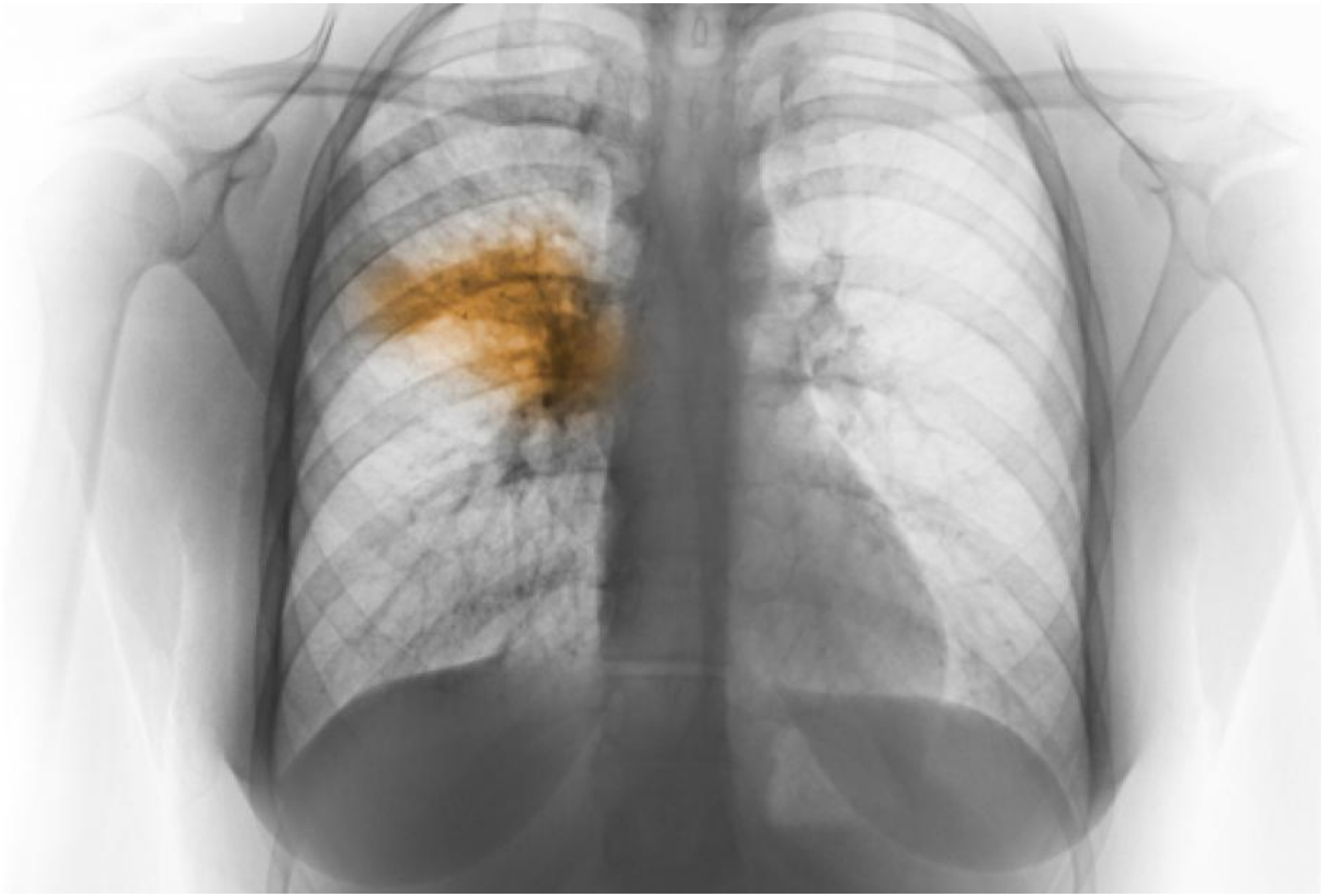
Capstone

presentation



OVERVIEW OF PNEUMONIA

The single most significant bacterial cause of death in children worldwide is pneumonia. In 2017, pneumonia killed 808,694 children under the age of 5, accounting for 15 percent of all deaths by children under five. Children and families worldwide are afflicted by pneumonia, but it is most common in South Asia and sub-Saharan Africa. It can be avoided with easy procedures and managed with low-cost, low-tech treatment and care.

PROBLEM STATEMENT

In this project, we analyze data with the knowledge of EDA. We build a detection model and present our findings based on the evaluations with the RSNA Pneumonia Detection Challenge dataset.

DATA DESCRIPTION

RSNA Pneumonia dataset consists of 29684 thousand images. All the images are in Dicom format. There are 3000 images for testing and the remaining for training.

Dicom images: The images are in a particular format called DICOM files (*. dcm). They contain a mix of header metadata as well as pixel data underlying raw image arrays.

There are three classes in the dataset - Normal, Not normal/No opacity, and Lung opacity.

- **Normal class** indicates there is no anomaly in the lungs.
- **Not normal/No opacity** demonstrates to those who do not have pneumonia, but the image still has some abnormality. Sometimes, this finding could mimic the appearance of the right pneumonia.
- **Lung opacity** class indicates there is definite pneumonia in the lungs.

