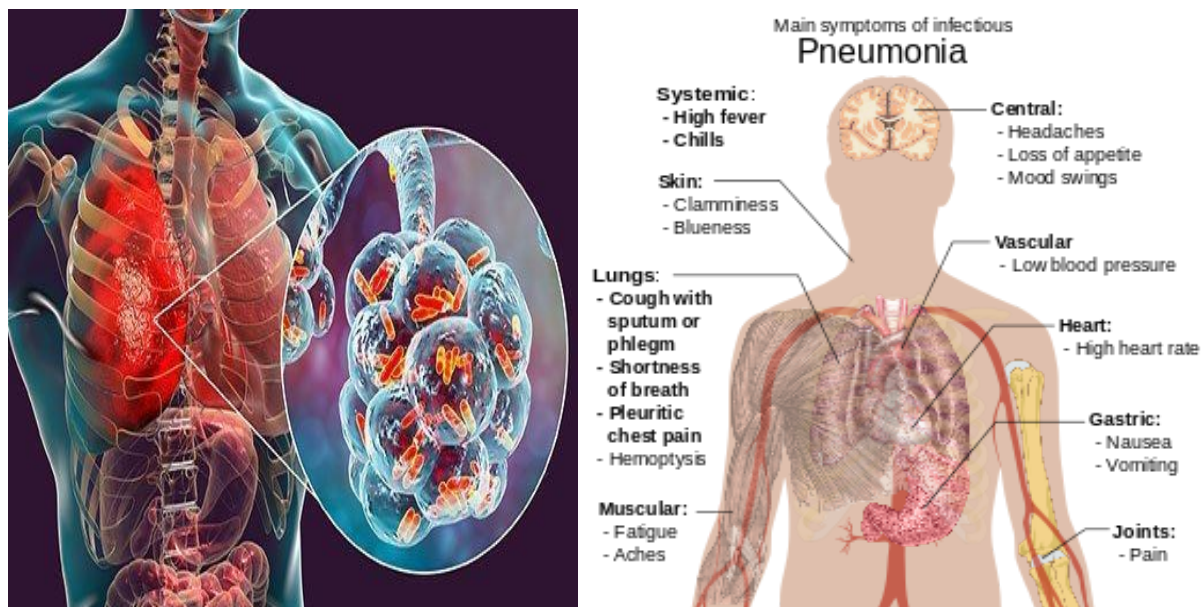

FINAL REPORT - PNEUMONIA DETECTION

Overview

Pneumonia is an inflammatory condition of the lung primarily affecting the small air sacs known as alveoli. Symptoms typically include some combination of productive or dry cough, chest pain, fever, and difficulty breathing. The severity of the condition is variable.^[1]



Pneumonia is usually caused by infection with viruses or bacteria, and less commonly by other microorganisms. Identifying the responsible pathogen can be difficult. Diagnosis is often based on symptoms and physical examination. Chest X-rays, blood tests, and culture of the sputum may help confirm the diagnosis. The disease may be classified by where it was acquired, such as community- or hospital-acquired or healthcare-associated pneumonia.

Risk factors for pneumonia include cystic fibrosis, chronic obstructive pulmonary disease (COPD), sickle cell disease, asthma, diabetes, heart failure, a history of smoking, a poor ability to cough (such as following a stroke), and a weak immune system.

Treatment depends on the underlying cause. Pneumonia believed to be due to bacteria is treated with antibiotics. If the pneumonia is severe, the affected person is generally hospitalized. Oxygen therapy may be used if oxygen levels are low.

Pneumonia often shortens the period of suffering among those already close to death and has thus been called "the old man's friend".

Literature Survey

In recent years, several methodologies have been investigated for successful detection of pneumonia using chest images. Deep learning-based models have been successfully applied to improve the performance of computer aided diagnosis technology (CAD), especially in the field of medical imaging ^[1], image segmentation ^[2,3] and image reconstruction ^[4].

With increase in computing power, popular neural network-based models such as ALEX Net^[5], VGG net^[6] have been proposed. However with increase in network complexity there is the inherent problem of vanishing gradient or exploding gradient.

Dejun .etal ^[7] proposed a residual connection structure, to solve the problem of network depth. In this the Histogram equalization was applied to improve the contrast. The model had six layers, with 3×3 kernel convolution layers whose strides are 1×1 and the activation function is ReLU. After each convolution layer, a 2×2 strides kernel operation was employed as a max pooling. The results of the obtained accuracy rate of 96.07% and precision rate of 94.41%

A pre-trained dense Net based model was developed by Almaslukh et al. ^[8] with 121 cnn layers to detect pneumonia. Dense Net has the advantage of providing best representation of lung features. Weighted cross entropy loss was used to account for class imbalance.

Shangjie et al ^[9] used clahe for improving contrast, the deep-learning pneumonia-detection method used in this paper is based on the Faster R-CNN, in which the backbone uses the low-complexity dilated bottleneck residual neural network called DeepConv-DilatedNet. In the DetNet network, the first four phases of the backbone are consistent with the original ResNet50 phase, maintaining a 16x receptive field from the fifth stage and adding a stage, using 1×1 convolutions and 3×3 dilated convolutions in the fifth and sixth stages.

To enhance the pictorial information output of the DetNet network, this paper combines FPN and DetNet to enhance the feature-extraction mode of the network. And put forward the DeepConv-DilatedNet.

Hashmi et al ^[10] applied image augmentation to address the problem of limited dataset and used a weighted classifier approach to compute the final predictions made by combination models like ResNet18,DenseNet 121,Inception V3,Xception,MobileNetv2.Fine tuning SGD,learning rate(0.001)and 25 epochs.

Summary of Problem Statement, Data & Findings:

Objective

Due to the high volume of chest radiography, it is very time consuming and intensive for the radiologists to review each image manually. As such, an automated solution is ideal to locate the position of inflammation in an image. By having such an automated pneumonia screening system, this can assist physicians to make better clinical decisions or even replace human judgement in this area.

Therefore, our objective is to build an algorithm to detect a visual signal for pneumonia in medical images. Specifically, our algorithm needs to automatically locate lung opacities on chest radiographs (Chest X-rays).

Approach

We've used the publicly available dataset which was created by the Radiological Society of North America (RSNA).

This dataset contains 29,684 chest radiographs and can be accessed through the below link:
<https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data>

The data is stored in a zip file labelled "rsna-pneumonia-detection-challenge.zip", which contains the following items:

1. stage_2_train_images: This folder contains all the training DICOM images.
2. stage_2_train_labels.csv: This file contains the corresponding patientID images to the folder 'stage_2_train_images' and contains the bounding box of areas of pneumonia detected in each image along with a target label of 0 or 1 for pneumonia detected.
3. stage_2_detailed_class_info.csv: This file contains the corresponding patientID images to the folder 'stage_2_train_images' and contains the target classes of the images.
4. stage_2_test_images: This folder contains all the test DICOM images.

5. `stage_2_sample_submission.csv`: This file contains the corresponding patientID images to the folder 'stage_2_test_images'.

All the images are stored as .dcm images. So, to use Object detection models, we would need to convert these images from .dcm to either .jpg or .png formats.

Firstly, we understood the data with a comprehensive EDA on the available set. Based on the insights of the same and the literature available, we went ahead with selecting the model.

After the EDA, the plan is to go through various models which can be used for the problem statement, implement and fine-tune them for maximum accuracy (IOU score). We will also try to use *Ensemble* of all the successful models to get the best results and minimize inaccuracies.

