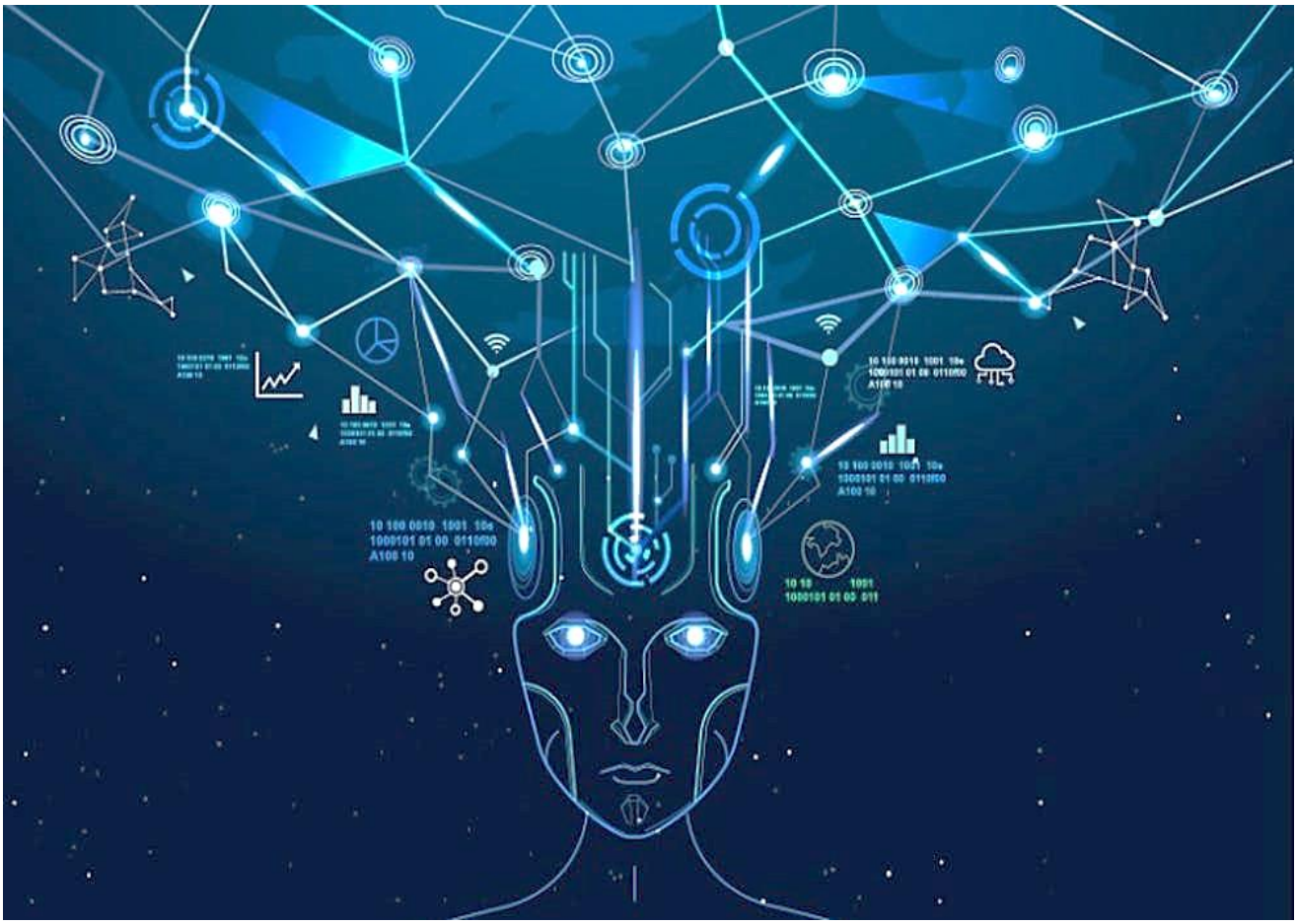


PGP AI ML – CAPSTONE FINAL SUBMISSION

PNEUMONIA DETECTION CHALLENGE

GROUP A CV BATCH 1

NOVEMBER 2021

TABLE OF CONTENTS

1. SUMMARY OF PROBLEM STATEMENT, DATA AND FINDINGS

THE REAL PROBLEM - Pneumonia accounts for over 15% of all deaths of children under 5 years old internationally. Accurately diagnosing pneumonia requires review of a chest radiograph (CXR) by highly trained specialists and confirmation through clinical history, vital signs and laboratory exams. Pneumonia usually manifests as an area or areas of increased opacity on CXR. However, the diagnosis of pneumonia on CXR is complicated because of a number of other conditions in the lungs such as fluid overload (pulmonary edema), bleeding, volume loss (atelectasis or collapse), lung cancer, or post-radiation or surgical changes. A number of factors such as positioning of the patient and depth of inspiration can alter the appearance of the CXR, complicating interpretation further. Tissues with sparse material, such as lungs which are full of air, do not absorb the X-rays and appear black in the image. Dense tissues such as bones absorb X-rays and appear white in the image

We need to build a Pneumonia detection model to detect a visual signal for pneumonia in medical images. Specifically, the algorithm needs to automatically locate lung opacities on chest radiographs, providing affected area details through bounding box

DATA - Data provided is Dicom original images: - Medical images are stored in a special format called DICOM files (*.dcm). They contain a combination of header metadata as well as underlying raw image arrays for pixel data. While we are theoretically detecting “lung opacities”, there are lung opacities that are not pneumonia related. In the data, some of these are labeled “Not Normal No Lung Opacity”. This extra third class indicates that while pneumonia was determined not to be present, there was nonetheless some type of abnormality on the image and oftentimes this finding may mimic the appearance of true pneumonia. Lung Opacity class refers to Pneumonia cases. There are around 26000 2D single channel CT images in the pneumonia dataset that provided in DICOM format.

FINDINGS-

- “Faster R-CNN” model predicting bounding boxes of pneumonia patients with confidence intervals.
- VGG19 seems to be the best model with highest recall, we will continue with Faster RCNN since this is an object detection problem

2. OVERVIEW OF THE FINAL PROCESS- DATA PRE-PROCESSING STEPS AND THE ALGORITHMS USED

Like most data science undertakings, we followed the CRISP-DM approach for this project. Since the dataset comprised of Dicom images, we installed Pydicom library to read the images. For the detectron2 model, we also used Albumentations library for Data augmentations on the data set such as Horizontal flip, Cropping, Scaling and so on.

We performed a number of diagnostics on the data using visualizations to gather preliminary insights. Next up, we ran a bunch of models such as VGG19, Resnet 50, InceptionNet v3, U-Net and Faster R-CNN.

We finally evaluate all model performance using classification matrix to select and pickle our model that can be used at the backend of API for further predictions.

Below is a figure depicting the process flow of our project approach -

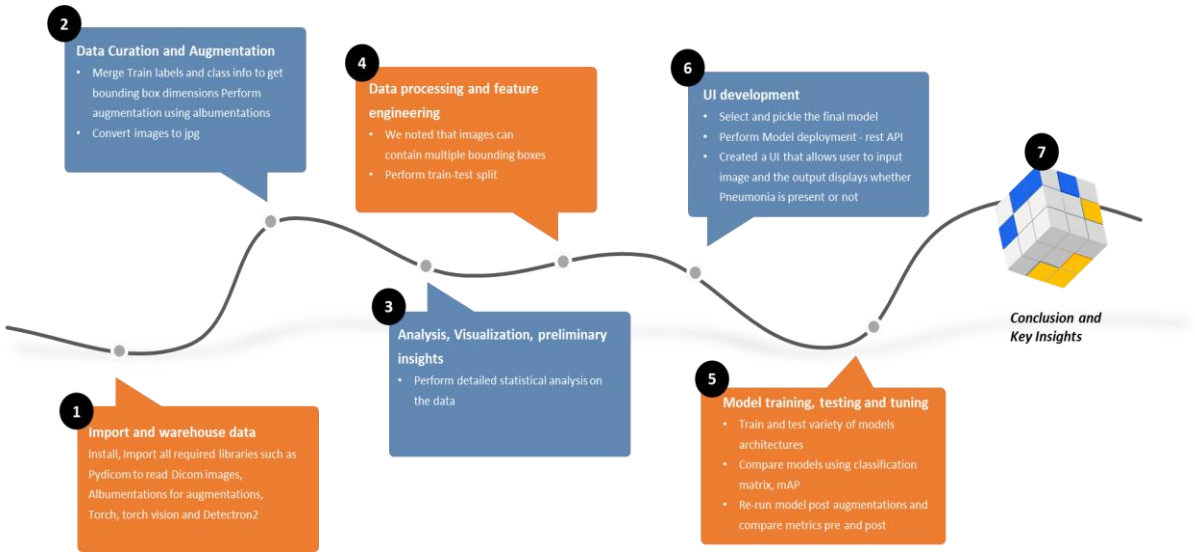


Figure 1: Project Approach

3. STEP-BY-STEP WALK THROUGH THE SOLUTION

As mentioned previously, we deployed a number of models such as VGG19, ResNet 50, InceptionNet v3, U-Net and Faster R-CNN

VGG19

VGG19 is a variant of VGG model which in short consists of 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer). There are other variants of VGG like VGG11, VGG16 and others. VGG19 has 19.6 billion FLOPs.