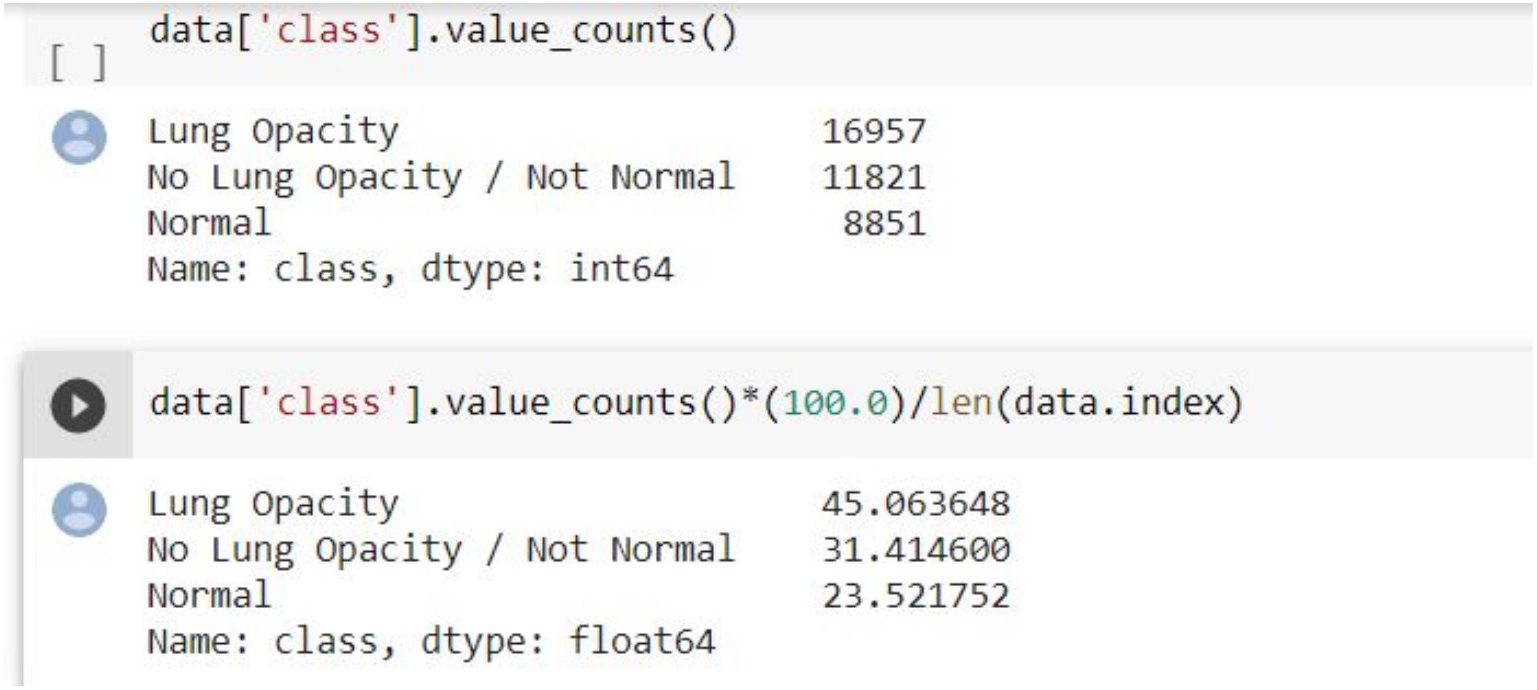
The train labels file consists of the bounding box coordinates belonging to each image. Bounding box coordinates are given in the following format:

- x -- the upper-left x coordinate of the bounding box.
- y -- the upper-left y coordinate of the bounding box.
- width -- the width of the bounding box.
- height -- the height of the bounding box.

With these bounding box coordinates, the target column is provided, which discriminates classes into categories of 0 and 1.