Overview of the final process:

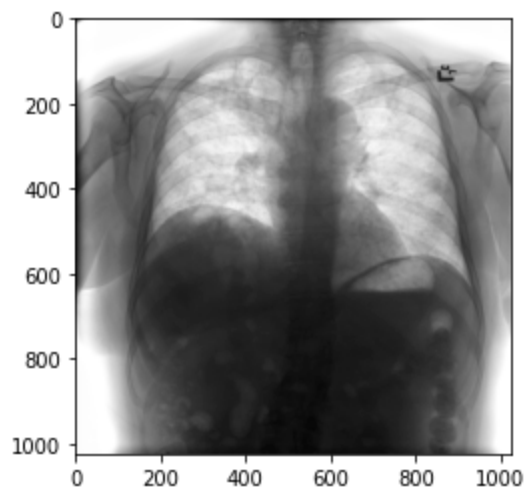
Exploratory Data Analysis

- The total number of images in the Train & Test folders are around 30k. Here's the split:

```
Number of images in the train set: 26684
Number of images in the test set: 3000
```

- Visualizing a Chest X-ray:

All the images are in the dicom format. All the images are of dimension 1000 * 1000. Here's an example:



- Metadata example:

Dicom images also contains each patient's metadata, visualizing an example:

```
Dataset.file_meta -----
(0002, 0000) File Meta Information Group Length  UL: 200
(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: 1.2.276.0.7230010.3.1.4.8323329.2151.1517874294.898960
(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: 1.2.276.0.7230010.3.1.4.8323329.2151.1517874294.898960
(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: 'a5122049-489e-4c01-b8f3-2de734058d2a'
(0010, 0020) Patient ID                      LO: 'a5122049-489e-4c01-b8f3-2de734058d2a'
(0010, 0030) Patient's Birth Date             DA: ''
(0010, 0040) Patient's Sex                   CS: 'M'
(0010, 1010) Patient's Age                    AS: '62'
(0018, 0015) Body Part Examined               CS: 'CHEST'
(0018, 5101) View Position                   CS: 'PA'
(0020, 000d) Study Instance UID               UI: 1.2.276.0.7230010.3.1.2.8323329.2151.1517874294.898959
(0020, 000e) Series Instance UID             UI: 1.2.276.0.7230010.3.1.3.8323329.2151.1517874294.898958
(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 158000 elements
```

There is important information in the metadata like the Age & Gender, we will look at getting the metadata into our combined dataframe.

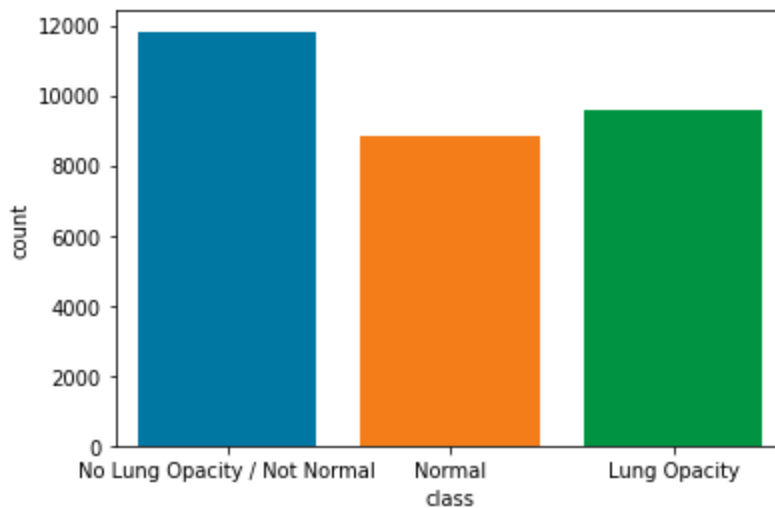
- Class info:

A quick glance of the 'stage_2_detailed_class_info.csv' file reveals that it has only 2 columns:

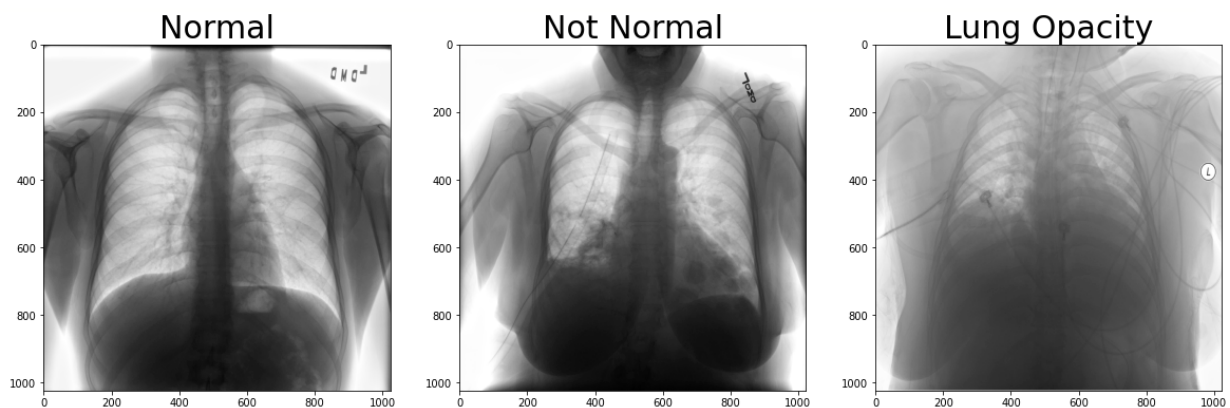
1. PatientId – which refers to the patientId's corresponding image name.
2. Class – target label of the patientId's image which can take 1 of 3 values:
 - a. **Normal**
 - b. **Lung Opacity**
 - c. **No Lung Opacity / Not Normal.**

- Class distribution:

Classes are more or less evenly distributed (visualization below)



- Visualizing images from each class:



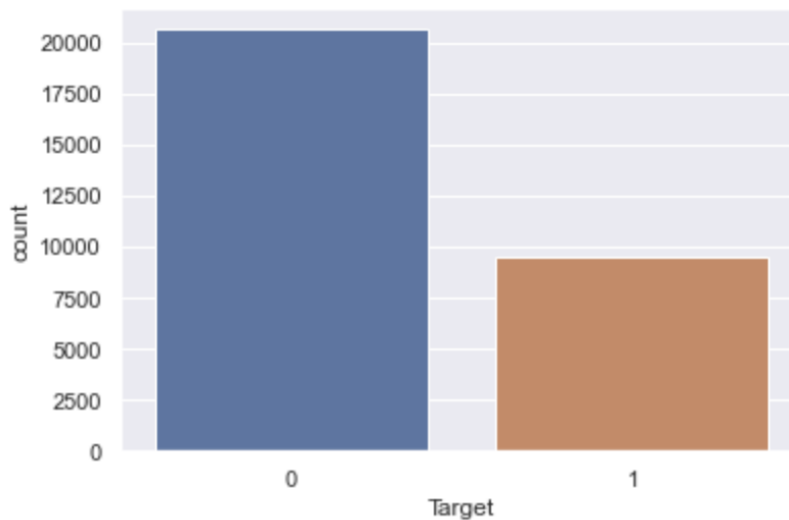
- Train Label info:

A quick glance of the 'stage_2_train_labels.csv' file reveals that it has 6 columns:

1. **PatientId** – which refers to the patientId's corresponding image name.
2. **X** – upper-left x coordinate of the bounding box.
3. **Y** – upper-left y coordinate of the bounding box.
4. **Width** – the width of the bounding box.
5. **Height** – the height of the bounding box.
6. **Target** – binary target indicating if this image has evidence of pneumonia.

- Target distribution:

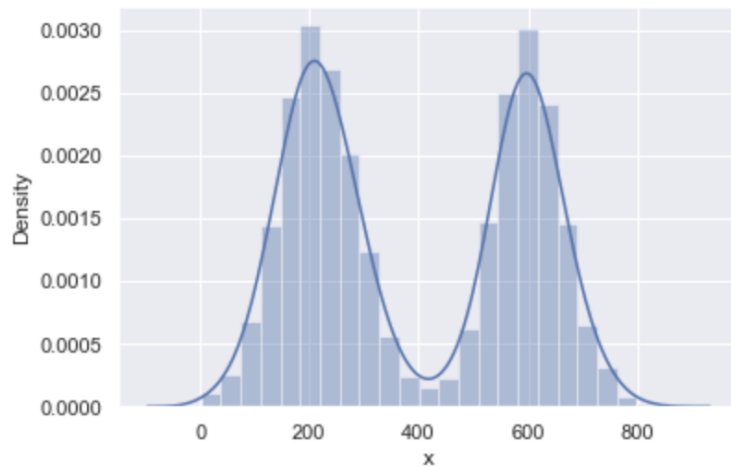
Are the target columns evenly distributed?



They look to be somewhat evenly distributed with ~2:1 ratio of class 0 to class 1.