The input to VGG based convNet is a 224*224 RGB image. Pre-processing layer takes the RGB image with pixel values in the range of 0–255 and subtracts the mean image values which is calculated over the entire ImageNet training set. VGG19 Variation has 19 weight layers consisting of 16 convolutional layers with 3 fully connected layers and same 5 pooling layers.

VGG-19 ARCHITECTURE

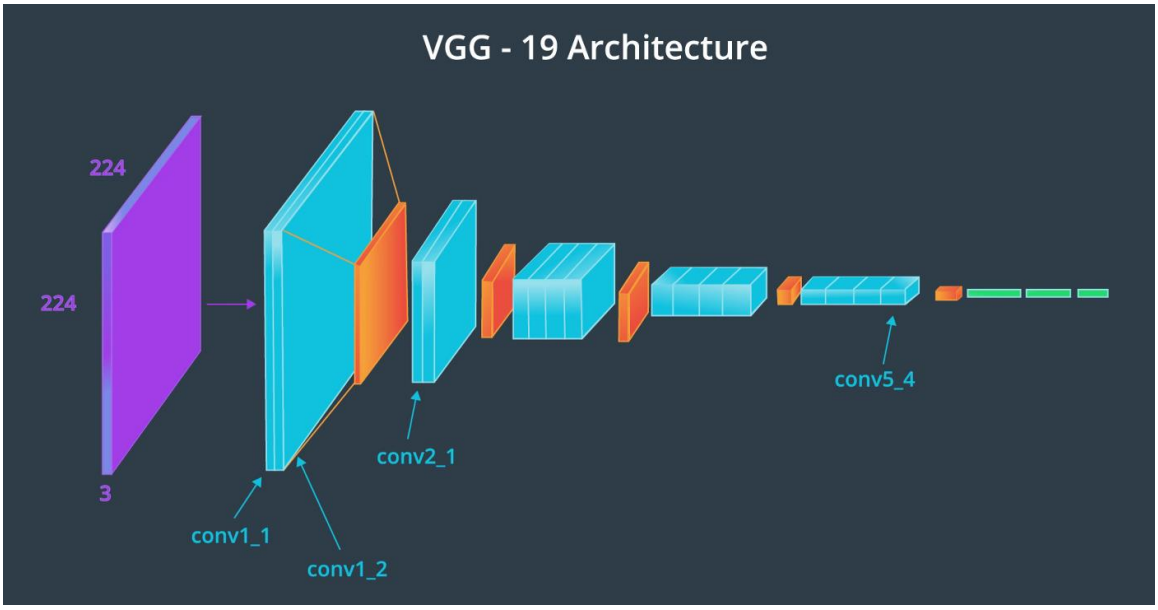


Figure 2: VGG19 Architecture

The output of our VGG19 model was as follows -

```
print(classification_report(y,pred[:len(y)]))
```

	precision	recall	f1-score	support
0.0	0.94	0.89	0.91	101
1.0	0.66	0.78	0.71	27
accuracy			0.87	128
macro avg	0.80	0.83	0.81	128
weighted avg	0.88	0.87	0.87	128

RESNET 50

ResNet-50 is a convolutional neural network that is 50 layers deep. Note that there is only one 3x3 convolution rather than two. 1x1 convolutions are used to map in lower dimension and then perform 3x3 convolution and then remap them to higher dimensions. This way the training time will be less.

The other part to note is in ResNet 50 when there is dimension change then the authors used 1x1 convolutions at x to make the dimension same.

RESNET 50 ARCHITECTURE

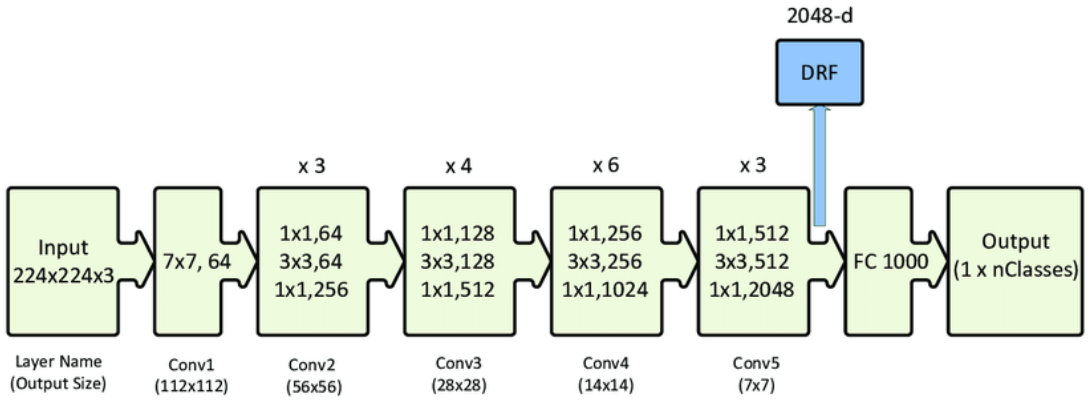


Figure 3: ResNet50 Architecture

The output of our RESNET 50 model was as follows -

```
print(classification_report(y,predr[:len(y)]))
```

	precision	recall	f1-score	support
0.0	0.98	0.80	0.88	105
1.0	0.50	0.91	0.65	23
accuracy			0.82	128
macro avg	0.74	0.86	0.76	128
weighted avg	0.89	0.82	0.84	128

INCEPTION NET V3

Inception-v3 is a convolutional neural network architecture from the Inception family that makes several improvements including using Label Smoothing, Factorized 7 x 7 convolutions, and the use of an auxiliary classifer to propagate label information lower down the network (along with the use of batch normalization for layers in the sidehead).

The architecture of an Inception v3 network is progressively built, step-by-step, as explained below:
1. Factorized Convolutions: this helps to reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency.

2. Smaller convolutions: replacing bigger convolutions with smaller convolutions definitely leads to faster training. Say a 5 × 5 filter has 25 parameters; two 3 × 3 filters replacing a 5 × 5 convolution has only 18 (3*3 + 3*3) parameters instead.

