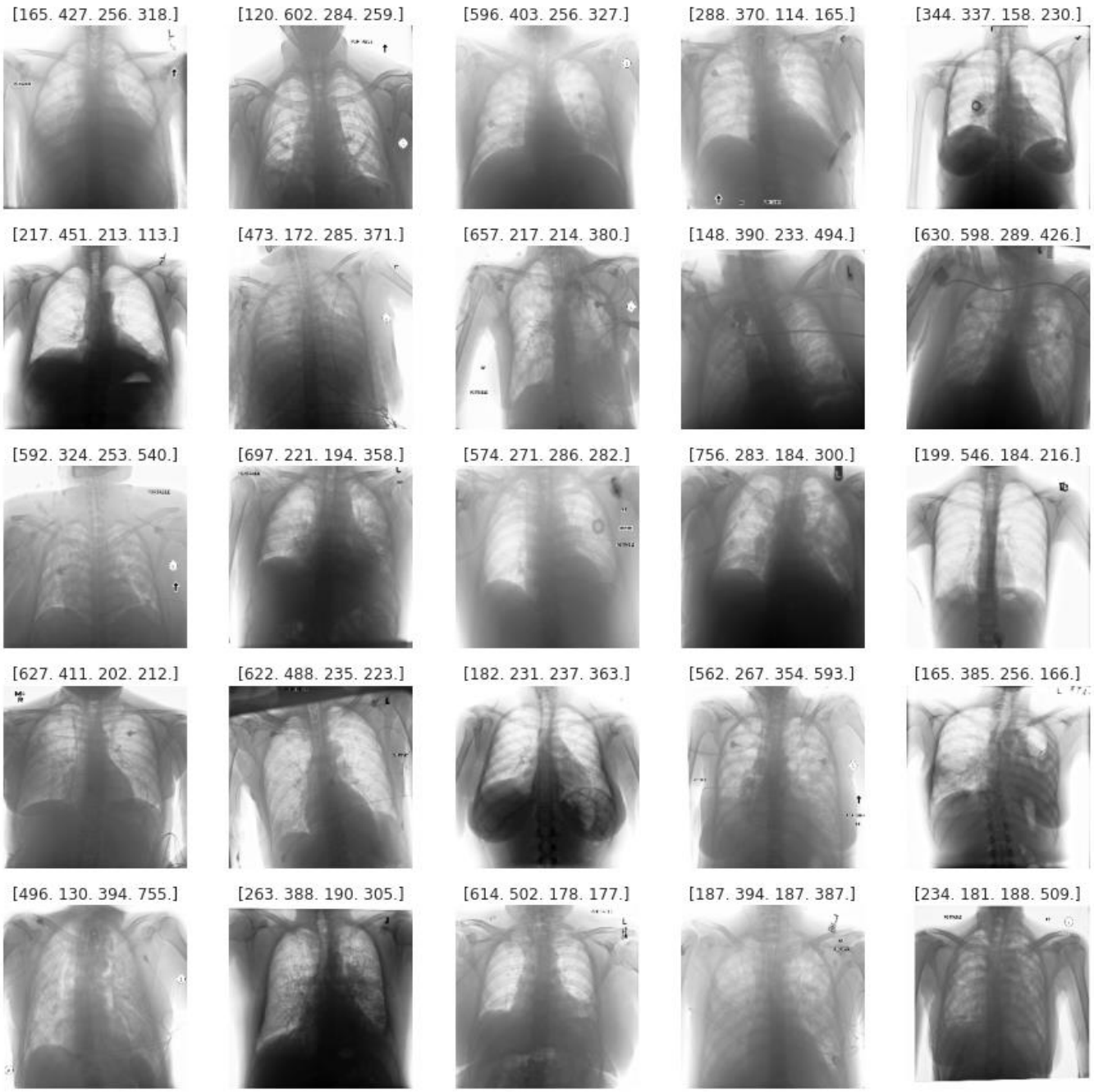
- Train / test split of 80:20
- Batch size of 32
- Loss function: Categorical cross entropy
- Metrics: Accuracy
- Optimizer: Adam
- Activation function: Softmax

As mentioned in the interim approach this model was selected through the TensorHub Portal. Details of this will be available in below link.
https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4