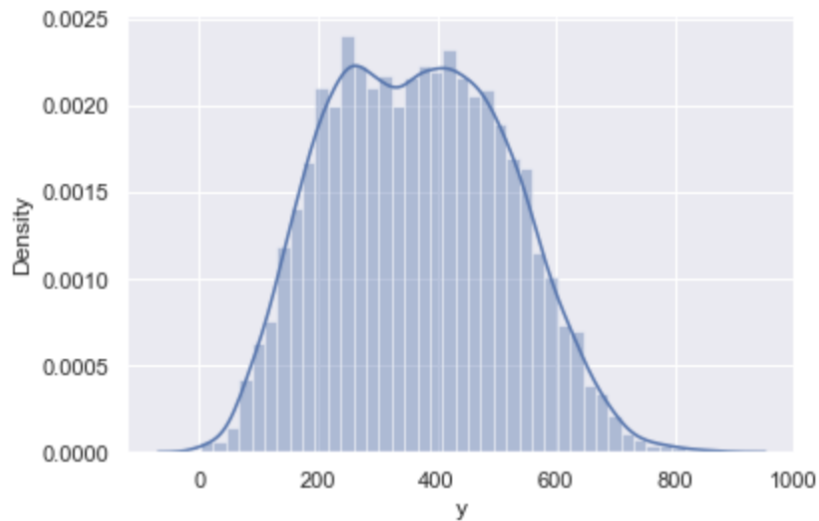
- Distribution of x, y, width and height:

Let's look at the distribution of x (in cases where pneumonia is detected), to get a rough sense of where the pneumonia lies:



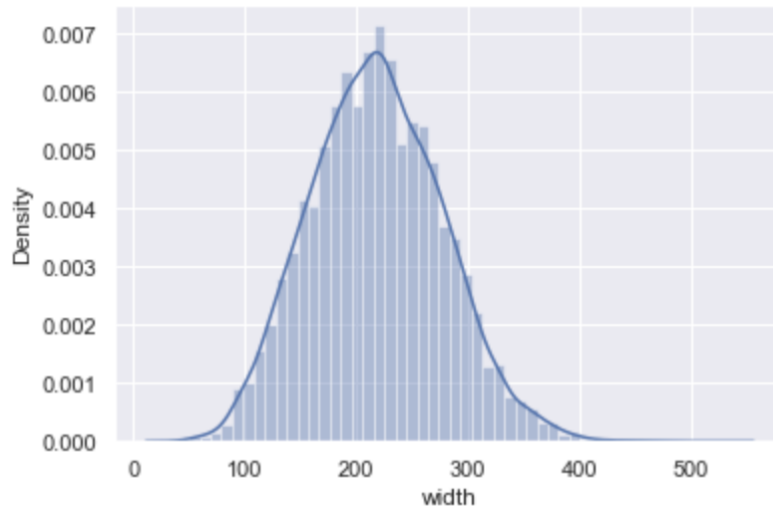
From the above we can safely say that there is equal likelihood of pneumonia to be present in either lungs (and the data doesn't have a bias with respect to a particular lung)

Let's now look at the distribution of y (in cases where pneumonia is detected) also:



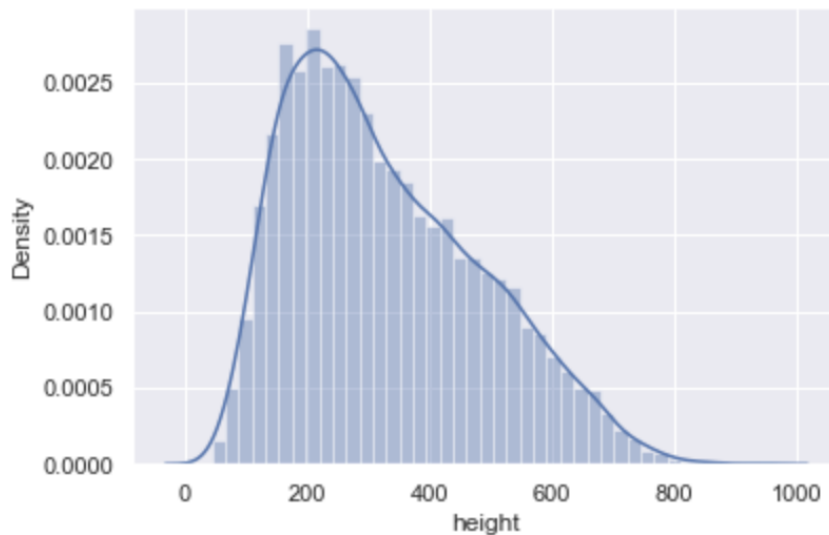
We can see from the above that there are roughly two peaks, but still the distribution is mostly normal. For most of the cases, the top of the box is roughly starting around 20-50% of the image (200-500 px out of 1000 px). Again, this means that the data hardly has any outlier in terms of lung position and all. It also suggests that all the images follow a particular orientation.

Distribution of width (of the bounding box):



Width of the boxes are following a near normal distribution with a peak around 200. This is in sync with the width of an infected lung.

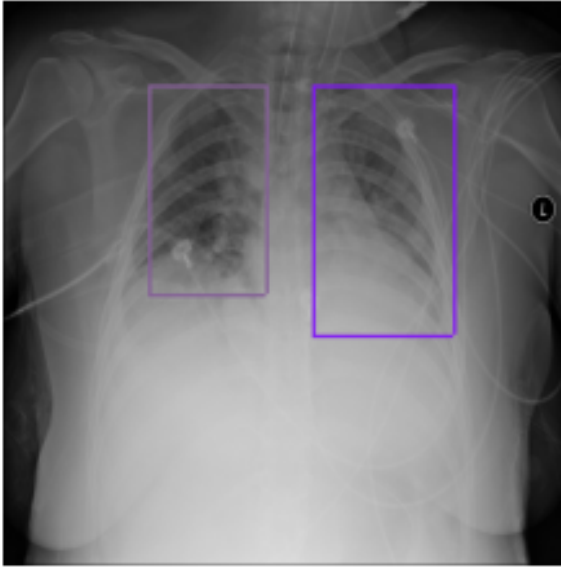
Distribution of height (of the bounding box):



From the above, it can be inferred that the distribution has a long tail on the right side. No significant outliers.

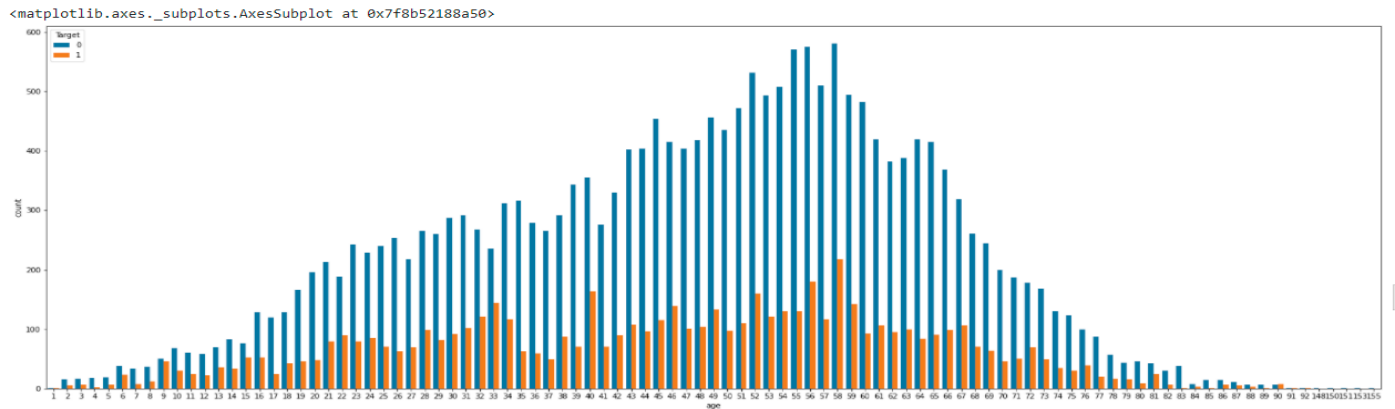
- Plotting bounding boxes:

Visualizing X-rays that have pneumonia with their bounding boxes.