EDA AND PREPROCESSING

CLASSES



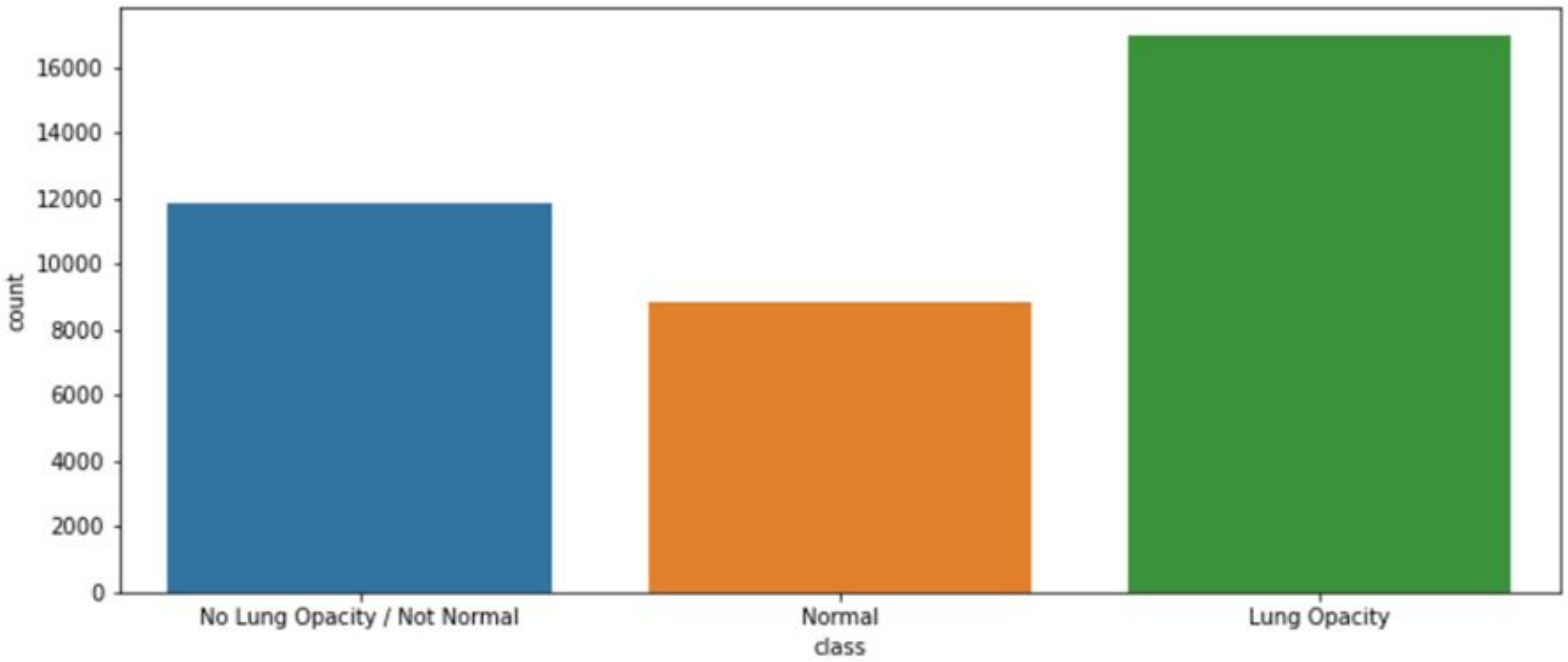
```
[ ]  data['class'].value_counts()
```

```
Lung Opacity                      16957
No Lung Opacity / Not Normal      11821
Normal                             8851
Name: class, dtype: int64
```

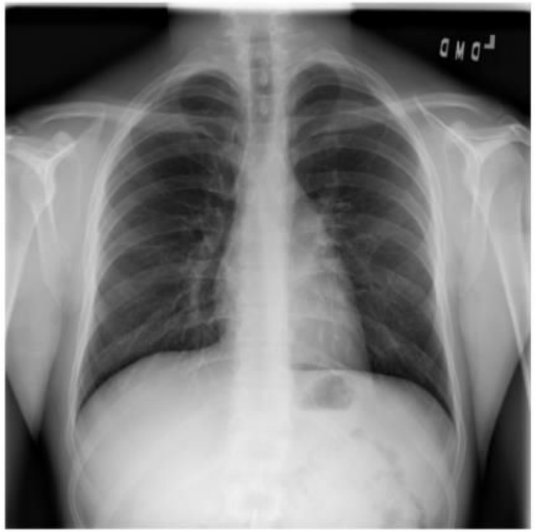
```
data['class'].value_counts()*(100.0)/len(data.index)
```

```
Lung Opacity                      45.063648
No Lung Opacity / Not Normal      31.414600
Normal                            23.521752
Name: class, dtype: float64
```

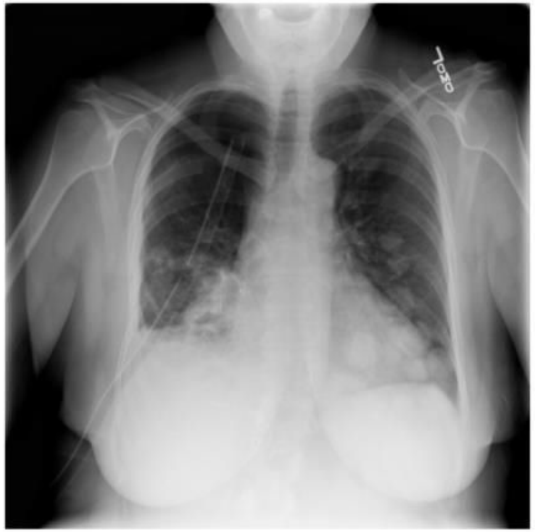
```
fig=plt.figure(figsize=(12, 5))
countplot = sns.countplot(data['class'])
```