INCEPTION NET V3 ARCHITECTURE

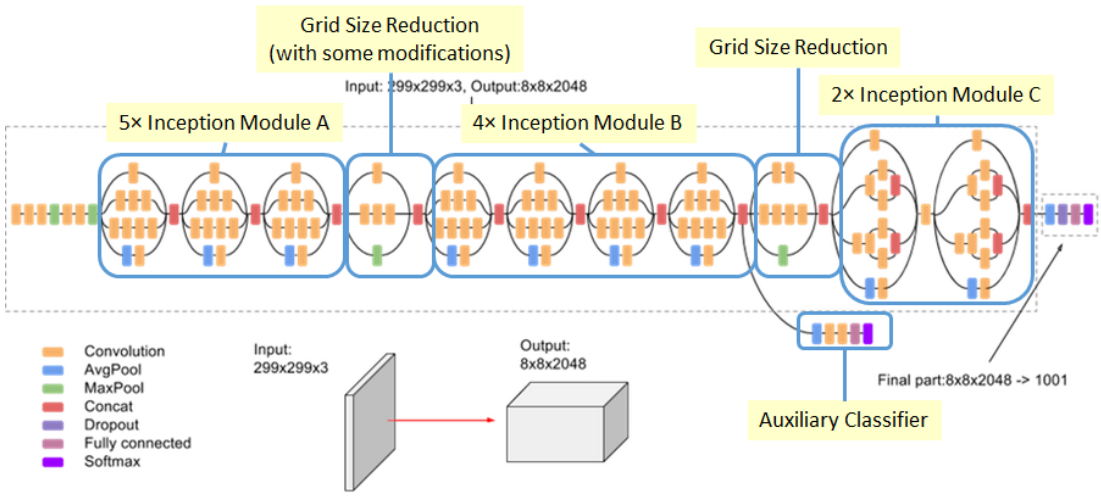


Figure 4: InceptionV3 Architecture

The output of our INCEPTION V3 model was as follows -

```
print(classification_report(y,predi[:len(y)]))
```

	precision	recall	f1-score	support
0.0	0.96	0.87	0.91	107
1.0	0.55	0.81	0.65	21
accuracy			0.86	128
macro avg	0.75	0.84	0.78	128
weighted avg	0.89	0.86	0.87	128

U-NET

U-Net model was developed for Bio Medical Image Segmentation. The model involves an encoder path and a decoder path. The encoder consists of a stacks of convolutions and max pooling layers. This helps to capture the context of the image. The decoder path is symmetric to the encoder it consists of stacks of up sampling and convolution layers.

Initially U-Net with pre-trained Resnet 50 was tried as encoder, validation accuracy was not satisfactory and number of predicted false positives were high. Idea of using pre-trained weights was dropped.

Next Customized U-Net model was built from scratch and image size converted to 64 by 64. Model was trained on the 50% of the images (10673 images) and validation samples were 2669 images. **Model achieved accuracy of 66%.**

U-NET ARCHITECTURE

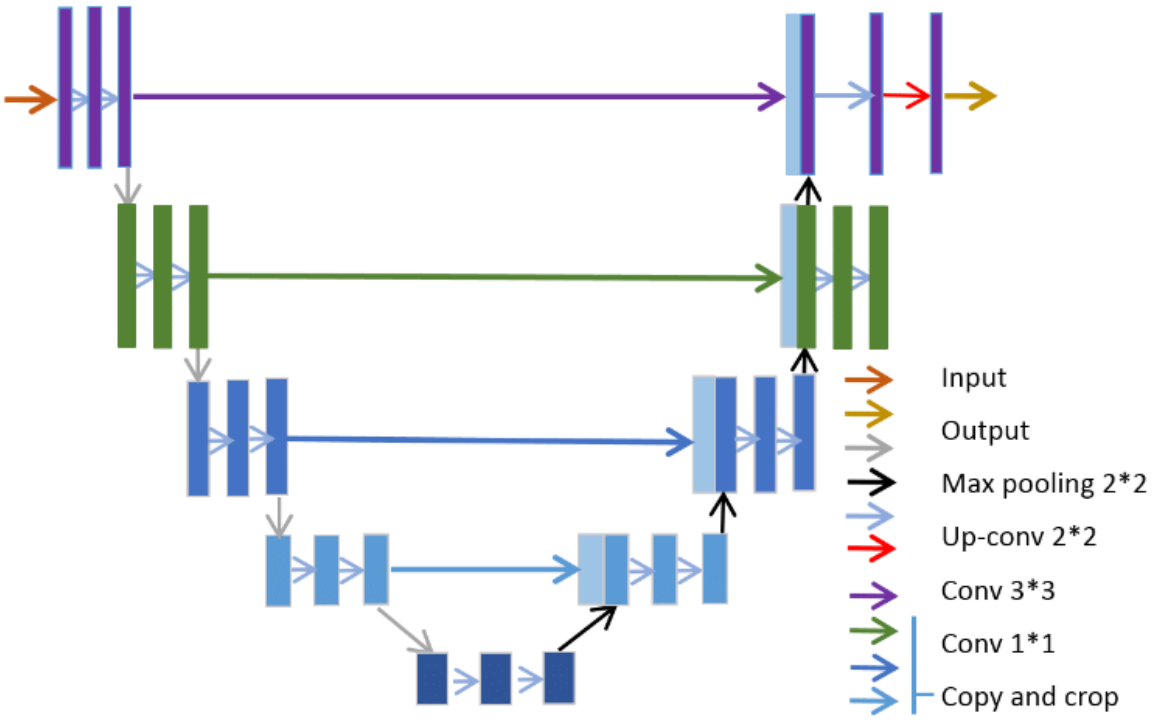
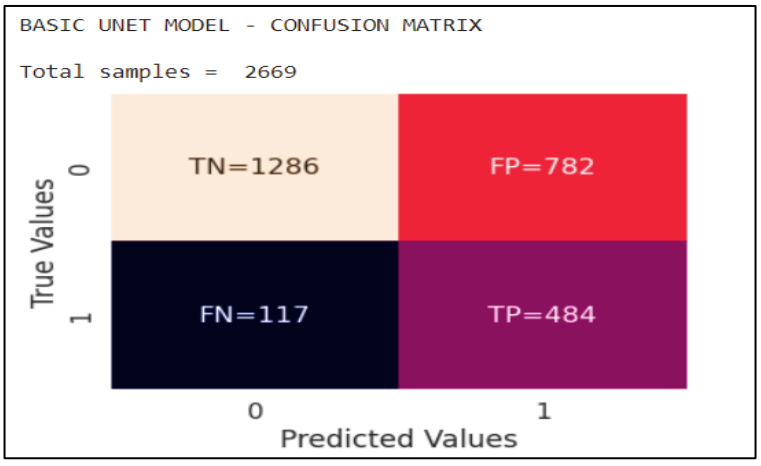


Figure 5 : U-NET Architecture

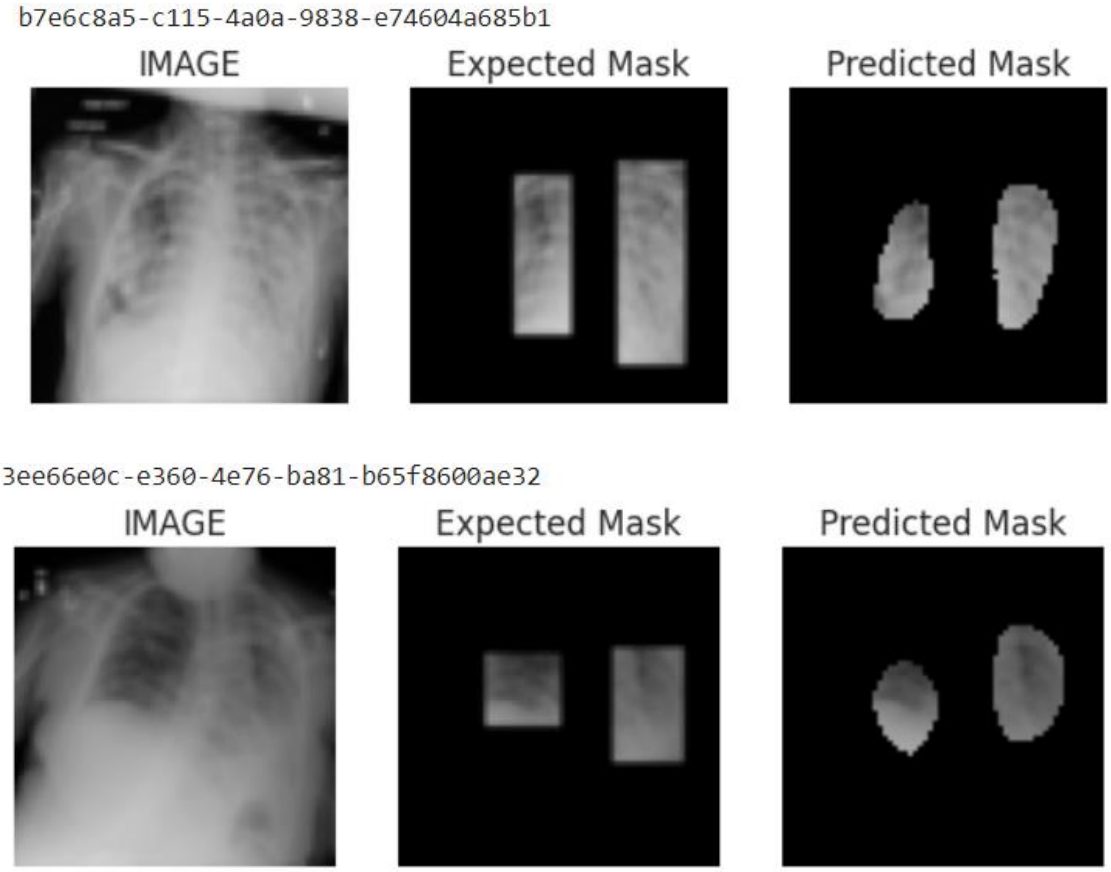
Confusion matrix output:



CUSTOM-UNET MODEL : CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.92	0.62	0.74	2068
1	0.38	0.81	0.52	601
accuracy			0.66	2669
macro avg	0.65	0.71	0.63	2669
weighted avg	0.80	0.66	0.69	2669

Few samples of the predicted bounding box using the customized U-Net Model:



FASTER R-CNN USING DETECTRON2

Faster R-CNN model was developed using Detectron2 on Pytorch. Detectron2 includes all the models that were available in the original Detectron, such as Faster R-CNN, Mask R-CNN, RetinaNet, and DensePose. It also features several new models, including Cascade R-CNN, Panoptic FPN, and TensorMask. All dicom images are converted to JPG as Detectron2 can't directly read dicom images. We prepare the dataset into Json format and then feed into the Detectron2 model for training. We apply transfer learning of the Faster R-CNN model i.e., pre-trained model on the COCO dataset. We are using cfg or better configs

which represents the complete configuration of a object detection model. These configurations are stored within a YAML-file and can be easily received from the modelzoo.

After the configuration is complete, we'll use the DefaultPredictor class to make predictions. Every dataset is associated with metadata. It is a key-value mapping that contains information about the dataset. It can be used to further interpret the dataset. This information can later be used for data augmentation, evaluation, visualization

FASTER R-CNN ARCHITECTURE

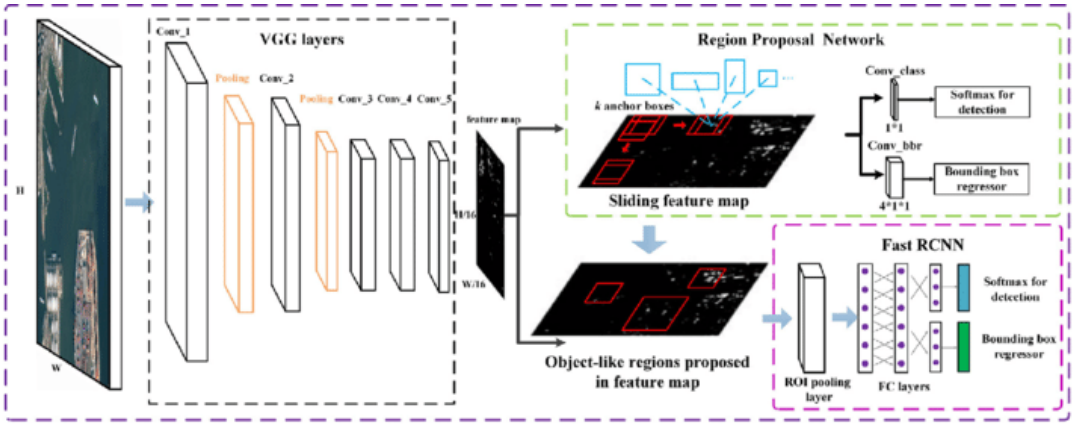
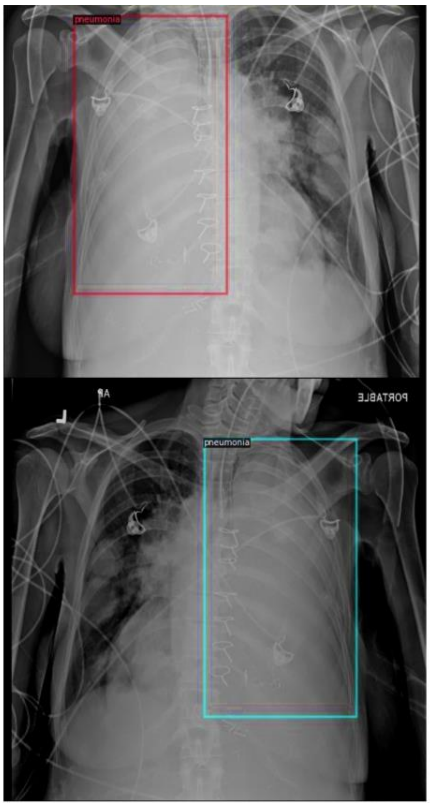


Figure 6 : Faster R-CNN Architecture

AUGMENTATION:

Standard augmentation function for images with bounding boxes was not feasible and hence we developed a pipeline using Albumentations library from PyPi. Augmentation was performed only on 10% of dataset sampled randomly.

The X-ray picture shows that augmentation is working with **horizontal flip of image** with bounding box.



The output of Faster R-CNN model was as follows -

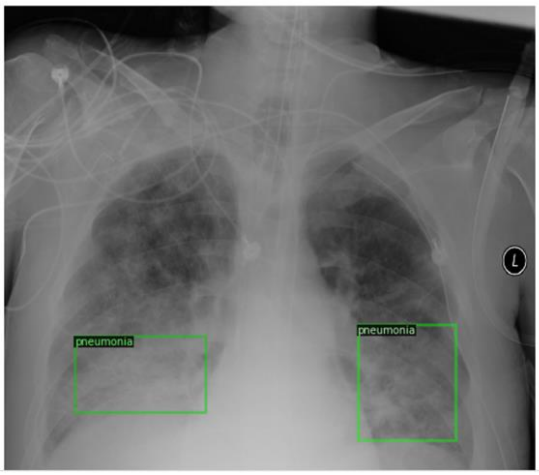
```
print(classification_report(df_aug['ground_truth'], df_aug['predicted']))
accuracy = accuracy_score(df_aug['ground_truth'], df_aug['predicted'])
print('Model accuracy is: ', accuracy)
```

```
              precision    recall  f1-score   support

           0       0.97      0.44      0.60      2076
           1       0.39      0.96      0.55       769

    accuracy                           0.58      2845
   macro avg       0.68      0.70      0.58      2845
weighted avg       0.81      0.58      0.59      2845

Model accuracy is:  0.5792618629173989
```

We split the data into 90:10, trained the model. The code is developed using RoIAlign instead of RoIPooling

Detectron2 is built using Pytorch, and provided a very easy API. Detectron2 originates from Mask R-CNN benchmark. Detectron2 helps in Panoptic segmentation (combination of semantic and instance based). It can also be used as a wrapper on top of other projects and can be exported to easily accessible formats. Detectron2 is flexible and fast training on single or multiple GPU servers.
The Faster R-CNN model is trained to predict the bounding box of the pneumonia area with a confidence score as shown below against the ground truth. We used visualizer utility to draw the predictions on the image

GROUND TRUTH



PREDICTION

Since the problem statement requires us to read images, we use CNN and various state of the art deep learning architectures listed here to identify lung opacity with higher recall and accuracy. We have run these architectures on full data set except UNet which was processed on subset of data

4. MODEL EVALUATION – DETERMINING THE SUCCESS OF THE MODELS

Since the problem statement requires us to read images, we use CNN and various state of the art deep learning architectures listed here to identify lung opacity with higher recall and accuracy. We have run these architectures on full data set except U-Net which was processed on subset of data

Architectures	Accuracy	Precision	Recall	AUC-ROC
UNet	66%	38%	81%	
Faster R-CNN	63.4%	37%	91%	
VGG19	81%	55%	93%	91.7%
VGG16	80%	50%	81%	90.7%
ResNet50	77%	55%	80%	83.2%
InceptionNet v3	86%	55%	81%	86.8%

Although from the consolidated results, **VGG19** seems to be the best model with highest recall, we will continue with **Faster R-CNN** since this is an object detection problem

We use classification matrix to check for precision values of each model. Since it's an object detection problem, we finalize Faster R-CNN among all the architectures. We also considered using mean average precision at different Intersection over Union (IoU) thresholds for evaluation. However, due to hardware and Colab version issues, we could not run our results for Mean Average Precision for different values of IoU successfully. Hence, we will basis our model evaluation only on the fact that it's an object detection model and faster R-CNN is the best fit.