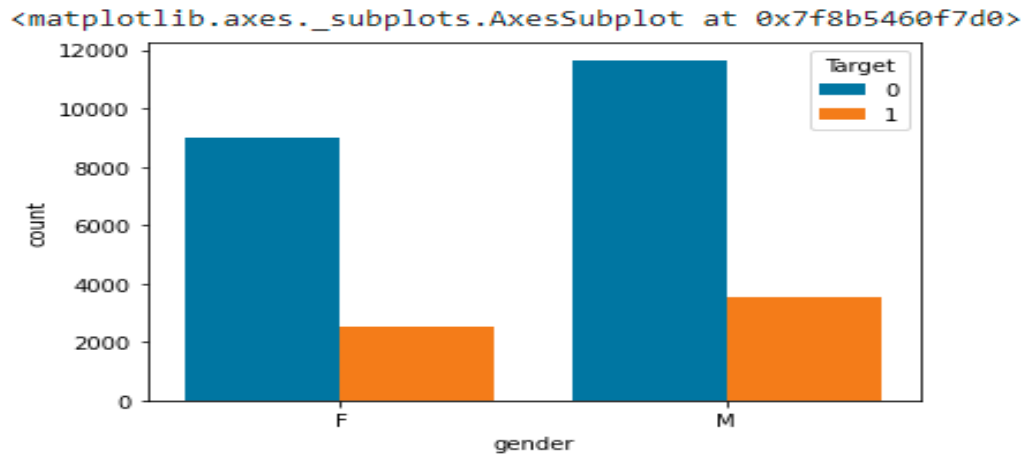


- Distribution of pneumonia among the age groups:

We noticed that the metadata has important information like the Age, Gender so we got this into our dataframe. We will now look at comparing the Age against our Target column:



- Gender distribution between Target:



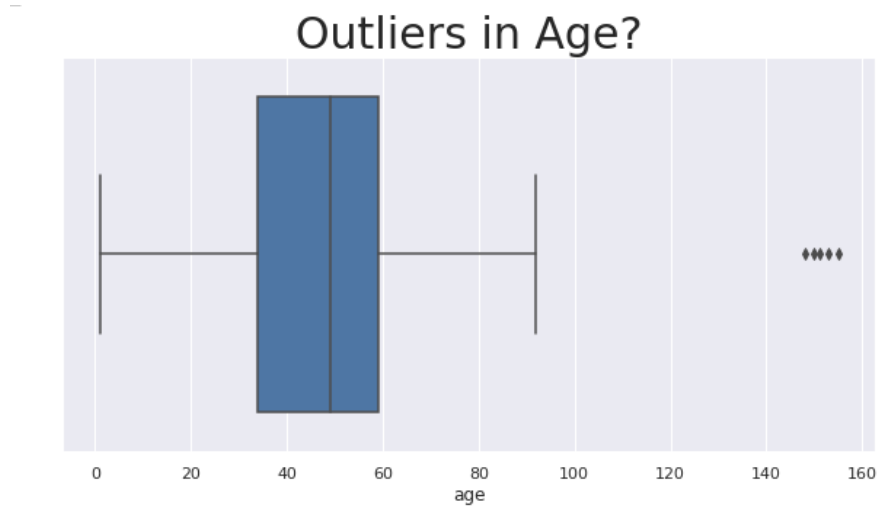
- Age Distribution:

Age has a range from 1 to 155 with a median of 49.



	x	y	width	height	Target	age
count	9555.000000	9555.000000	9555.000000	9555.000000	30227.000000	30227.000000
mean	394.047724	366.839560	218.471376	329.269702	0.316108	46.797764
std	204.574172	148.940488	59.289475	157.750755	0.464963	16.892940
min	2.000000	2.000000	40.000000	45.000000	0.000000	1.000000
25%	207.000000	249.000000	177.000000	203.000000	0.000000	34.000000
50%	324.000000	365.000000	217.000000	298.000000	0.000000	49.000000
75%	594.000000	478.500000	259.000000	438.000000	1.000000	59.000000
max	835.000000	881.000000	528.000000	942.000000	1.000000	155.000000

- Outliers in age:



	patientId	x	y	width	height	Target		class	age	gender	modality
5525	3b8b8777-a1f6-4384-872a-28b95f59bf0d	NaN	NaN	NaN	NaN	0		Normal	148	M	CR
13818	73aeea88-fc48-4030-8564-0a9d7fdecac4	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal		151	F	CR
21301	a4e8e96d-93a6-4251-b617-91382e610fab	NaN	NaN	NaN	NaN	0	No Lung Opacity / Not Normal		153	M	CR
32970	ec3697bd-184e-44ba-9688-ff8d5fbf9bbc	NaN	NaN	NaN	NaN	0		Normal	150	M	CR
34436	f632328d-5819-4b29-b54f-adf4934bbee6	NaN	NaN	NaN	NaN	0		Normal	155	F	CR

There looks to be a few outliers as the ages of these patients are above 148. These Patient IDs will be dropped.

Deciding Models and Model Building

As our objective is Localization (Detecting where the inflammation in the X-ray is) we would need to look at Object Detection models.

Object detection is the task of detecting instances of objects of a certain class within an image. The state-of-the-art methods can be categorized into two main types: one-stage methods and two stage-methods. One-stage methods prioritize inference speed, and example models include YOLO, SSD and RetinaNet. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN, and Cascade R-CNN.

Currently we've looked at implementing YOLO (Versions 3, 4 & 5), CHEXNET & SSD.

- YOLO (V3, V4 & V5):

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy.

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

The YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

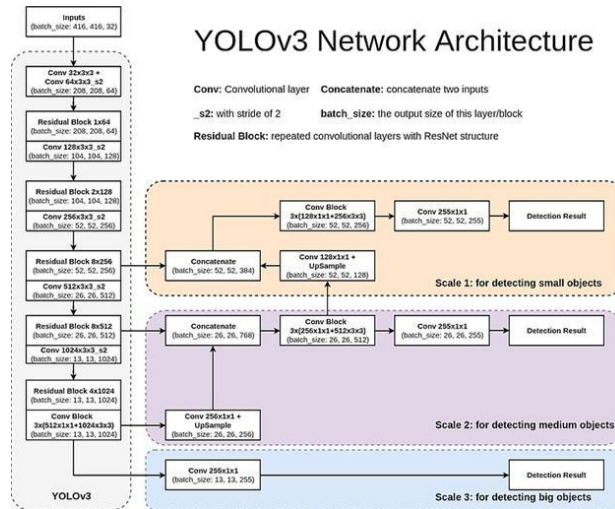
This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include YOLOv3, YOLOv4 & YOLOv5 .

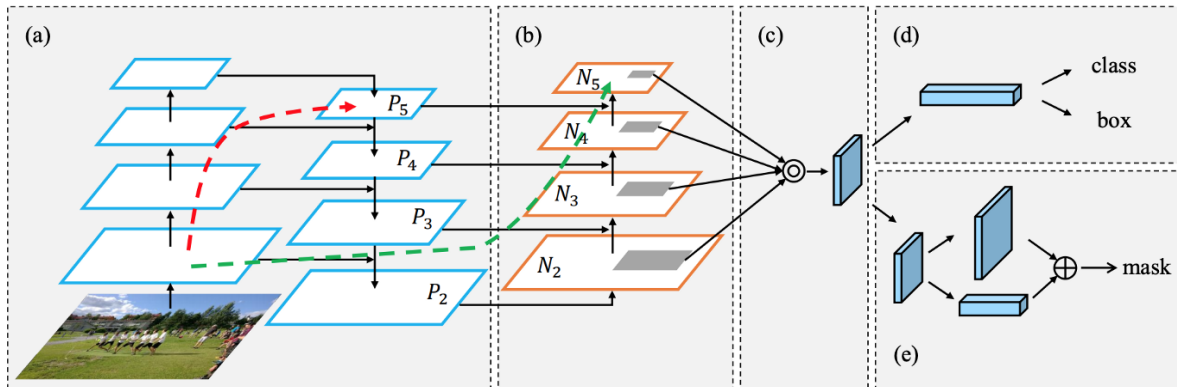
We've currently implemented the following versions of YOLO in our workbooks –

- YOLO V3**
- YOLO V4**
- YOLO V5**

We've currently only used 130 images out of our entire training set to validate the usage of these models. We will look at training these models on our entire dataset within the next couple of weeks.