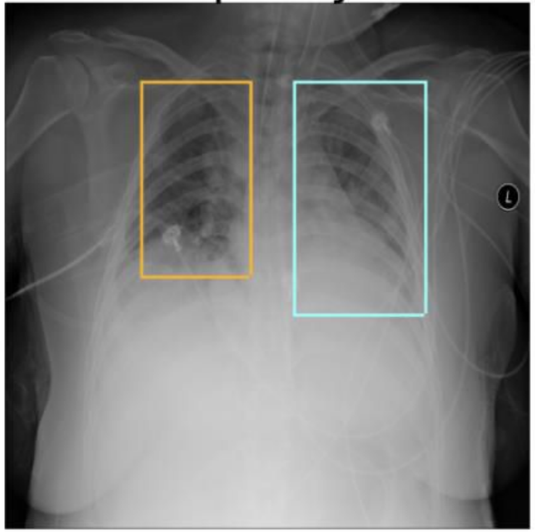
Normal



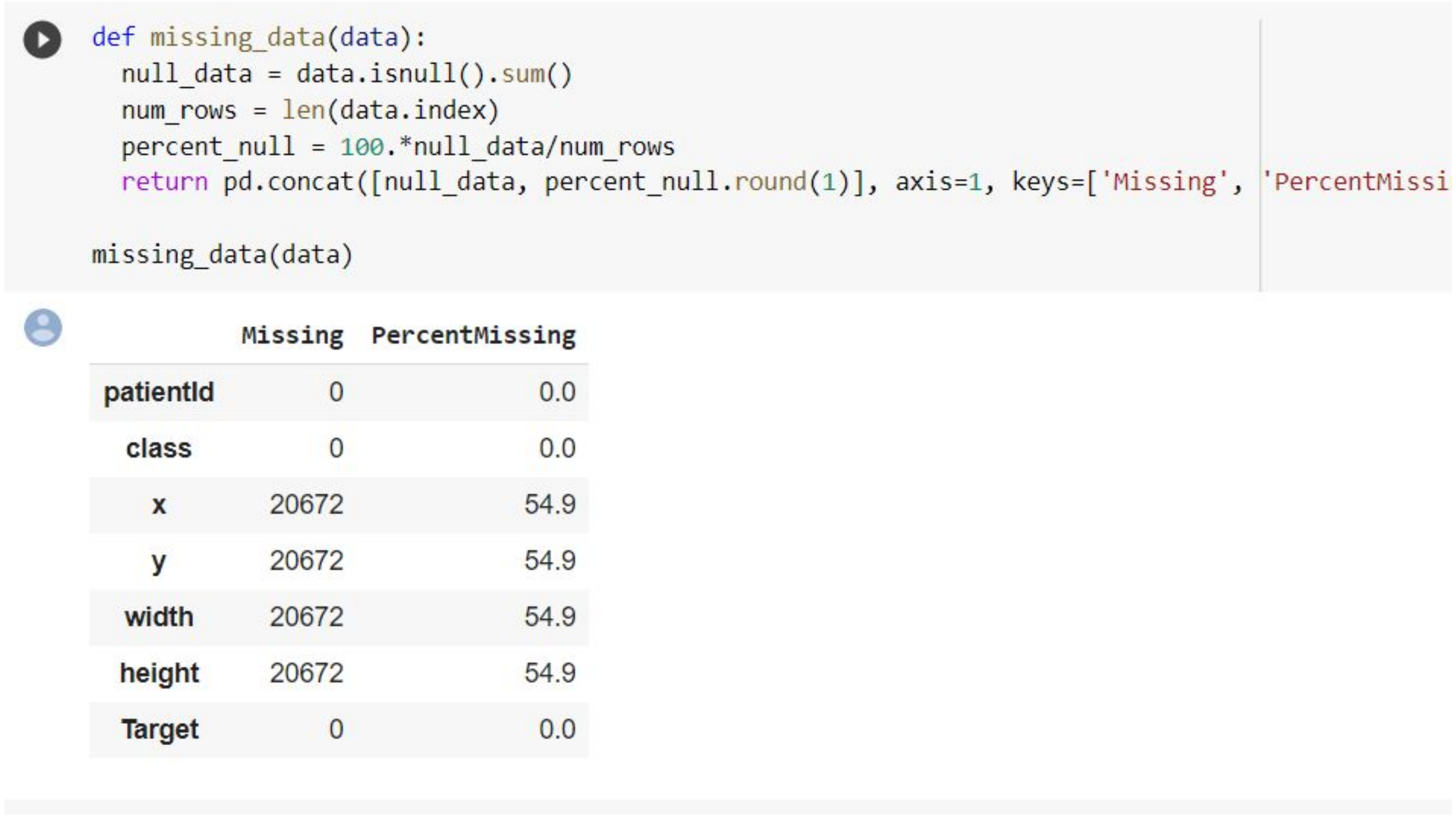
Not Normal



Opacity



MISSING VALUES



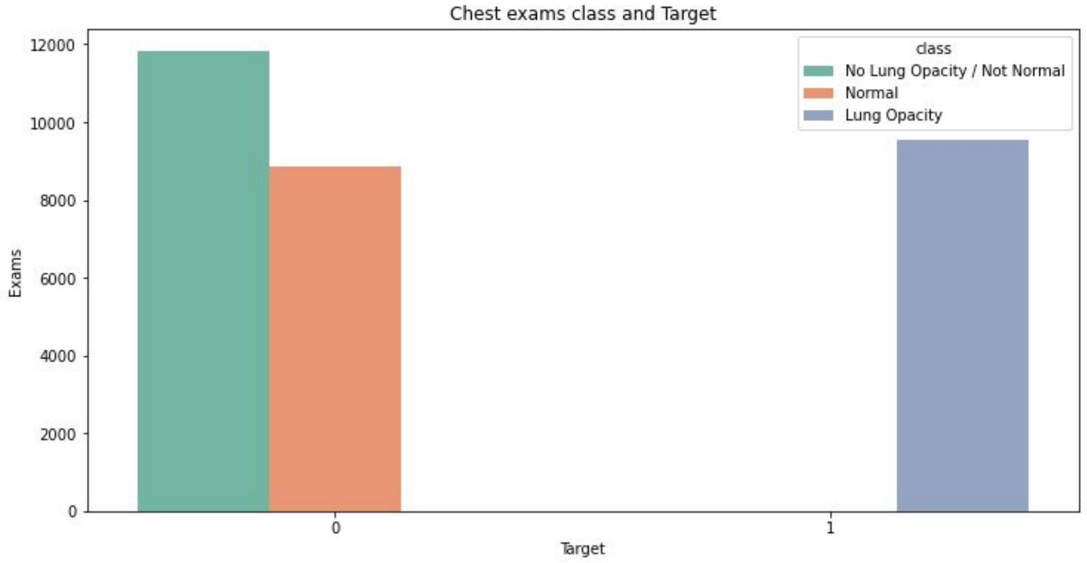
```
def missing_data(data):
  null_data = data.isnull().sum()
  num_rows = len(data.index)
  percent_null = 100.*null_data/num_rows
  return pd.concat([null_data, percent_null.round(1)], axis=1, keys=['Missing', 'PercentMissi

missing_data(data)
```

	Missing	PercentMissing
patientId	0	0.0
class	0	0.0