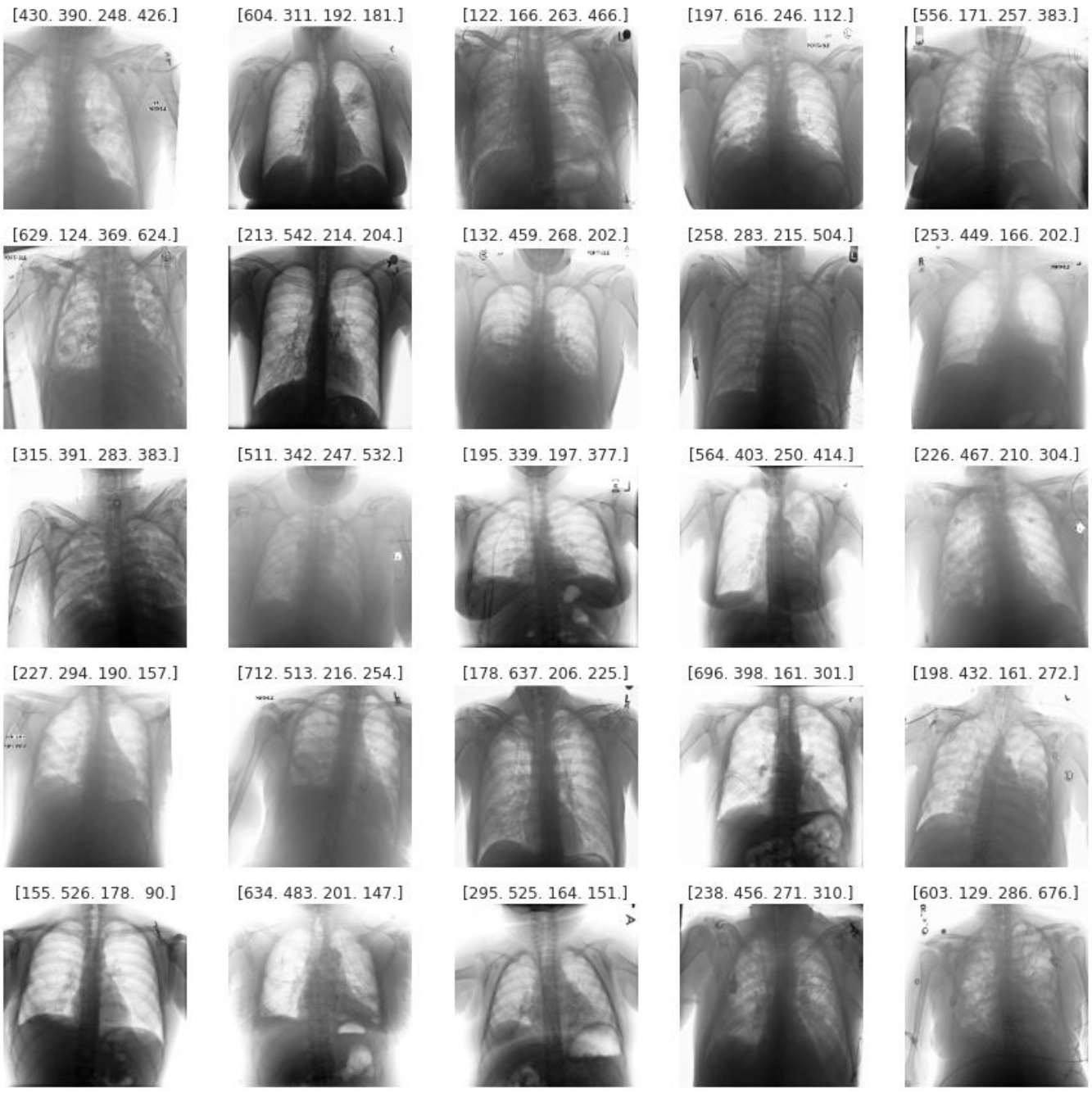
6.1.2 Sample image representations

Visualizing the first 25 images from the training batch:

```
show_25_images(train_images, train_labels)
```



Visualizing the first 25 images from the validation batch:
show_25_images(val_images, val_labels)



6.1.3 Model Architecture

This section details the core layers that MobileNet is built on which are depthwise separable filters. This is a form of factorized convolutions which considers a standard convolution into a depthwise convolution and a 1x1convolution called a pointwise convolution.

MobileNet Body Architecture is as depicted below

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Current implementation of Model Summary:

```
Building model with:
https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     multiple                  4363712
_________________________________________________________________
dense (Dense)                multiple                  7172
=================================================================
Total params: 4,370,884
Trainable params: 7,172
Non-trainable params: 4,363,712
```

6.1.4 Results and conclusion

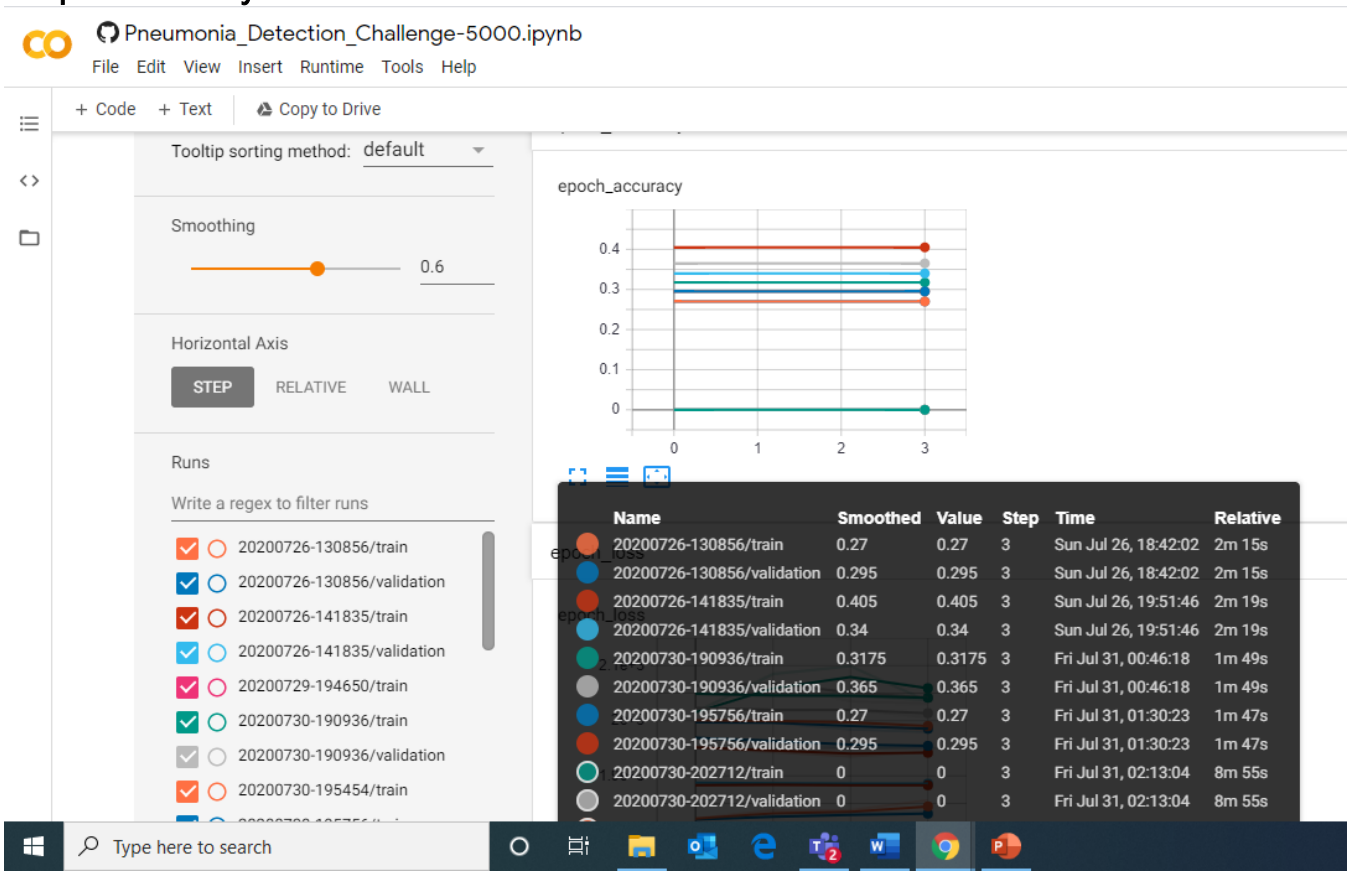
Model was executed for 100 epochs, however we saw that model exited since there is not much improvement in accuracy.

```
[ ] # Fit the model to the data
    model = train_model()
```