5. COMPARISON TO THE BENCHMARK

At the outset, we began with Basic CNN model yielding only about less than 80% recall score. To enhance the score and become more effective in detecting pneumonia, we used augmentation techniques, combined multiple models such as ResNet and U-Net to determine Pneumonia with higher accuracy.

6. VISUALIZATIONS – EXPLORATORY DATA ANALYSIS

From our EDA, we learned that there are 26684 unique patients. Overall, the distribution of data is imbalanced with Target class being only 31.6% of the whole dataset. This tends to result in bias. We have addressed such data issues using augmentation techniques or used sampling methods to equally represent data classes.

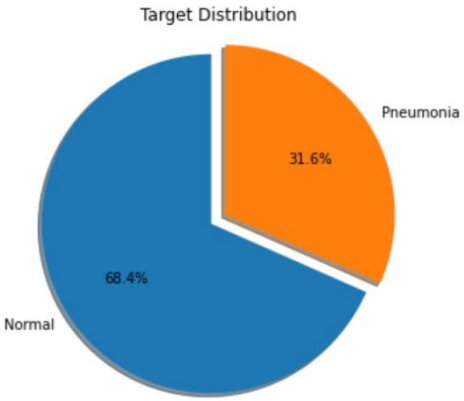
Here, are some of our findings from the Dicom images dataset –

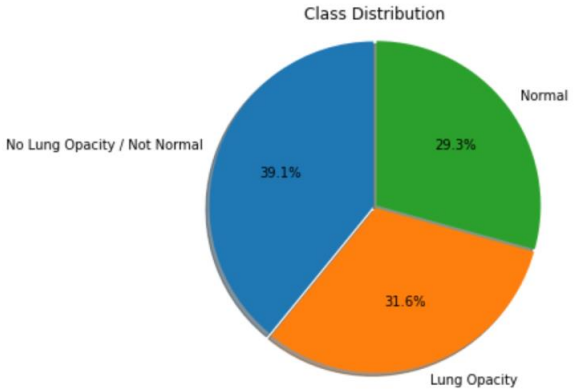
Few patients have multiple bounding boxes -

a. 3266 patients have 2 bounding boxes defined
b. 119 patients have 3 bounding boxes defined
c. 13 patients have 4 bounding boxes defined

We will run a binary classification to predict patients with pneumonia. Target – 1 indicates patients with pneumonia and target – 0 indicates normal patients or patients having lung opacity but not pneumonia

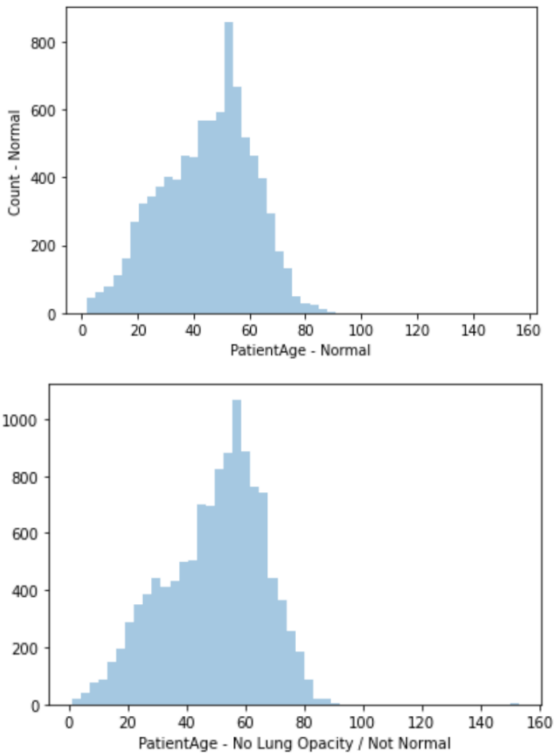
8,851 (29.3%) patients are healthy/Normal
9,555 (31.6%) patients have Lung Opacity
11,821 (39.1%) patients have No Lung Opacity but may have other lung abnormality

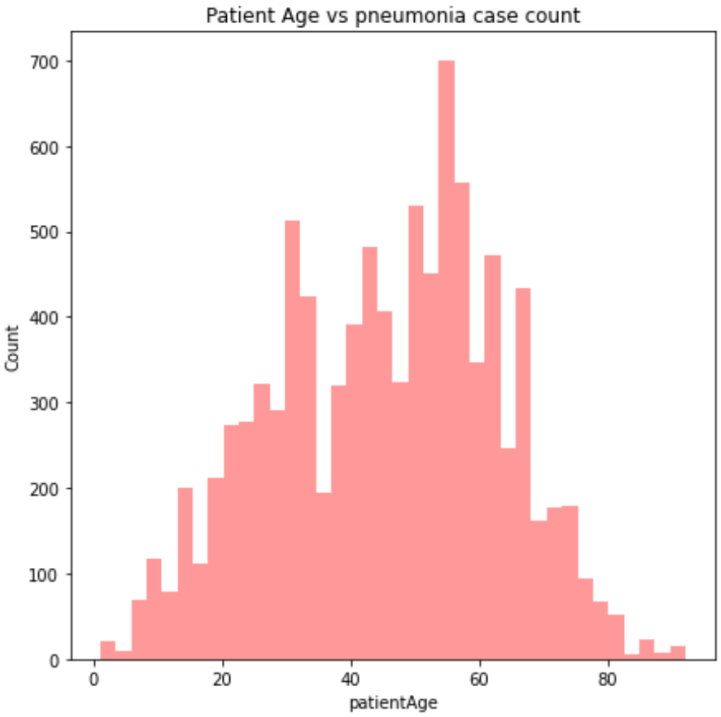


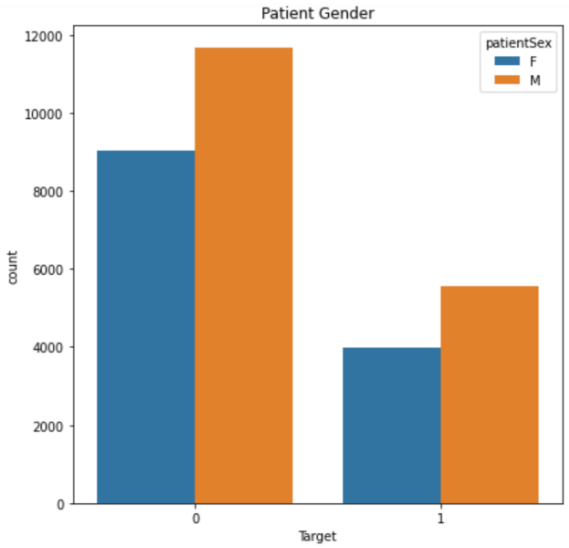


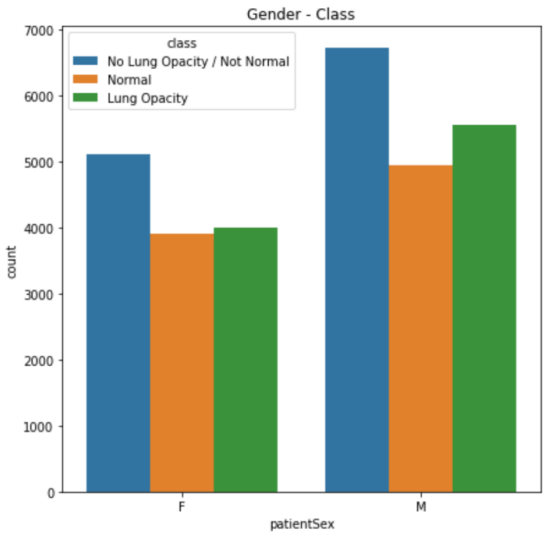
	class	Target	Patient Count
0	Lung Opacity	1	9555
1	No Lung Opacity / Not Normal	0	11821
2	Normal	0	8851