x	20672	54.9
y	20672	54.9
width	20672	54.9
height	20672	54.9
Target	0	0.0

TARGET

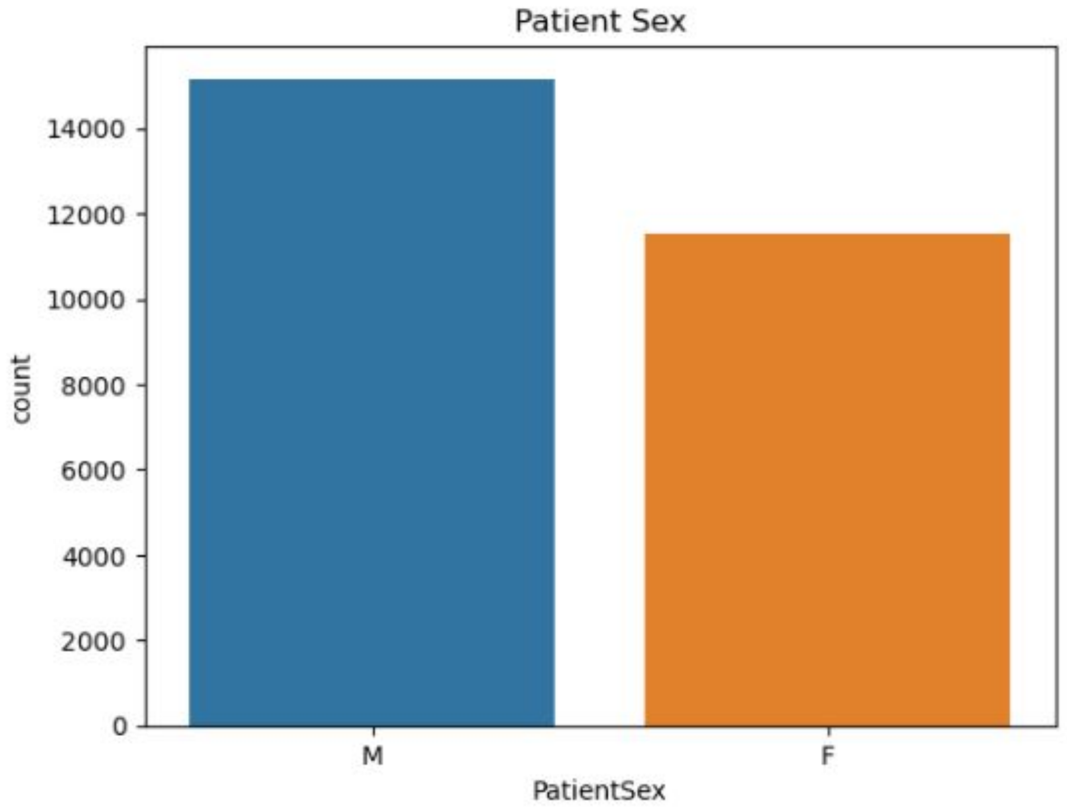
In [12]:
```
fig, ax = plt.subplots(nrows=1,figsize=(12,6))
tmp = comb_bounding_box_df.groupby('Target')['class'].value_counts()
df = pd.DataFrame(data={'Exams': tmp.values}, index=tmp.index).reset_index()
sns.barplot(ax=ax,x = 'Target', y='Exams',hue='class',data=df, palette='Set2')
plt.title("Chest exams class and Target")
plt.show()
```