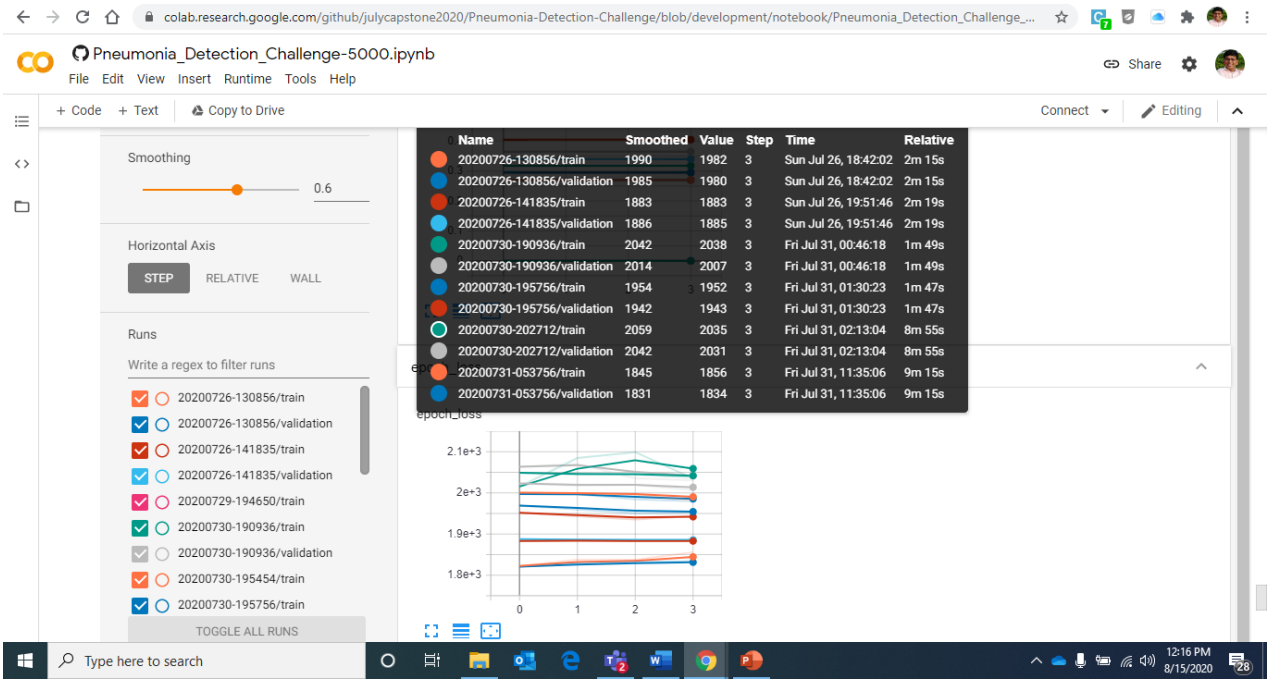
```
Building model with:  https://tfhub.dev/google/imagenet/mobilenet_v2_140_224/feature_vector/4
Epoch 1/100
125/125 [==============================] - 1057s 8s/step - loss: 1822.2823 - accuracy: 0.0000e+00 - val_loss: 1820.7301 - val_accuracy: 0.0000e+00
Epoch 2/100
125/125 [==============================] - 184s 1s/step - loss: 1837.7909 - accuracy: 0.0000e+00 - val_loss: 1829.0399 - val_accuracy: 0.0000e+00
Epoch 3/100
125/125 [==============================] - 187s 1s/step - loss: 1837.8735 - accuracy: 0.0000e+00 - val_loss: 1832.3391 - val_accuracy: 0.0000e+00
Epoch 4/100
125/125 [==============================] - 181s 1s/step - loss: 1855.8871 - accuracy: 0.0000e+00 - val_loss: 1834.0760 - val_accuracy: 0.0000e+00
```

We leveraged Tensorboard to depict the accuracy results and below are representations of them.

Epoch Accuracy:



Epoch Loss:



Based on the results, MobileNet architecture may not be suitable model to rightly understand for the bounding box scenario's. Hence we moved on implementing the relevant model to validate the outcomes for the specific tasks.

6.2 Yolo

The YOLO framework (You Only Look Once) deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity.

YOLO is considered best for real time detection.

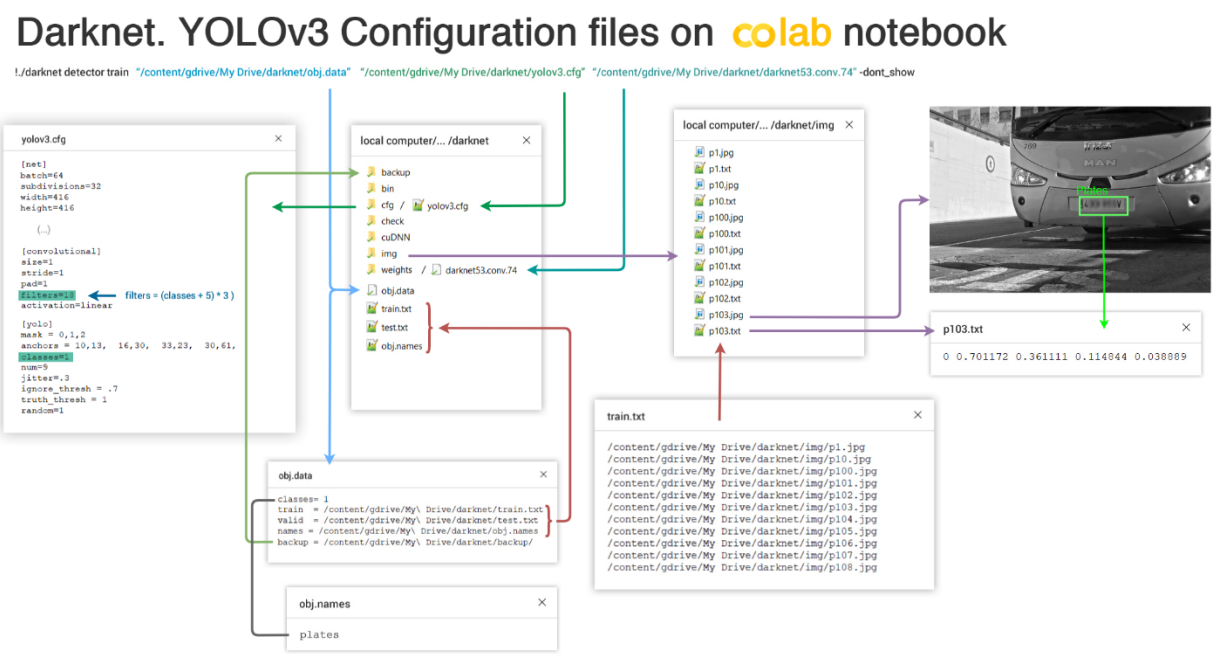
6.2.1 YOLO Application Reference

Code implementation path:
https://github.com/julycapstone2020/Pneumonia-Detection-Challenge/blob/master/notebook/yolo%20(2).ipynb

For YOLO implementation we explored Transfer learning option and finalized to use Darknet implementation and leverage the benefits of this deployment to drive better accuracy results. As part of that, we decided to close the darknet package and modify the configurations according to our model requirement which needs to be implemented. With some of the basic due diligence tests performed by testing Yolov3 and Yolov4, we finalized to adopt to use Yolov4 algorithm for our Pneumonia deployment perspective.