Patient distribution shows most pneumonia patients or patients are found between age 40-60. More records for male is observed in the data. Proportion of pneumonia and normal patients re equal for female patients. However, for males, proportion of pneumonia patients are more than normal

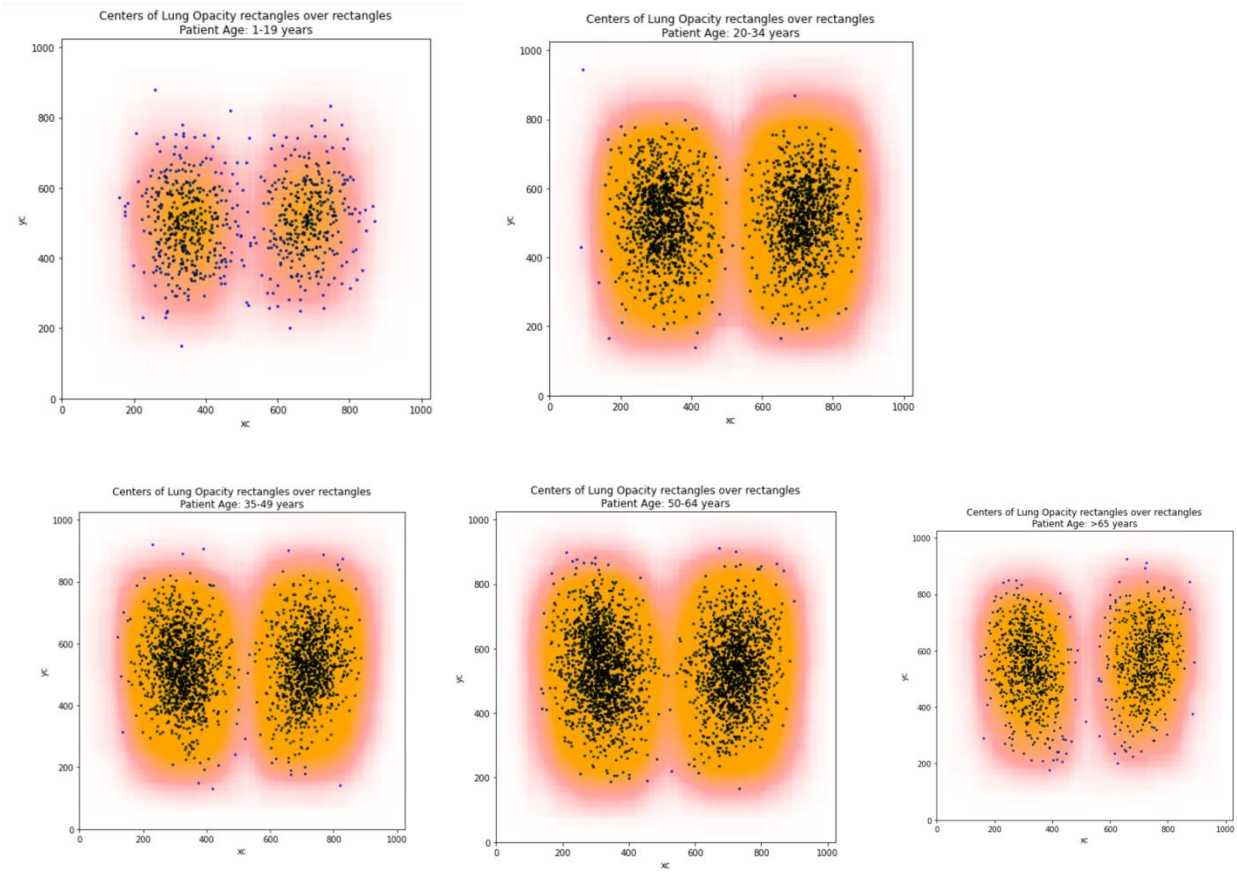




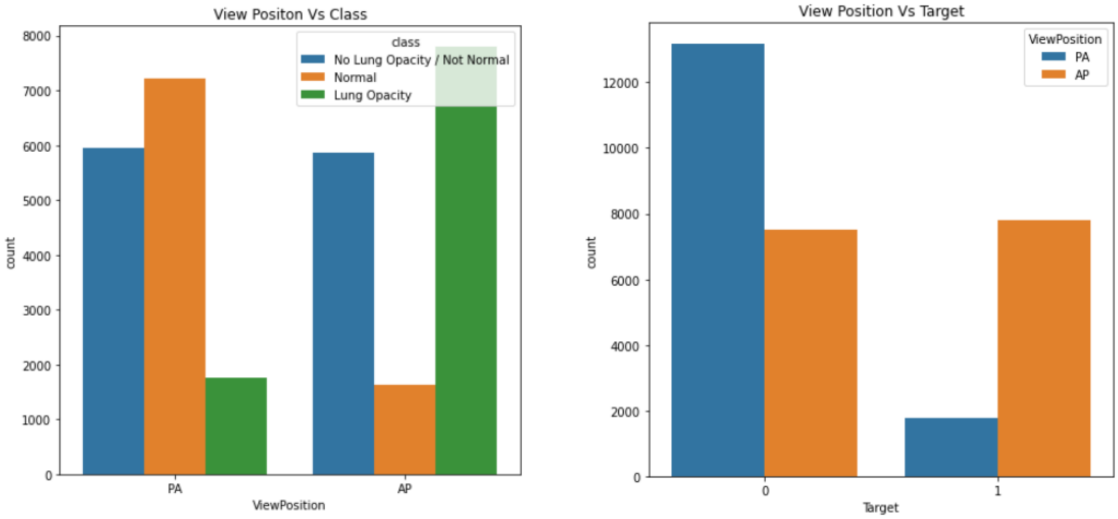


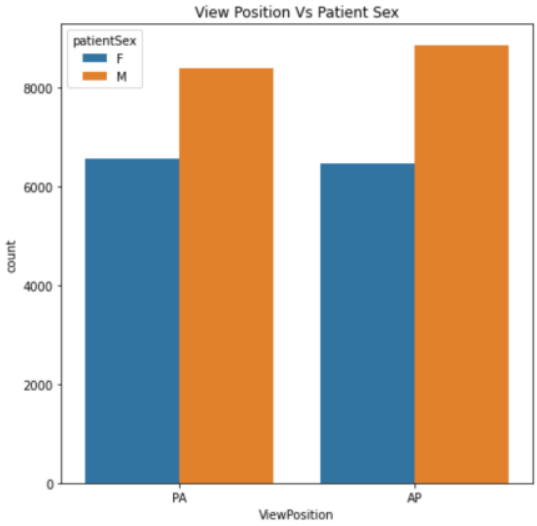


LUNG OPACITY IS SEEN MAXIMUM IN AGE 50-65 FOLLOWED BY AGE GROUP 20-50

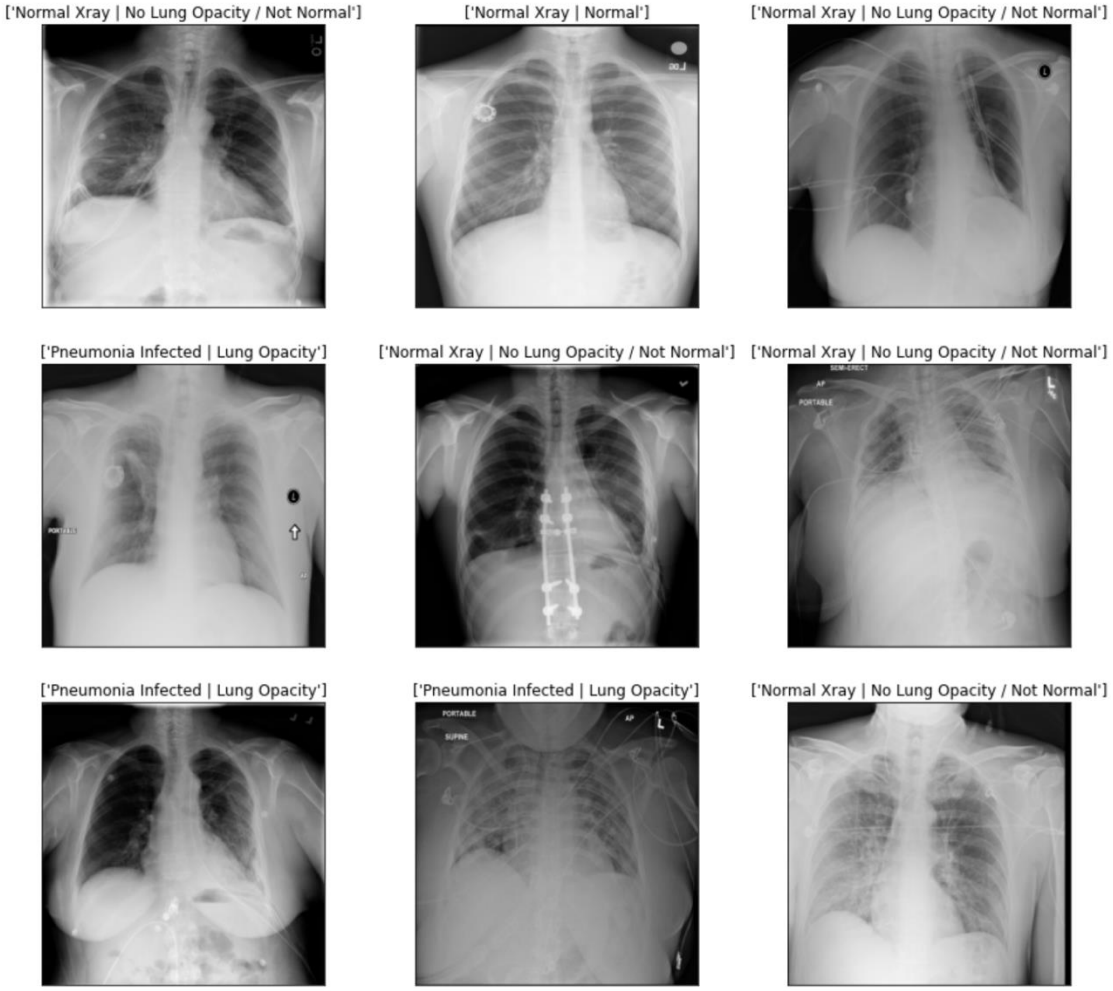


Chest X-ray can be done in **Posterior-Anterior (PA)** projection and **Anterior-Posterior (AP)** Projection. Lung opacity seems to be more visible in AP position

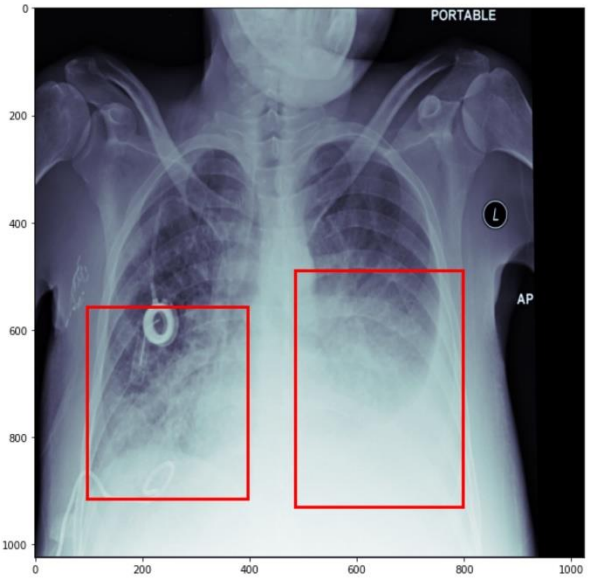




VISUALIZATION OF SAMPLE X-RAY IMAGES FOR EACH CATEGORY

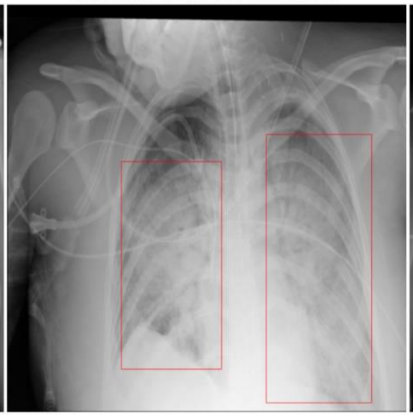


VISUALIZATION OF SAMPLE X-RAY IMAGES WITH BOUNDING BOXES DEPICTING LUNG OPACITY



f76c9846-d88a-4d7c-b887-2759a4333f7c

0d45f420-10d0-41a2-9b87-f5ea81c28599

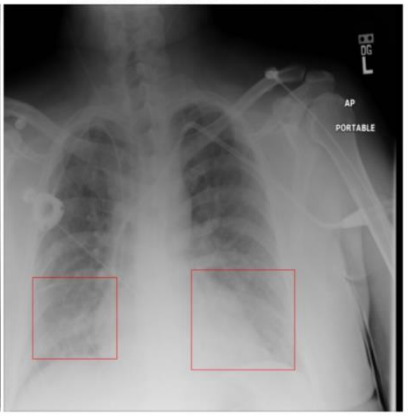


9ab76ba7-f0f0-4dc0-a4e8-57edce55eaf9

2626a2bb-234f-4309-aff4-317f6fac9b06

6c43dfcc-13ea-41d1-8d6d-9b872969140c

bfbbedfe-0ab1-42bd-ac7f-e67b2a8e842a



7. MODEL DEPLOYMENT

The trained model has been packaged in the form of a containerized Python Flask REST API. The web client has also been developed to call the backend service for making the prediction after uploading the test image. Below are the details about the solution:

Rest API: For object detection CPU version of Detectron2 has been used. When the application starts it first loads the base configuration file. It then loads the weights from the trained model. It exposes a POST endpoint which receives the uploaded image file. After making the predictions, *Visualizer* generates image with bounding boxes of the objects detected. The image is then converted to a base64 string and returned in the response JSON payload.

Dockerfile: *python:3.8-slim-buster* is used as the base image. The necessary libraries are then installed along with *pytorch* and *detectron2. Saved* model weight file is also bundled in the package. This can be subsequently hosted elsewhere, so that the new model gets loaded by simply restarting the application.

Web Client: Javascript, Jquery, CSS and html based client which can be deployed on web server such as nginx.

The entire solution has been deployed and tested on local machine and is easily deployable on any cloud platform. We have attached some screenshots from the deployed solution and the software has been uploaded to github along with the other project files.