META DATA

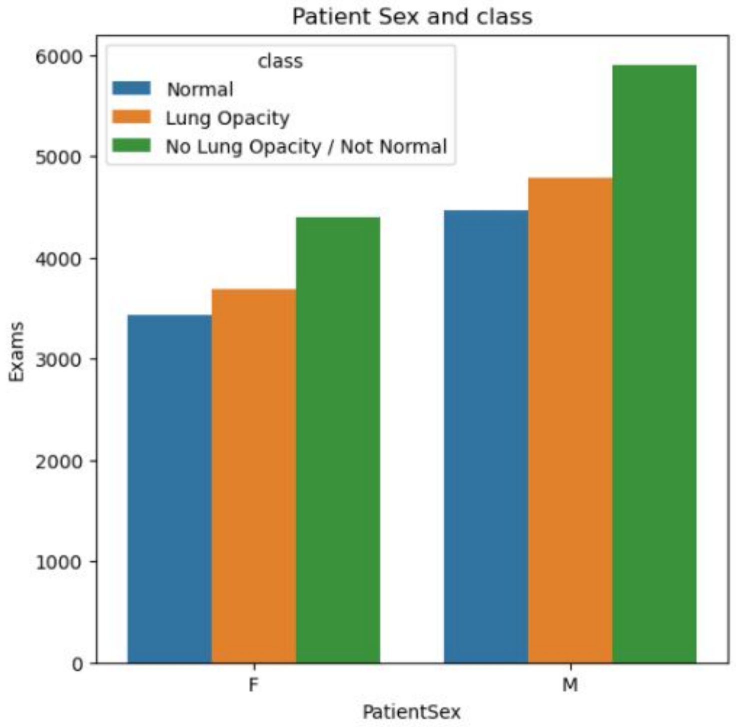
[56]:

```
sns.countplot(image_meta_df['PatientSex'])
plt.title("Patient Sex")
plt.show()
```



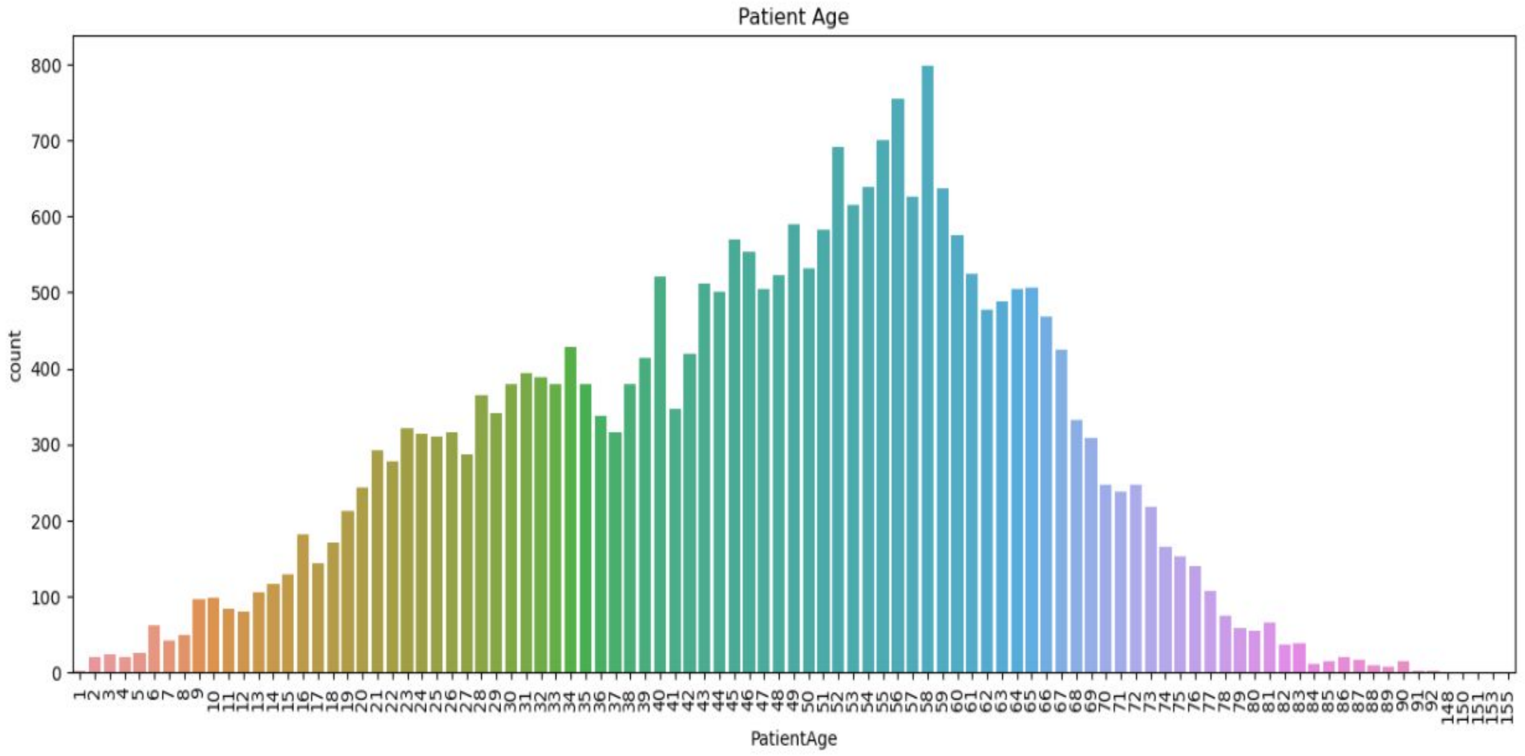
[55]:

```
tmp = image_meta_df.groupby(['class', 'PatientSex'])['patientId'].count()
df1 = pd.DataFrame(data={'Exams': tmp.values}, index=tmp.index).reset_index()
tmp = df1.groupby(['Exams','class', 'PatientSex']).count()
df3 = pd.DataFrame(data=tmp.values, index=tmp.index).reset_index()
fig, (ax) = plt.subplots(nrows=1,figsize=(6,6))
sns.barplot(ax=ax, x = 'PatientSex', y='Exams', hue='class',data=df3)
plt.title("Patient Sex and class")
plt.show()
```