For Darknet model configuration – we referred to the deployment guide in the link
https://colab.research.google.com/drive/1ITGZsfMaGUpBG4inDIQwIJVW476ibXk_#scrollTo=wkzMqLZV-rF5

This below reference diagram was referred to build the clone of the project and develop the project specific directory structure along the specific parameters that we adopted to make our model suit the requirement.



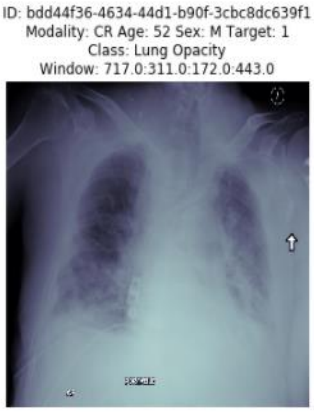
As part of that for transfer learning we leveraged **yolov4-tiny.conv.29** weights while executing the model on all the images.
For processing them further in the model. Per Yolo's recommendation, we also deci ded to use metrics of Average IOU (Intersection Over Union) Metrics to measure the success of the model.

Below diagram shows the accuracy measures of Average IOU scores for the model developed.

6.2.2 Sample Image representations

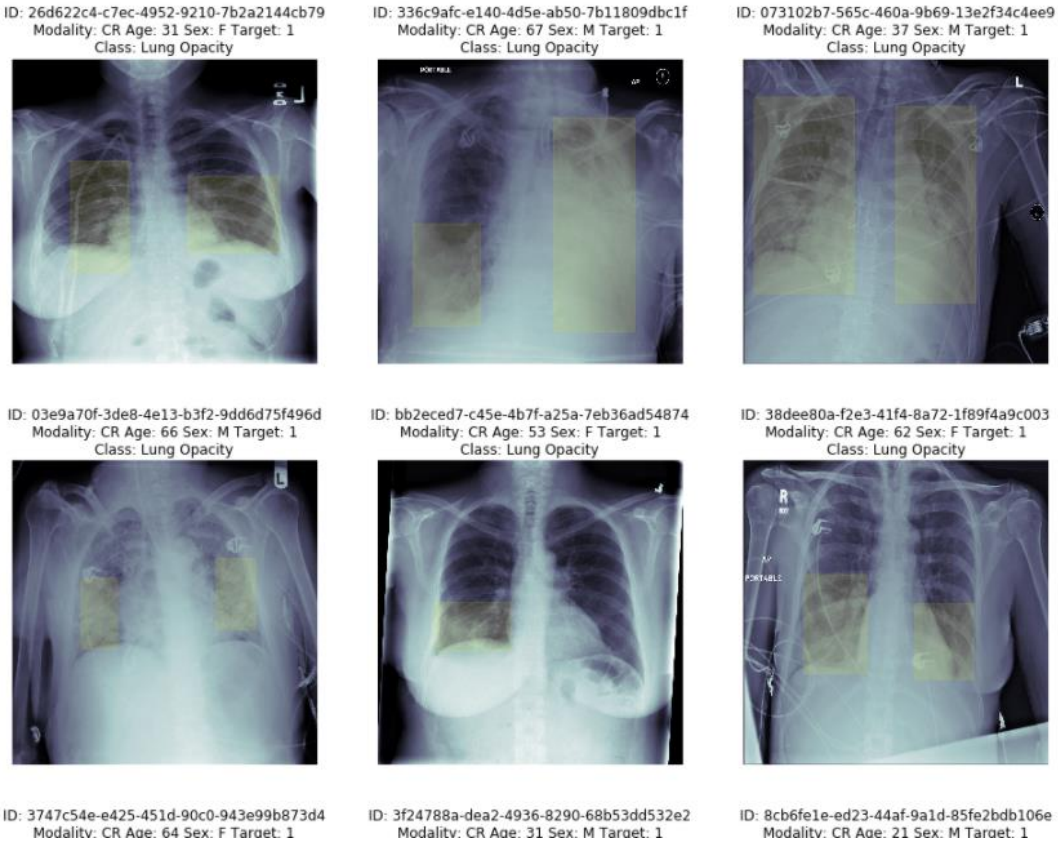
Based on the preformed tests below are the ground truth representations that we have in the dataset against which we will be doing the predictions

```
[17]: 1 show_ground_truth()
```



The below images represents the actual predictions for the tests performed along with its bounding boxes. Only few samples are represented as below.

```
In [18]: 1 show_predictions()
```



6.2.3 Model Architecture

ResNet has won several competitions and its architecture allows for better learning in deeper networks. I've used the Keras implementation with weights of ResNet50 and modified the code to have the YOLO classifier at the end.

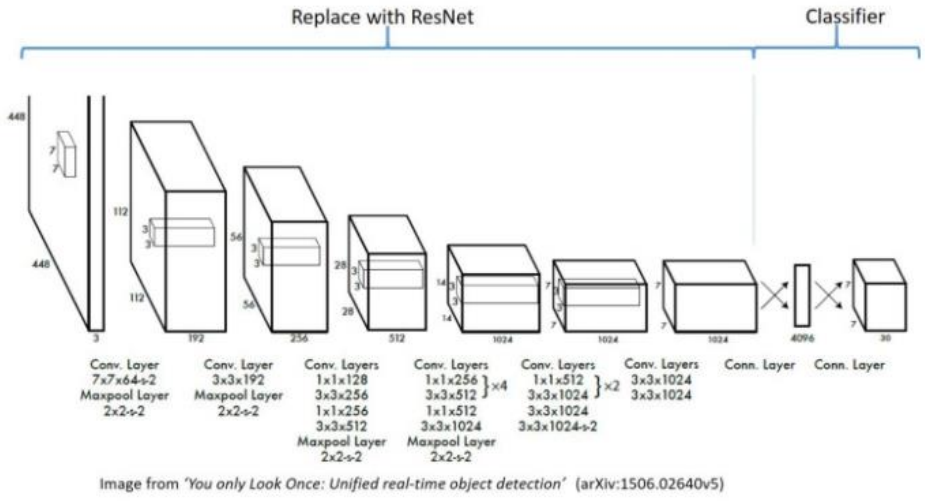


Image from *'You only Look Once: Unified real-time object detection'* (arXiv:1506.02640v5)