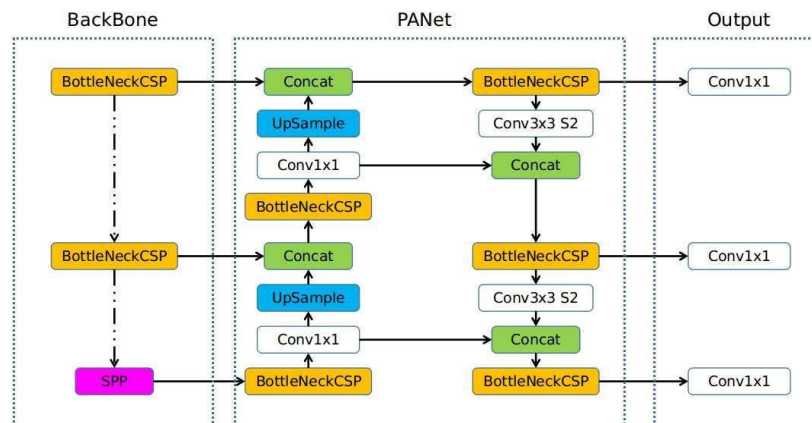


Architecture for Yolov3 ^[11]



Architecture for Yolov4 ^[12]

Overview of YOLOv5

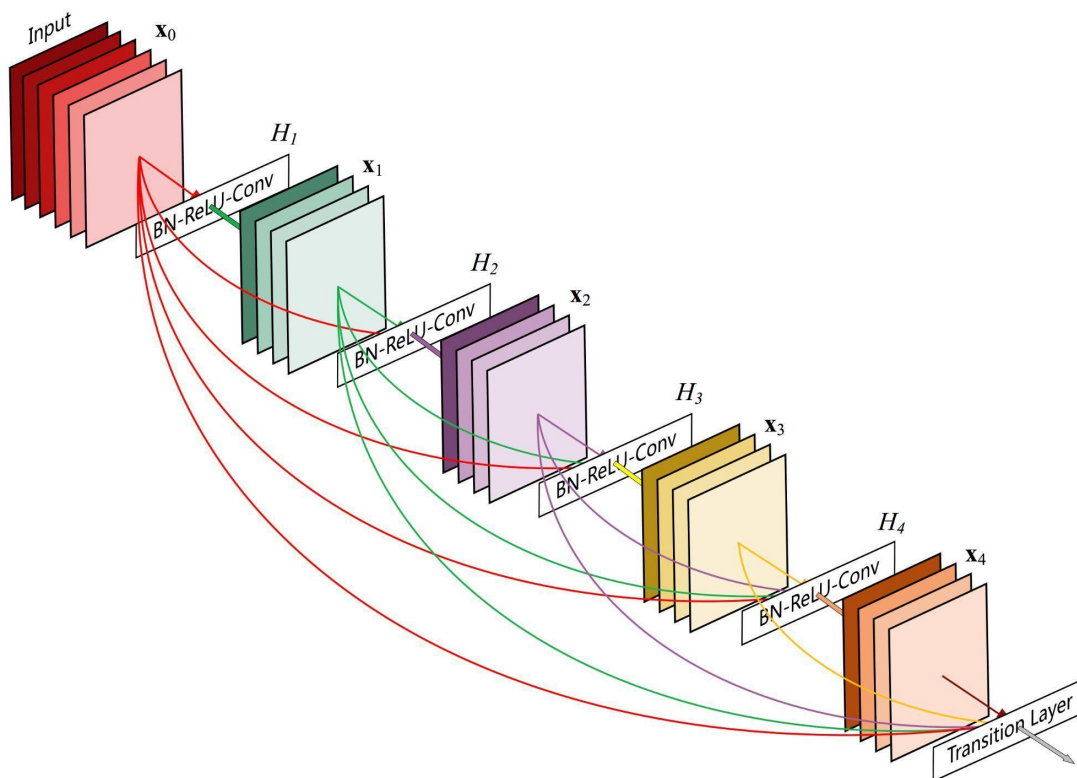


Architecture for Yolov5 ^[13]

- CHEXNET:

The CheXNet algorithm is a 121-layer deep 2D Convolutional Neural Network; a Densenet after Huang & Liu. The Densenet's multiple residual connections reduce parameters and training time, allowing a deeper, more powerful model. The model accepts a vectorized *two-dimensional image of size 224 pixels by 224 pixels*.

CheXNet is a 121-layer Dense Convolutional Network (DenseNet) (Huang et al., 2016) trained on the ChestX-ray 14 dataset. DenseNets improve flow of information and gradients through the network, making the optimization of very deep networks tractable. We replace the final fully connected layer with one that has a single output, after which we apply a sigmoid nonlinearity.



Architecture for Chexnet ^[14]

- SSD

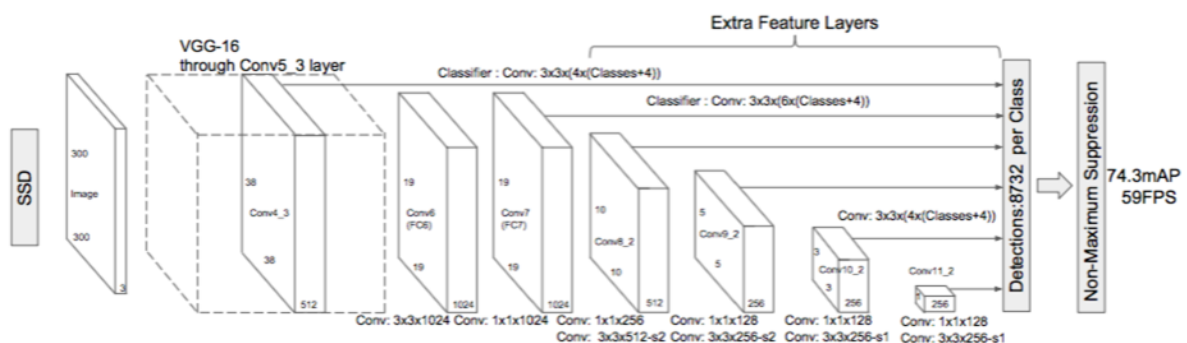
Single shot detectors like Yolo, a single pass is only needed to detect multiple objects in the image. SSD is much faster than two phase Region proposals methods like RCNN, Fast RCNN and Faster RCNN. Unlike Yolo it has predefined Anchor boxes and the final fully connected layer takes input from multiple intermittent layers. SSD is also better in accuracy and speed when compared YOLO V1. If input image size is $m \times n$ with K Anchors then the output will be class score $(c+4)$ bounding box coordinates multiplied by $m \times n$.

The architecture of SSD consists of 3 main components^[15]

- Base network
- Extra Feature Layer
- Prediction Layers

SSD's architecture builds on the venerable **VGG-16 architecture** but discards the fully connected layers. The reason VGG-16 was used as the base network is because of its strong performance in high quality image classification tasks.

Currently SSD with Pytorch and tensor flow is being implemented with a small set of images for training to check how the model works. We will train the entire datasets on this model in the coming week.



Architecture for SSD ^[15]

Step-by-step walk through the solution:

As mentioned above, we've used the publicly available dataset which was created by the Radiological Society of North America (RSNA).

This dataset contains 29,684 chest radiographs and can be accessed through the below link:
<https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data>

Dataset: The data is stored in a zip file labelled "rsna-pneumonia-detection-challenge.zip", which contains the following items:

1. stage_2_train_images: This folder contains all the training DICOM images.
2. stage_2_train_labels.csv: This file contains the corresponding patientID images to the folder 'stage_2_train_images' and contains the bounding box of areas of pneumonia detected in each image along with a target label of 0 or 1 for pneumonia detected.
3. stage_2_detailed_class_info.csv: This file contains the corresponding patientID images to the folder 'stage_2_train_images' and contains the target classes of the images.
4. stage_2_test_images: This folder contains all the test DICOM images

EDA: Firstly, we understood the data with a comprehensive EDA on the available set. Based on the insights of the same and the literature available, we went ahead with selecting the model.

Multiple models to analyze: We decided to go through multiple models to proceed with the object detection. These are YOLO, SSD and Chexnet. All these came up with different sets of requirements for pre-processing as well as model-building.

Conversion and resizing of images: All the images are stored as .dcm images. So, to use Object detection models, we would need to convert these images from .dcm to either .jpg or .png formats.

Prediction and Visualization: As a group, we were able to run all the three models after initial hiccups. Productionizing these models was a challenge for all three models that we were looking into. Metrics and visualization with respect to all the models are given in the later section.