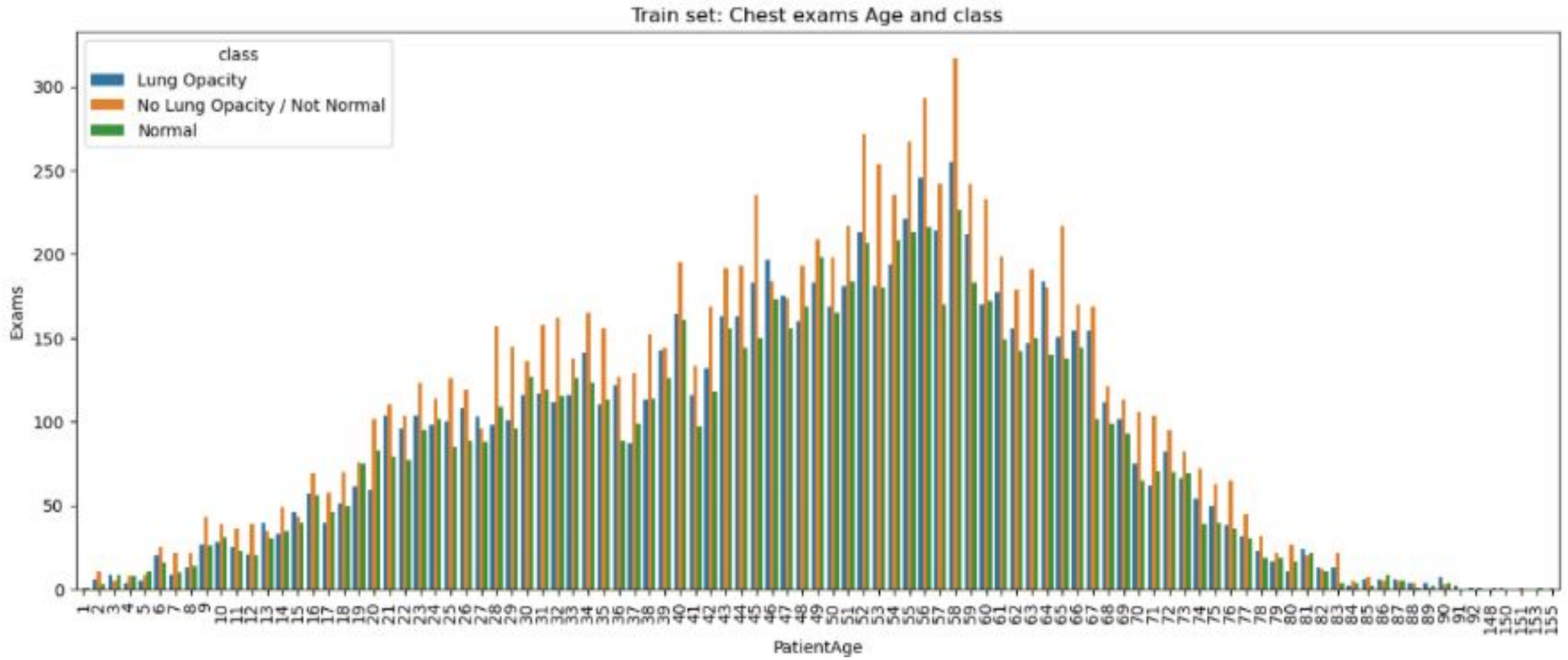
[54]:

```
fig, (ax) = plt.subplots(nrows=1,figsize=(16,6))
sns.countplot(image_meta_df['PatientAge'], ax=ax)
plt.title("Patient Age")
plt.xticks(rotation=90)
plt.show()
```



```python
tmp = image_meta_df.groupby(['class', 'PatientAge'])['patientId'].count()
df1 = pd.DataFrame(data={'Exams': tmp.values}, index=tmp.index).reset_index()
tmp = df1.groupby(['Exams','class', 'PatientAge']).count()
df3 = pd.DataFrame(data=tmp.values, index=tmp.index).reset_index()


fig, (ax) = plt.subplots(nrows=1,figsize=(16,6))
sns.barplot(ax=ax, x = 'PatientAge', y='Exams', hue='class',data=df3)
plt.title("Train set: Chest exams Age and class")
plt.xticks(rotation=90)
plt.show()
```