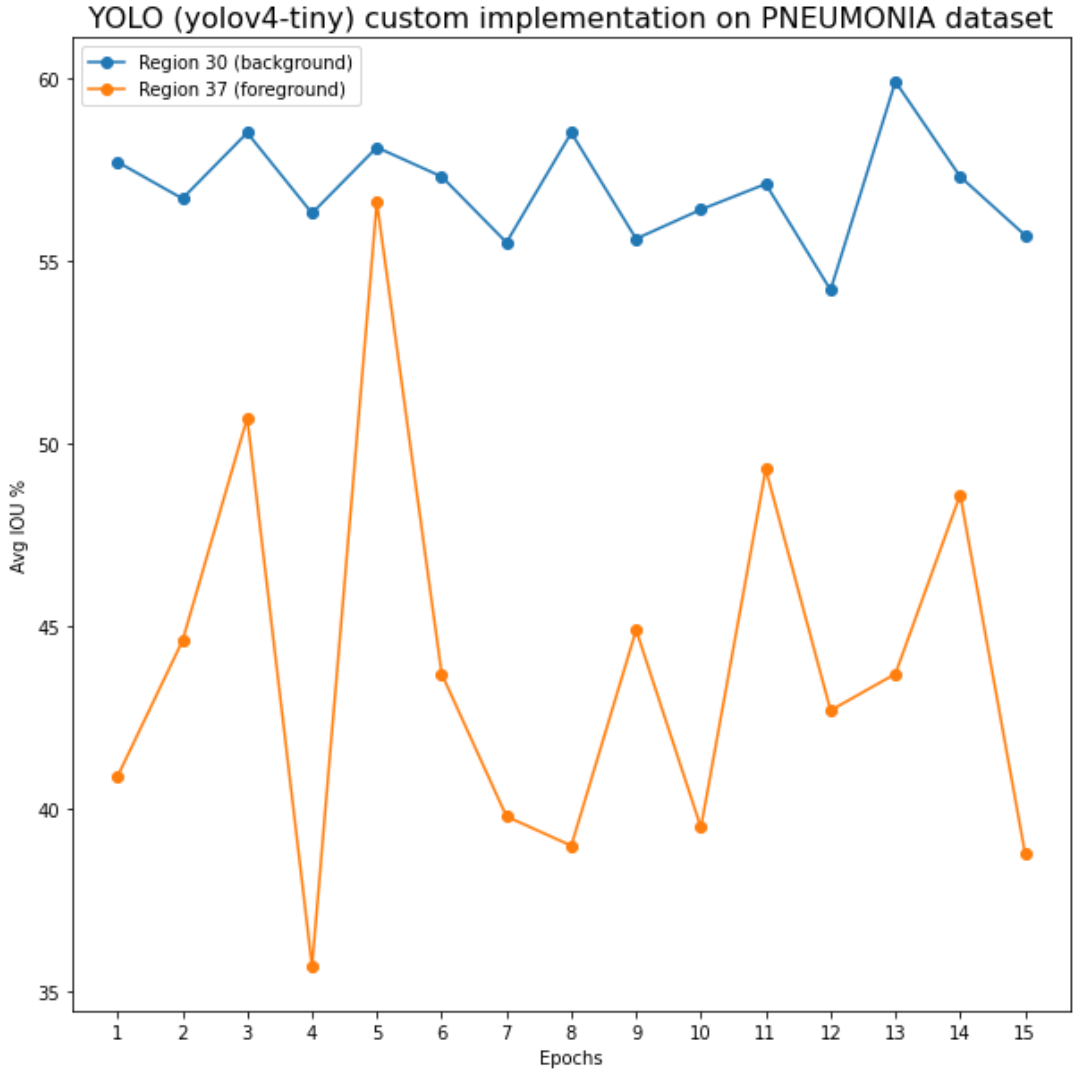
Original architecture and modifications

The YOLO detector is broken into three main pieces.

- **YOLO Backbone** — The YOLO backbone is a convolutional neural network that pools image pixels to form features at different granularities. The Backbone is typically pretrained on a classification dataset, typically ImageNet.
- **YOLO Neck** — The YOLO neck (FPN is chosen above) combines and mixes the ConvNet layer representations before passing on to the prediction head.
- **YOLO Head** — This is the part of the network that makes the bounding box and class prediction. It is guided by the three YOLO loss functions for class, box, and objectness.

6.2.4 Result and conclusions

Based on the test results plotted we see the below prediction trend for with foreground and background outcome.



With respect to the above outcome of high IOU coverage in prediction we consider YOLO to be one of the best model to adopt for such predictions.

6.3 Mask RCNN

Mask R-CNN is been a deep neural network which is aimed to solve instance segmentation problem in Computer Vision and Machine Learning. This can separate different objects in an image or video by providing object bounding boxes, classes and masks associated with the image or video. Mask R-CNN first generates proposals about the regions where there might be an object and it predicts the class of the object. In this process it refines the bounding boxes and generates a mask at pixel level of the object based on the proposal created.

6.3.1 Mask-RCNN Application references

Code implementation path:
https://github.com/julycapstone2020/Pneumonia-Detection-Challenge/blob/master/notebook/maskRCNN.ipynb

For this specific implementation – we referenced the "matterport" package implementation of mask r-cnn and cloned it to use it in our project. Details of this is available in this link
The reference that cloned includes the following;

- Source code was built on Mask R-CNN with FPN and ResNet101
- Pretrained weights for MS COCO dataset
- ParallelModel class for multi-GPU training.