Model Evaluation:

Describe the final model in detail. What was the objective, what parameters were prominent, and how did you evaluate the success of your models?

- YOLO (V3, V4 & V5): (YOLOv5 has been used)

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy.

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

As our objective was Object Detection, the YOLO models were perfect therefore we decided that we would test out the various YOLO models made available through Roboflow.

YOLOv5 outperformed all other versions of YOLO therefore we've used YOLOv5 as one of our final models. We've included all relevant visualizations for the YOLOv5 model under the 'Visualizations' sub-section.

- CHEXNET:

For Chexnet we needed the following alterations in the image:

1. We had to convert the dicom images into jpg format
2. Since chexnet model takes input images of size 224*224, we had to resize the images from 1024*1024 to 224*224

After the conversion, we ran the Chexnet model built on Densenet-121 architecture. We used 10.5k rows as the training set, 1.5k as the validation set and the rest 3k as the test set.

Used a skeleton using the github repo of Chou Bruce^[16]

Ideally would have wanted to run close to ~40-50 epochs, but due to constraints on running resources, wasn't able to proceed with the same.

Here's the summary of the params used:

- *Total params: 7,039,554*
- *Trainable params: 6,955,906*
- *Non-trainable params: 83,648*

Other config params:

- *base_model_name=DenseNet121*
- *epochs=10*
- *batch_size=64*
- *initial_learning_rate=0.001*
- *min_lr=1e-8*

From the final classification obtained using Chexnet, we created a class activation map, which was further used to create the bounding box and hence compare against the ground truth. We will visualize a few in the visualization section.

Also, we look at the training and validation metrics as per the model output.

- SSD:

Input image (300 *300) is given to VGG16 network to extract feature map. SSD makes 8732 predictions i.e. 8732 bounding boxes. Non Max suppression is used to limit the bounding boxes by checking the confidence of detections ,top 200 predictions .

Training of SSD:

Images with Ground truth boxes -Annotations

CNN is used for boxes of diff aspect ration at each location with different scales .The bounding box with highest overlap(IOU)is taken as the final output.

To run the pytorch version using Anaconda and without nvidia gpu with the following config:

```
voc = {  
    'num_classes': 2,  
    'lr_steps': (80000, 100000, 120000),  
    'max_iter': 120000,  
    'feature_maps': [38, 19, 10, 5, 3, 1],  
    'min_dim': 300,  
    'steps': [8, 16, 32, 64, 100, 300],
```

```
'min_sizes': [30, 60, 111, 162, 213, 264],
'max_sizes': [60, 111, 162, 213, 264, 315],
'aspect_ratios': [[2], [2, 3], [2, 3], [2, 3], [2], [2]],
'variance': [0.1, 0.2],
'clip': True,
'name': 'VOC',
}
```

The biggest challenge was to prepare the data set with correct annotation xml , converting images into jpeg . Making the split between Training and Testing databases like VOC 2007 and replicate the detection ratio.

The checkpoint callback was used which saves the best weights of the model, so next time we want to use the model, we do not have to spend time training it. The early stopping callback stops the training process when the model starts becoming stagnant, or even worse, when the model starts overfitting. Since we set `restore_best_weights` to True, the returned model at the end of the training process will be the model with the best weights (i.e. low loss and high accuracy).

Comparison to benchmark:

How does your final solution compare to the benchmark you laid out at the outset? Did you improve on the benchmark? Why or why not?

- YOLOv5:

In our benchmark that we laid out at the outset, we had only used 130 images out of our entire training set to validate the usage of these models.

Now that we've trained these models on more images and increased the number of epochs our validation accuracies have improved.

Although the accuracies of the model have improved, we've still faced a few limitations while implementing these models. These limitations are highlighted under the 'Limitations' sub-section.

- CHEXNET:

The Chexnet model gave a very high ROC and very low loss in classifying the data. Also, with the help of the class activation maps, we were able to get the bounding boxes created which look pretty good. It works well against the benchmark that we had set out. The reason for good performance was a mixture of proper class balance, proper pre-processing and model-building as well as the Densenet-121 structure used for detecting pneumonia.

- SSD:

Tensor flow SSD implementation for training 30226 images were used and for validation 20000 images were used ,The model gave 98% accuracy as it saw a larger dataset than YOLO.

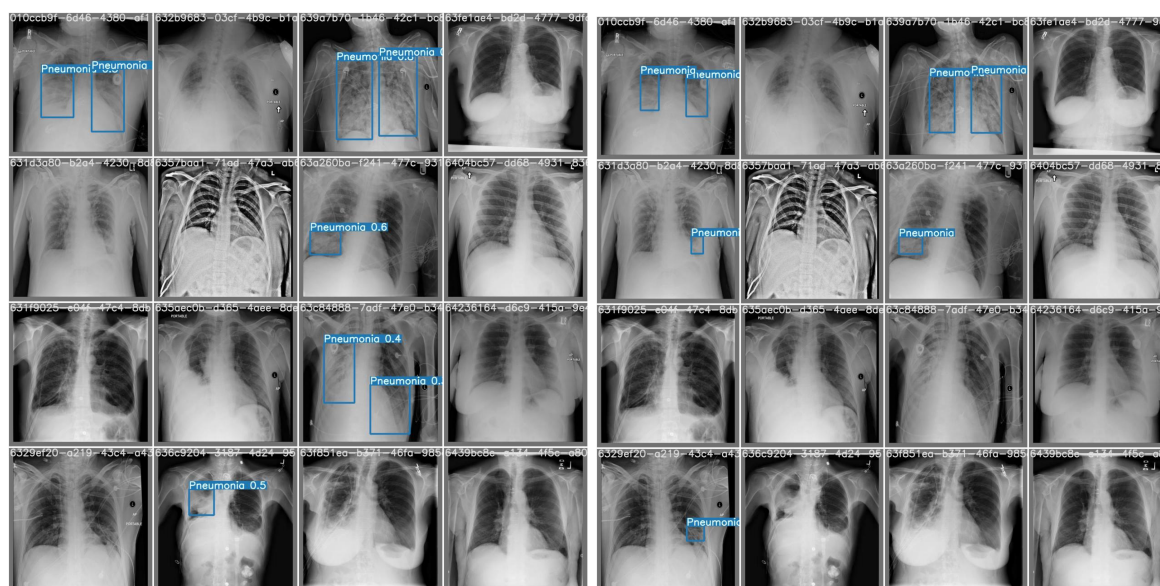
Visualizations:

In addition to quantifying your model and the solution, please include all relevant visualizations that support the ideas/insights that you gleaned from the data.

- YOLOv5:

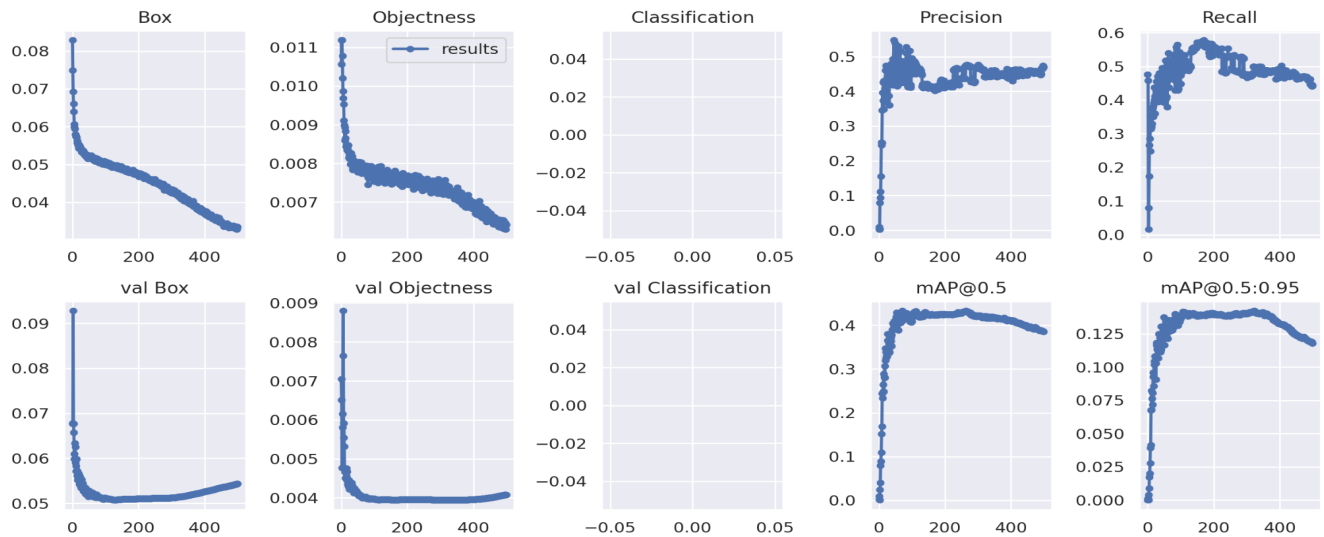
Predictions

Actual Labels



The YOLOv5 model correctly classifies a few dicom images but the sizes of the bounding boxes are not accurate. There are a few cases where the model incorrectly predicts a bounding box. In order to solve this we need to train this model with more images but unfortunately due to Roboflow pro costing upwards of \$1,000 we weren't able to include more images in the free version.