The User interface takes in an input file and throws the prediction -

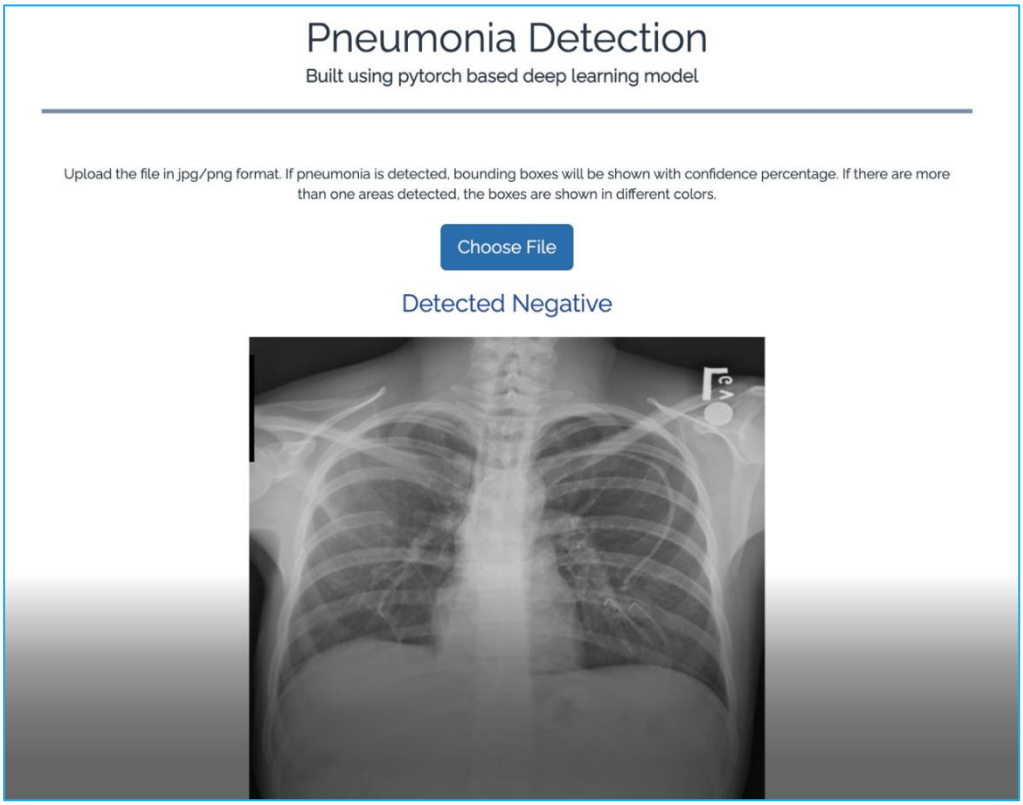


Figure 7: Input detected negative or “No Pnuemonia”

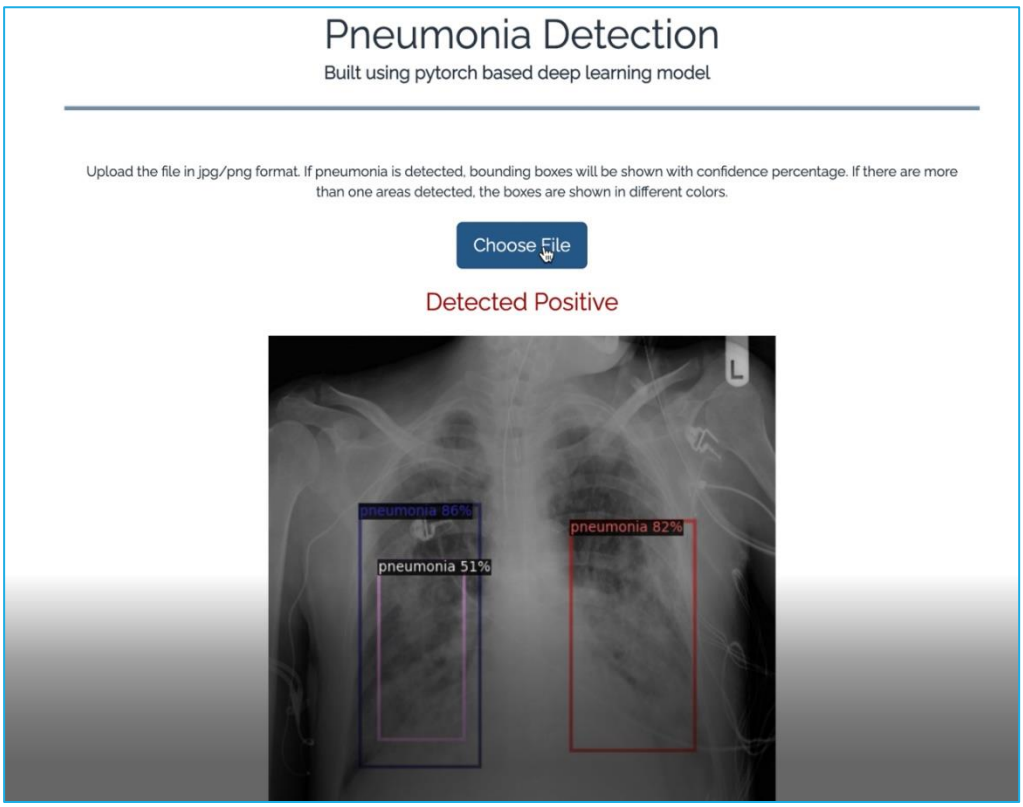


Figure 8: Input detected Positive or “Pneumonia Detected” with bounding boxes depicting confidence rate

8. IMPLICATIONS

The size, shape, and position of pneumonia can vary a great deal. Its target contour is very vague, which leads to great difficulty with detection, and enhancing the accuracy of detection is a major research problem. The main disadvantage of advanced CNN architectures is that it doesn't encode the position and orientation of object. In order to overcome this, the proposed system was compiled using Faster RCNN (Region Convolutional Neural network). Medical testing has high requirements for accuracy, and hence two-stage detectors such as **Faster R-CNN** have an advantage in this respect.

9. LIMITATIONS

There are still problems with the backbone network of the current detection algorithms. For example, ResNet generally has two problems: a large network depth leading to long training time and massive down-sampling that leads to the target position and semantic information being lost

The dataset contains three categories of subjects, normal, pneumonia, and abnormal(cancer or other diseases) but only provides the bounding box for pneumonia images. However, the features of pneumonia and abnormal (cancer or other diseases) are pretty similar, which caused the failure to distinguish pneumonia and abnormal images for Faster R-CNN. This results in predicting bounding box for abnormal images.

One of the gaps with the object detection API is that it is unable to properly process negative examples. In general, it's not necessary to explicitly include "negative images". However, in this case since a big portion of the data set is a negative examples that are different from positive examples, Tensorflow is not able to train properly. The loss function doesn't penalize for detections on negative images. As a result, the model has many false positives where the not normal lung but no Pneumonia case is annotated as Pneumonia.

Another limitation was different distribution of train and test datasets, most likely due to different labeling methodology.

Lastly, we ran into hardware limitations as well. Running Dicom images dataset on Colab environment (even with Google Pro) takes very long time to process and execute leading to inefficiency in time.

10. CLOSING REFLECTIONS

We had class imbalance for pneumonia cases in our dataset. Hence, using NIH dataset since It's a bigger dataset but with lower quality of labels, would be very interesting to check if training the model to predict both datasets would improve the result, or at least use it to pretrain the base model. In future, we will include metrics such as MAP (Mean Average

Precision) at different thresholds of IoU to evaluate the performance of object detectors. Also, a post processing technique that has worked well with some state-of-the-art new models is shrinking the predicted boxes a bit. This works because different radiologists have different opinions on the extent of Pneumonia. So overall the model predicted bounding boxes will be somewhat on the larger side and reducing them can result in a better score. We can also explore CheXNet, which is an algorithm that can detect pneumonia from chest X-rays at a level exceeding practicing radiologist. CheXNet, is a 121-layer convolutional
neural network trained on ChestX-ray14, currently the largest publicly available chest Xray dataset, containing over 100,000 frontal view X-ray images with 14 diseases and claims to exceed average radiologist performance on the F1 metric.

CODE BASE

This project report is supported by findings from the following code base

Model/ Analysis	HTML or ipynb
EDA, VGG19, VGG16, ResNet, InceptionNet v3	Pneumonia_EDA_VGG19_VGG16_ResNet50_InceptionNet_Complete.html
UNET	Capstone_Custom_UNET.ipynb
Faster RCNN using Detectron2	Pnuemonia_pytorch_complete.html

REFERENCES

1. VGG19 Architecture – Figure2
   https://medium.com/geekculture/style-transfer-using-vgg19-test-version-aa48bf9cd47e

2. ResNet 50 Architecture – Figure3
   https://www.researchgate.net/figure/ResNet-50-architecture-26-shown-with-the-residual-units-the-size-of-the-filters-and_fig1_338603223

3. InceptionV3 Architecture –Figure4
   https://www.researchgate.net/figure/Block-diagram-of-Inception-v3-improved-deep-architecture_fig3_341563435

4. U-NET Architecture – Figure5
   https://www.researchgate.net/figure/The-architecture-of-Unet_fig2_334287825

5. Faster R-CNN Architecture – Figure6
   https://www.researchgate.net/figure/The-architecture-of-Faster-R-CNN_fig2_324903264