Since this model also requires annotations on the images, we also explored options by referring to the details on Annotator project provided by MD.ai and this can be referenced in this link

Below image refers to the code block where we are leveraging the MD.ai annotations..

```
In [72]: p = mdai_client.project('LxR6zdR2', path='./lesson3-data')

Using path './lesson3-data' for data.
Preparing annotations export for project LxR6zdR2...
Downloading file: mdai_public_project_LxR6zdR2_annotations_labelgroup_all_2018-09-05-185713.json
Preparing images export for project LxR6zdR2...
Downloading file: mdai_public_project_LxR6zdR2_images_2018-08-20-184248.zip
18.7MB [00:00, 78.2MB/s]
  0%|          | 0.00/3.18G [00:00<?, ?B/s]
Success: annotations data for project LxR6zdR2 ready.
3.18GB [00:53, 64.2MB/s]
Extracting archive: mdai_public_project_LxR6zdR2_images_2018-08-20-184248.zip
Success: images data for project LxR6zdR2 ready.
```

Below section represents the key configuration items for the model that we developed by leveraging Resnet50 model.

```
Configurations:
BACKBONE                       resnet50
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     16
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        3
DETECTION_MIN_CONFIDENCE       0.9
DETECTION_NMS_THRESHOLD        0.1
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 16
IMAGE_CHANNEL_COUNT            3
IMAGE_MAX_DIM                  64
IMAGE_META_SIZE                14
IMAGE_MIN_DIM                  64
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [64 64  3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'm
rcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               3
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           pneumonia
NUM_CLASSES                    2
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         200
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    16
STEPS_PER_EPOCH                100
TOP_DOWN_PYRAMID_SIZE          32
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           16
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               50
WEIGHT_DECAY                   0.0001
```

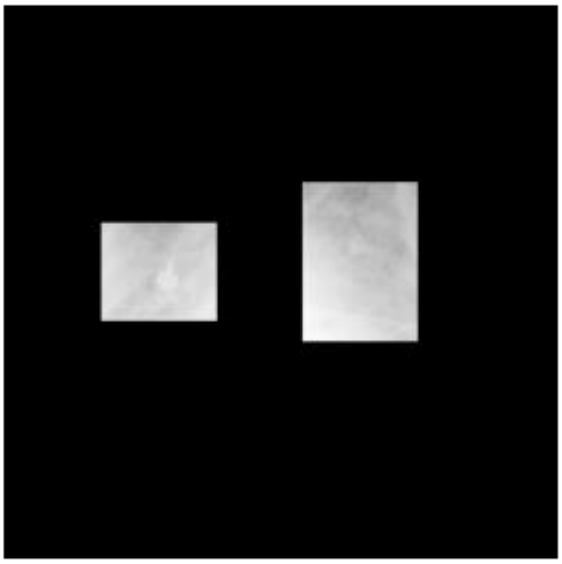
As part of transfer learning we leveraged the weigh file provided by the model developm ent and training.

mask_rcnn_pneumonia_0006.h5

6.3.2 Sample image representations

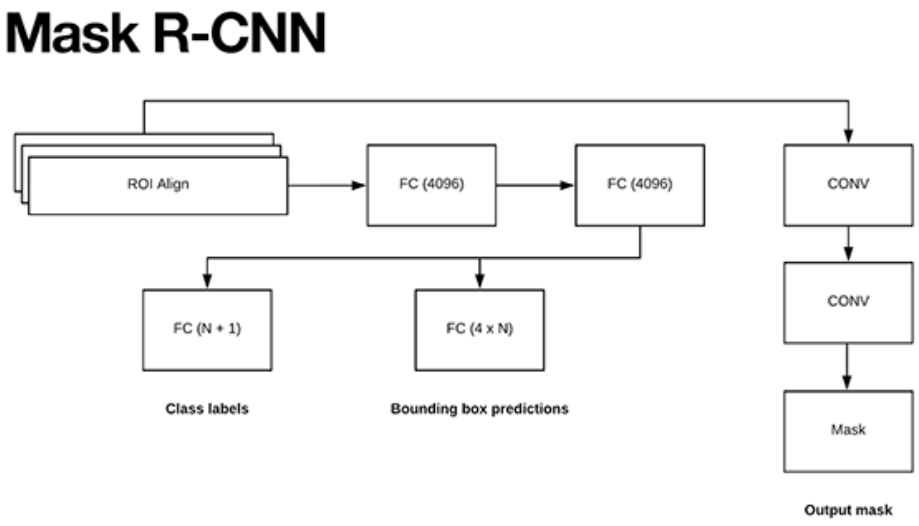
While developing the model, initially we started with training for 1000 images samples and then extended further.

Below represents one such reference having the bounding box and masks applied to it.



6.3.3 Model Architecture

Mask R-CNN used Resnet 101 architecture to extract features from the images. However for this model development we leveraged the Resnet50 Architecture.



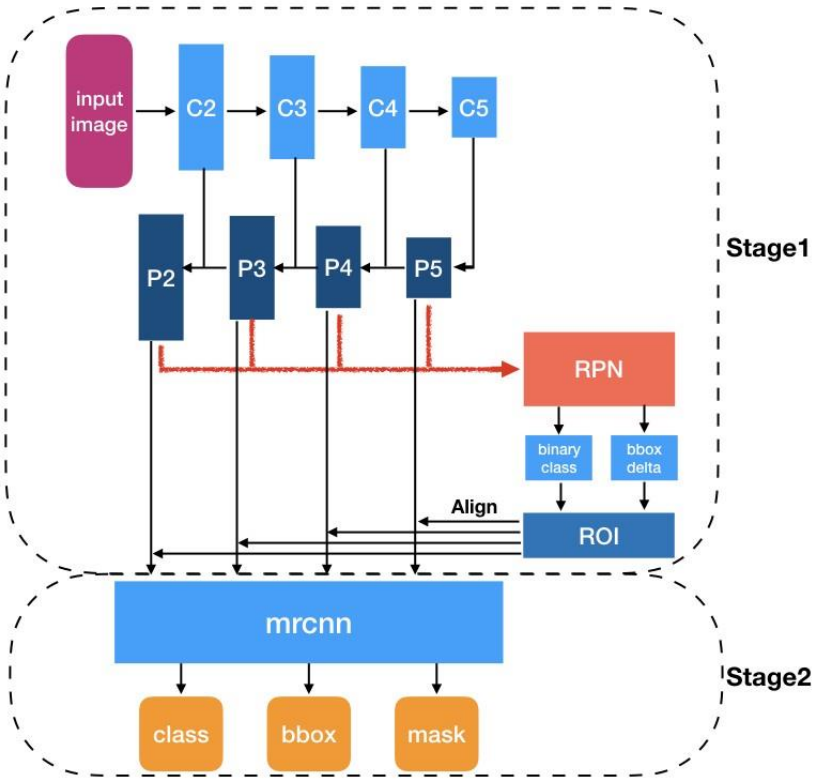
Base architecture of Resnet50 is detailed below.

ResNet50 Architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

6.3.4 Results and Conclusion

Mask R-CNN's 2 staged approach is as below.



Basis the above we documented the output of the loss functions for the class, bounding box and the mask for both training and validation process that we have detailed below.

Here are the details of steps that we followed to train the model and results in each phases.