Training Metrics:

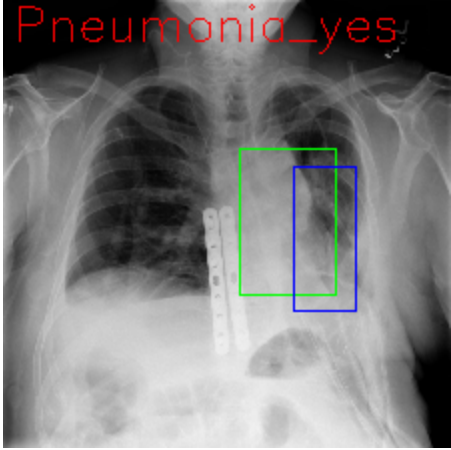
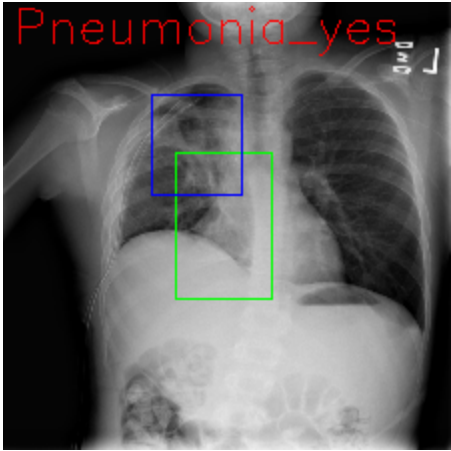
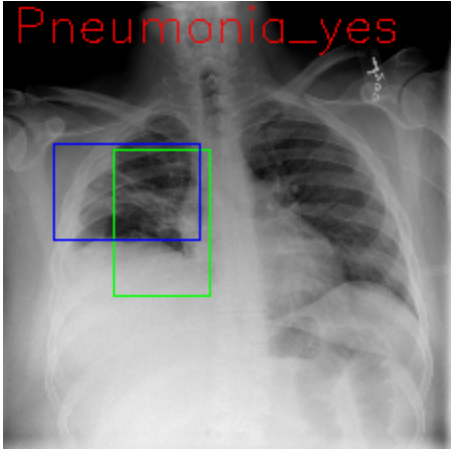



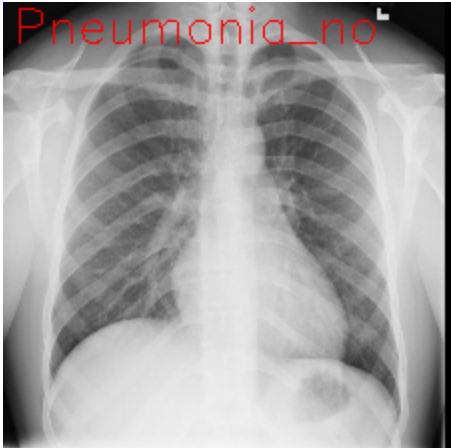
- CHEXNET:

Predictions and bounding boxes:

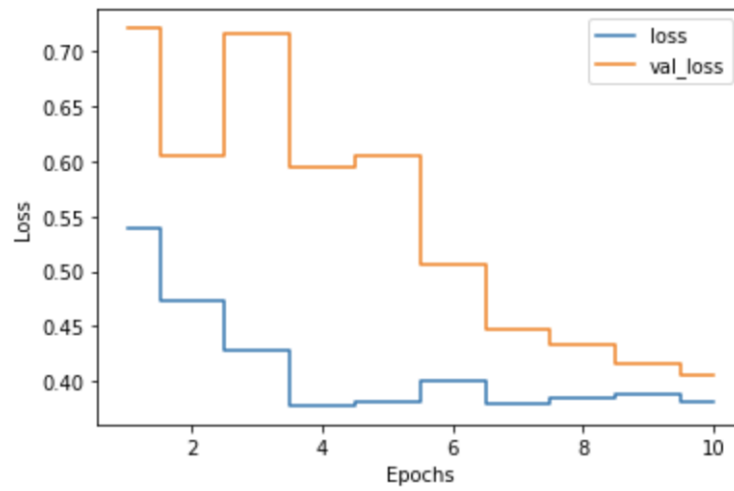
Green box represents the *predicted box*

Blue box represents the *ground truth*

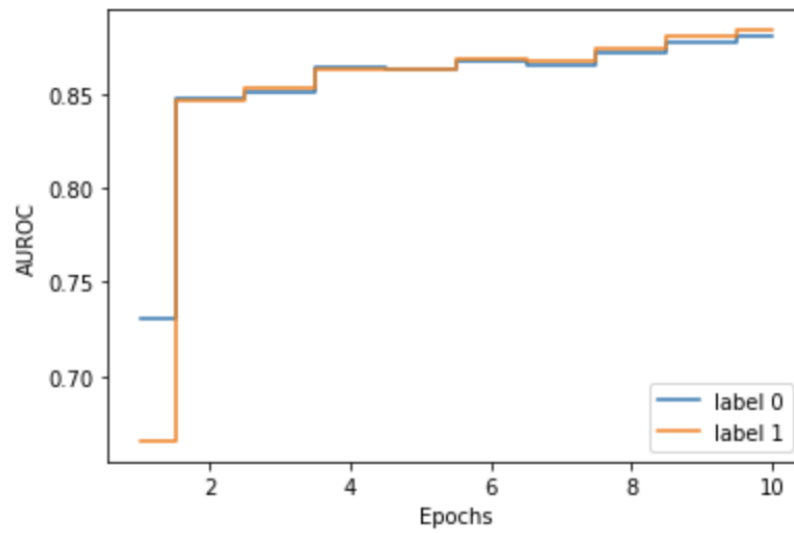
 <p>Pneumonia_yes</p>	<p>File name: 01cd2ba5-2baf-44b2-bf15-ee57e1ea4599.png</p> <p>Predicted class: Pneumonia yes</p> <p>Actual class: Pneumonia yes</p> <p>Predicted box (x1p, y1p, x2p, y2p) in 224 dim: 118, 74, 166, 147</p> <p>Actual box (x1, y1, x2, y2) in 224 dim: 145, 83, 176, 155</p>
 <p>Pneumonia_yes</p>	<p>File name: 0ad9934b-bf35-497d-8082-1ecbc955d6de.png</p> <p>Predicted class: Pneumonia yes</p> <p>Actual class: Pneumonia yes</p> <p>Predicted box (x1p, y1p, x2p, y2p) in 224 dim: 86, 75, 134, 148</p> <p>Actual box (x1, y1, x2, y2) in 224 dim: 74, 46, 119, 96</p>
 <p>Pneumonia_yes</p>	<p>File name: 0a2c130c-c536-4651-836d-95d07e9a89cf.png</p> <p>Predicted class: Pneumonia yes</p> <p>Actual class: Pneumonia yes</p> <p>Predicted box (x1p, y1p, x2p, y2p) in 224 dim: 55, 74, 103, 147</p> <p>Actual box (x1, y1, x2, y2) in 224 dim: 25, 71, 98, 119</p>

	<p>File name: a0426825-2cd7-47ae-b20c-f536a1ec2249.png</p> <p>Predicted class: Pneumonia no</p> <p>Actual class: Pneumonia no</p> <p>Predicted box (x1p, y1p, x2p, y2p) in 224 dim: NA</p> <p>Actual box (x1, y1, x2, y2) in 224 dim: NA</p>
	<p>File name: a045b782-7673-4f3f-a018-07eb3ae96321.png</p> <p>Predicted class: Pneumonia no</p> <p>Actual class: Pneumonia no</p> <p>Predicted box (x1p, y1p, x2p, y2p) in 224 dim: NA</p> <p>Actual box (x1, y1, x2, y2) in 224 dim: NA</p>