With Epoch set to 1, the results are as below.

```
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/keras/callbacks.py:708: The name tf.summary.FileWriter is deprecated. Plea
se use tf.compat.v1.summary.FileWriter instead.

Epoch 1/1
 99/100 [============================>.] - ETA: 45s - loss: 3.1424 - rpn_class_loss: 0.4958 - rpn_bbox_loss: 1.2366 - mrcnn_class_loss:
0.2318 - mrcnn_bbox_loss: 0.6381 - mrcnn_mask_loss: 0.5400WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/keras/callbacks.
py:791: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.

100/100 [==============================] - 5946s - loss: 3.1372 - rpn_class_loss: 0.4941 - rpn_bbox_loss: 1.2334 - mrcnn_class_loss: 0.23
24 - mrcnn_bbox_loss: 0.6374 - mrcnn_mask_loss: 0.5399 - val_loss: 2.6425 - val_rpn_class_loss: 0.3213 - val_rpn_bbox_loss: 0.9511 - val_
mrcnn_class_loss: 0.2849 - val_mrcnn_bbox_loss: 0.5899 - val_mrcnn_mask_loss: 0.4953
```

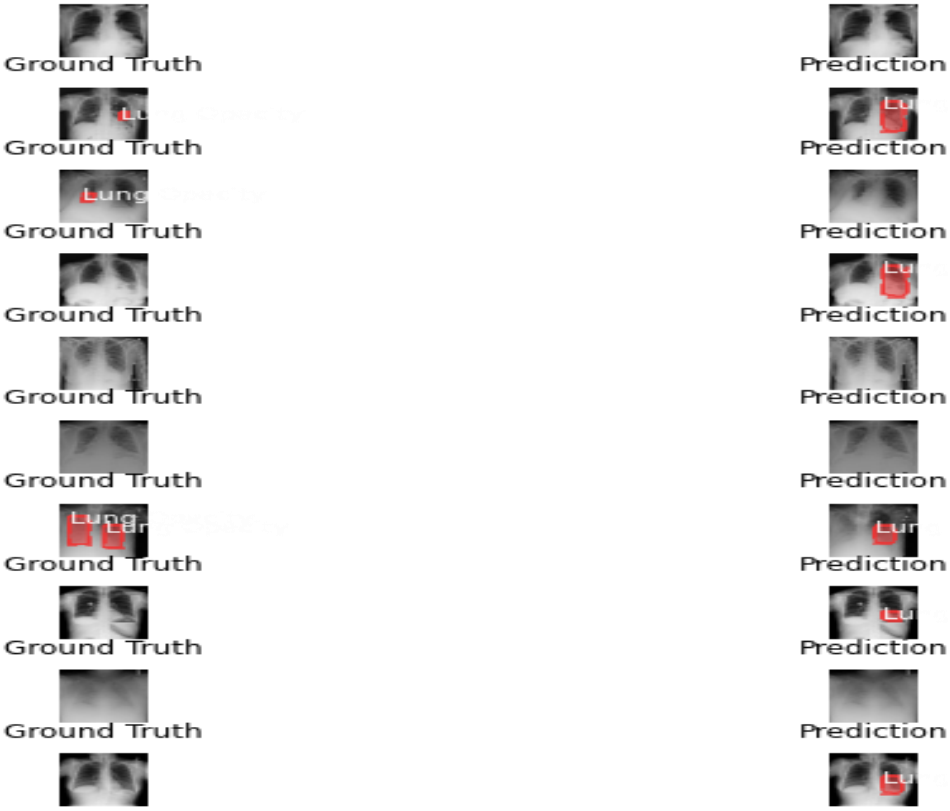
With Epoch set to 5:

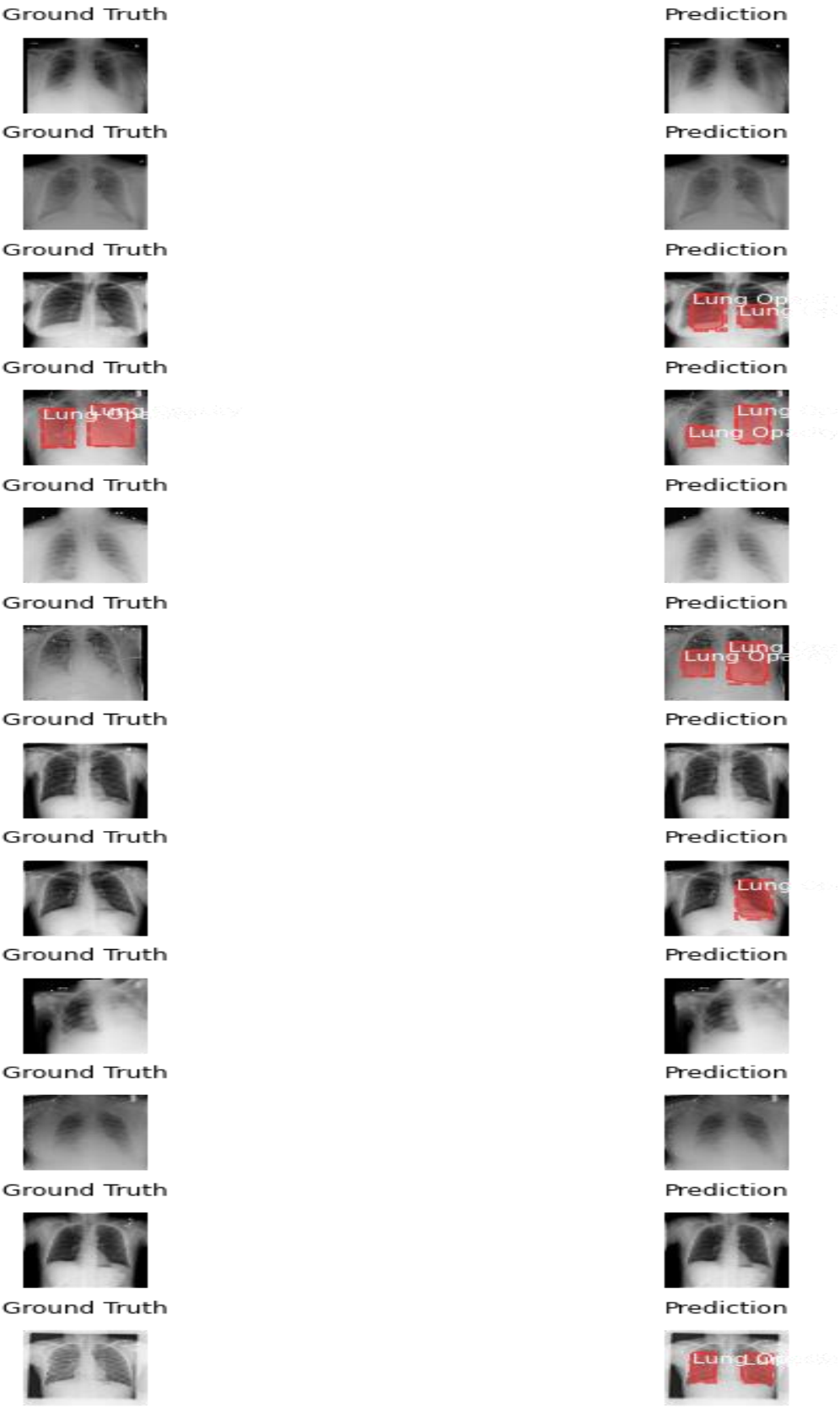
```
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/keras/callbacks.py:708: The name tf.summary.FileWriter is deprecated. Plea
se use tf.compat.v1.summary.FileWriter instead.

Epoch 1/5
 99/100 [============================>.] - ETA: 45s - loss: 3.4576 - rpn_class_loss: 0.5247 - rpn_bbox_loss: 1.4579 - mrcnn_class_loss:
0.2005 - mrcnn_bbox_loss: 0.6974 - mrcnn_mask_loss: 0.5771WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/keras/callbacks.
py:791: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.

100/100 [==============================] - 5974s - loss: 3.4479 - rpn_class_loss: 0.5227 - rpn_bbox_loss: 1.4513 - mrcnn_class_loss: 0.20
17 - mrcnn_bbox_loss: 0.6957 - mrcnn_mask_loss: 0.5765 - val_loss: 2.6595 - val_rpn_class_loss: 0.3320 - val_rpn_bbox_loss: 0.9204 - val_
mrcnn_class_loss: 0.2883 - val_mrcnn_bbox_loss: 0.5765 - val_mrcnn_mask_loss: 0.5423
Epoch 2/5
100/100 [==============================] - 5162s - loss: 2.6242 - rpn_class_loss: 0.3221 - rpn_bbox_loss: 0.9565 - mrcnn_class_loss: 0.24
82 - mrcnn_bbox_loss: 0.5770 - mrcnn_mask_loss: 0.5204 - val_loss: 2.5185 - val_rpn_class_loss: 0.2981 - val_rpn_bbox_loss: 0.8741 - val_
mrcnn_class_loss: 0.2596 - val_mrcnn_bbox_loss: 0.5761 - val_mrcnn_mask_loss: 0.5105
Epoch 3/5
100/100 [==============================] - 5226s - loss: 2.5103 - rpn_class_loss: 0.3021 - rpn_bbox_loss: 0.8728 - mrcnn_class_loss: 0.27
19 - mrcnn_bbox_loss: 0.5592 - mrcnn_mask_loss: 0.5043 - val_loss: 2.5898 - val_rpn_class_loss: 0.2999 - val_rpn_bbox_loss: 0.9829 - val_
mrcnn_class_loss: 0.2450 - val_mrcnn_bbox_loss: 0.5162 - val_mrcnn_mask_loss: 0.5458
Epoch 4/5
 46/100 [============>.................] - ETA: 2068s - loss: 2.4340 - rpn_class_loss: 0.2885 - rpn_bbox_loss: 0.8728 - mrcnn_class_loss:
0.2502 - mrcnn_bbox_loss: 0.5327 - mrcnn_mask_loss: 0.4898
```

As we progressed further, below are actual results that we have noted.


Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction


Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction
Ground Truth
Prediction

Following are the different loss functions that we captured to run our analysis and findings.

Training set loss items:

- Loss: Overall loss which is a combination of the smaller losses
- rpn_class_loss : How well the RPN separates background with objects
- rpn_bbox_loss : How well the RPN localize objects
- mrcnn_bbox_loss : How well the Mask RCNN localize objects
- mrcnn_class_loss : How well the Mask RCNN recognize each class of object
- mrcnn_mask_loss : How well the Mask RCNN segment objects

Validation set loss items:

- val_loss: Overall loss which is a combination of the smaller losses
- val_rpn_class_loss : How well the RPN separates background with objects
- val_rpn_bbox_loss : How well the RPN localize objects
- val_mrcnn_bbox_loss : How well the Mask RCNN localize objects
- val_mrcnn_class_loss : How well the Mask RCNN recognize each class of object
- val_mrcnn_mask_loss : How well the Mask RCNN segment objects

We notice that the total loss is consistently reducing from Epoch to Epoch and accordingly the accuracy rate is improving too.

Loss value starting with 3.4576 has reduced to 2.4340 in 5 epochs
Validation Loss in 2nd epoch 2.6595 has reduced to 2.5898 in 4th epoch..

We also note that there is loss reduction and therefore the accuracy is consistent with the training dataset results and hence we don't expect to have the overfit situation based on the outcome seen so far.

Hence we conclude that mask R-CNN is a good model for us to consider and if further extended the performance of this model can provide better results.

7.Conclusion

Based on the outcome of the results between MobileNet, YOLO and mask R-CNN models and comparing the results, we conclude that MobileNet is not an option to consider for the Pneumonic detection cases. Hence, for comparison we analysed results of YOLO and mask R-CNN.

In YOLO result we see that the IOU is nearly 50% and shows improved trend as we train more and more epochs. We observe this quite faster as well like, it performs feature extraction, bounding box prediction, non-max suppression and contextual reasoning all concurrently.

However we do notice that YOLO may struggle with small objects that appear in groups and hence it cannot generalize to objects with unusual aspect ratios or configurations. Also, Yolo imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can have only one class. Hence, this limits the number of nearby objects that the model can predict.

In mask RCNN outcome analysis we notice that in just 5 epochs how the loss is reducing for each epoch and the results at the beginning of the train and test set is very encouraging to the tune of 55% and above accuracy level. This is we believe is because of the backbone model it adopts and the 2 stage approach in deriving at the bounding box and masks.

To conclude our approach, we have dived into making a current approach with one of the most sought out approach for medical diagnosis. mask R-CNN has shown promising results with epochs up-to 5, further training could ideally scale the results to much better values(as per the log and the loss encountered during the training).

8. Next Steps

DenseNet architecture has shown prediction accuracies with results as good as 95% predictions. It serves as an industry benchmark for an evaluation approach. Our current algorithm implementation was able to achieve about 55% and with further epochs and hyperparameter tuning we are confident it can scale to better results.

Likewise, we also recommend to build a model using SSD (Single Shot Detector) algorithm for the Pneumonic detection. By using SSD, we only need to take one single shot to detect multiple objects within an image. SSD the way it works can be considered an improvement over RCNN models due to this since its much faster and efficient compared to 2 shot RPN based approaches.

In addition, we would also like to build an API layer using FLASK model such that the model that we built can be put to use in the real deployment cycle.