Performance Metrics:

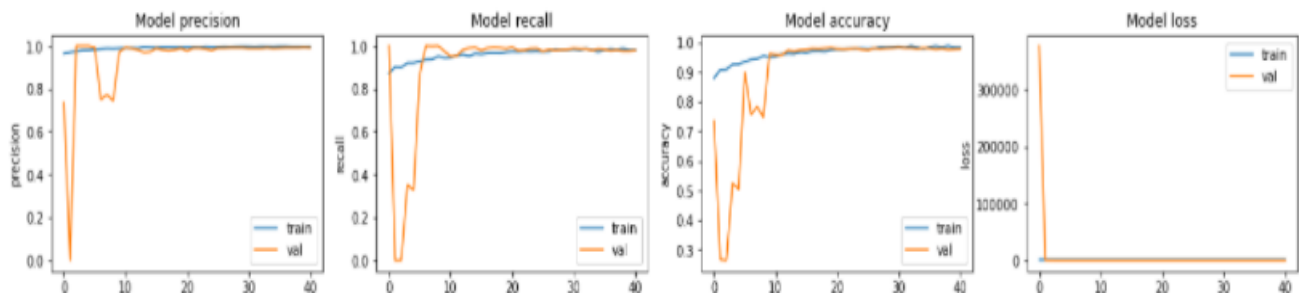


Training and validation loss for the epochs we ran the Chexnet model



AUROC for both the classes. 0 signifies the class with no pneumonia and 1 signifies the class with pneumonia

- SSD:



Implications:

How does your solution affect the problem in the domain or business? What recommendations would you make, and with what level of confidence?

At a macro level, all the three models yielded good results. Ideally an ensemble of all the three models would give a much better performance as there is no fast real-time processing capability required. The ensemble can work as a very good first level screen which can be then worked upon by the radiologists pretty well.

- YOLOv5:

With this model at its current stage, it can help healthcare specialists to semi-automate the process of detecting pneumonia in lung images. This model can detect pneumonia with ease but with a certain degree of variance. The precise area needs to be further investigated by the healthcare specialists though as the predicted area might be too large or not accurate enough.

Hence if there is an immediate requirement for model usage, these models can serve as a first line of defense for screening the DICOM images.

- CHEXNET:

CheXnet is traditionally known as a great substitute for all the radiologists out there as it predicts the traditional 10 diseases almost the best. With the current model prediction of

around 89% AUROC, there is still scope for a lot more. Also, it can serve as a great tool for detection of lung opacities. The model currently trained has a huge potential if the hyper params are properly tuned. It can at least help as a good first level screen which can be then worked upon by the radiologists pretty well.

Limitations:

What are the limitations of your solution? Where does your model fall short in the real world? What can you do to enhance the solution?

- YOLOv5:

As we've used the Roboflow frameworks for all Yolo models, we've encountered a few limitations. Our primary limitation was that we could use only about 13,000 images from our Training set due to Roboflow only accepting 13,000 in their free version. Ideally, we would've needed to upgrade to the pro version of Roboflow but that wasn't feasible as their pro version cost upwards of \$1000.

Our secondary limitation was that it took us quite some time to train these models and unfortunately our systems couldn't handle the load of running these models.

Therefore assuming there are no time constraints and in order to fully automate the process of detecting pneumonia, these models need to be trained on more dicom images and also need to be trained on more epochs to improve the model accuracies.

- CHEXNET:

Few limitation of Chexnet model that has been used here:

1. The existing literature is very well supported to document classification using chexnet. However, there are very limited efforts done to use it for object detection. That posed some teething issues while starting to explore the same.
2. We built bounding boxes after getting the center of mass of the class activation map (using some fine tunings).
3. We trained the model using 10 epochs only. Had we got more time and better processing capacity, we would have used 50+ epochs which would have given us even better performances.

4. Some pictures in the dataset had a different alignment with other pictures. That could have been better.

Closing Reflections:

What have you learned from the process? What would you do differently next time?

To start with let's comment about the data volume, velocity, variety, veracity and values.

The dataset had a good **volume** of images. Roughly 30k which gave us a good amount to train, validate and test properly. Given that this dataset is used to build models to help radiologists, it needs to be updated frequently. For the purpose of this capstone, it wasn't necessarily needed to update it.

In terms of **veracity**, it can be said that the dataset had real images and values and hence it can be used for solving other real-life problems. Also, the classes were properly balanced. We had a good mix of images with and without lung opacities (or pneumonia).

In terms of model processing and evaluation, we ideally wanted to create an ensemble model by combining all the models that we've used but unfortunately we weren't able to complete this in time due to various factors. If there wasn't a time constraint then we would've been able to create the ensemble model. We also faced a few issues with the processing resources due to which the speed of work was impacted.

We learned to get through and solve a lot of things about productionizing the model. We went through the Single Shot Detectors and YOLO models. Also, we learned how to productionize and build models using Chexnet.

In conclusion, through the help of this project, we've had a better understanding of how to solve real world problems. This project has helped us to tackle multiple scenarios which in turn has helped us gain more knowledge in the field of data science.

Therefore we sincerely thank you for this opportunity.

References:

1. Le, W.T.; Maleki, F.; Romero, F.P.; Forghani, R.; Kadoury, S. Overview of machine learning: Part 2. *Neuroimaging Clin. N. Am.* 2020, 30, 417–431.
2. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*; Springer International Publishing: Cham, Switzerland, 2015; pp. 234–241.
3. Milletari, F.; Navab, N.; Ahmadi, S.A. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV)*, Stanford, CA, USA, 25–28 October 2016; pp. 565–571.
4. Jeelani, H.; Martin, J.; Vasquez, F.; Salerno, M.; Weller, D. Image quality affects deep learning reconstruction of MRI. In *Proceedings of the 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, Washington, DC, USA, 4–7 April 2018; pp. 357–360.
5. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* 2012, 25, 1097–1105.
6. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* 2014, arXiv:1409.1556.
7. Zhang, D.; Ren, F.; Li, Y.; Na, L.; Ma, Y. Pneumonia Detection from Chest X-ray Images Based on Convolutional Neural Network. *Electronics* **2021**, 10, 1512. <https://doi.org/10.3390/electronics10131512>.
8. Bandar Almaslukh, "A Lightweight Deep Learning-Based Pneumonia Detection Approach for Energy-Efficient Medical Systems", *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 5556635, 14 pages, 2021. <https://doi.org/10.1155/2021/5556635>.
9. Shangjie Yao, Yaowu Chen, Xiang Tian, Rongxin Jiang, "Pneumonia Detection Using an Improved Algorithm Based on Faster R-CNN", *Computational and Mathematical Methods in Medicine*, vol. 2021, Article ID 8854892, 13 pages, 2021. <https://doi.org/10.1155/2021/8854892>

10. Hashmi, M.F.; Katiyar, S.; Keskar, A.G.; Bokde, N.D.; Geem, Z.W. Efficient Pneumonia Detection in Chest Xray Images Using Deep Transfer Learning. *Diagnostics* **2020**, *10*, 417. <https://doi.org/10.3390/diagnostics10060417>
11. Rosina De Palma. Yolov3 Architecture:Best Model in Object Detection <https://bestinau.com.au/yolov3-architecture-best-model-in-object-detection/>
12. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection <https://arxiv.org/pdf/2004.10934.pdf>
13. Seekfire. Overview of Model Structure about Yolov5 <https://github.com/ultralytics/yolov5/issues/280>
14. Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, Andrew Y. Ng, "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning" <https://arxiv.org/abs/1711.05225>
15. Liu, Wei and Anguelov, Dragomir and Erhan, Dumitru and Szegedy, Christian and Reed, Scott and Fu, Cheng-Yang and Berg, Alexander C;"SSD: Single Shot MultiBox Detector",Lecture Notes in Computer Science. 2016.http://dx.doi.org/10.1007/978-3-319-46448-0_2.
16. Chexnet like models using Keras by Bruce Chou <https://github.com/brucechou1983/CheXNet-Keras>