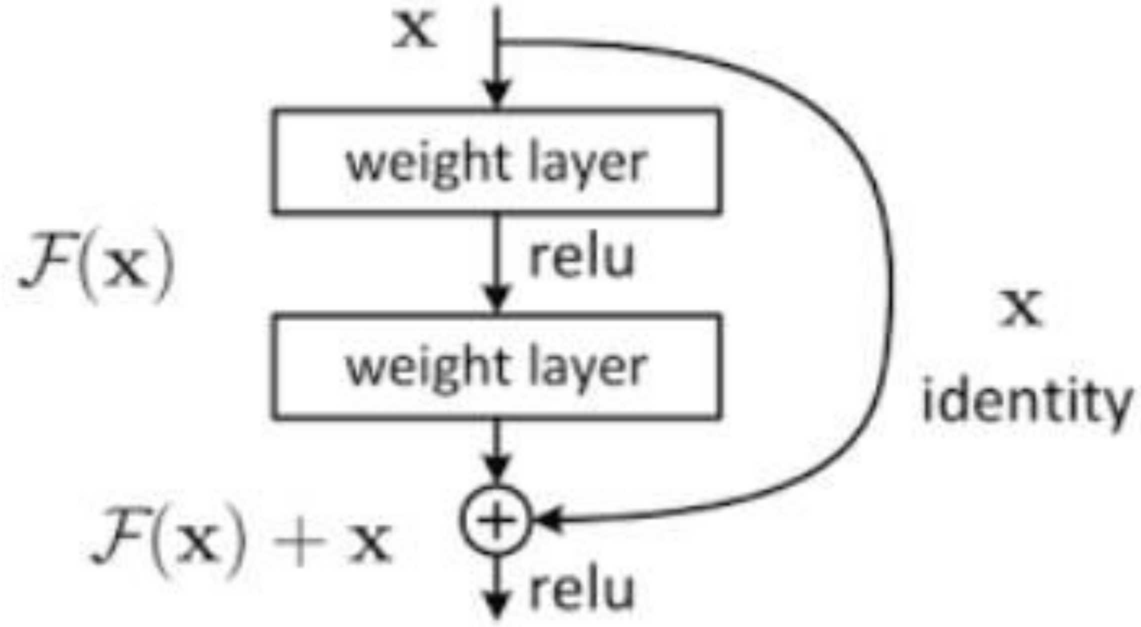


MODEL BUILDING

CNN CUSTOM MODEL WITH RESNET BLOCKS

NETWORK AND ARCITECTURE

The network consists of several residual blocks with convolutions and downsampling blocks with max pooling. At the end of the network, a single upsampling layer converts the output to the same shape as the input.
As the input to the network is 256 by 256 (instead of the original 1024 by 1024) and the network downsamples several times without any meaningful upsampling (the final upsampling is to match in 256 by 256 mask), the final prediction is very crude. If the network down samples four times, the final bounding boxes can only change with at least 16 pixels.


$\mathbf{x}$
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
$\mathbf{x}$
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

U-NET WITH MOBILENET BACKBONE

NETWORK AND ARCHITECTURE

This network consists of u-shaped architecture, as the name suggests. During the downsampling, features are generated gradually. Whereas on upsampling, the features are duplicated to that of the original image with mask segmentation.
This network takes an image input of 224*224 pixels with three channels red, green, and blue. It has around 3 million parameters. A standard convolution network consisting of repeated use of convolutions, each followed by a rectified linear unit (ReLU) and a max-pooling process, is the contracting part. The spatial information is decreased during the contraction, while feature information is increased. Via a series of up-convolutions and concatenations with high-resolution characteristics from the contracting path, the expansive path incorporates the function and spatial details.

DENSENET 121

NETWORK AND ARCHITECTURE

There are four Dense Blocks with varying layer numbers in each architecture. For instance, in the four dense blocks, DenseNet-121 has [6,12,24,16] layers. We can see that a 7x7 stage 2 Conv Layer followed by a 3x3 stride-2 MaxPooling layer consists of the first part of the DenseNet architecture. A Classification Layer that accepts the feature maps of all network layers to perform the classification follows the fourth dense block.
Downsampling and upsampling are added to the dense layers to learn the mask implementation of the given input images of 224*224 with three channels (RGB). This network has 6million parameters.

Mask RCNN

NETWORK AND ARCHITECTURE

The first step is to install the library. Installation involves cloning the GitHub repository and running the installation script on the workstation. Coco dataset pretrained weights are loaded to use for transfer learning.

Mask RCNN architecture consists of two stages,

Stage 1: The first stage consists of two networks, a backbone network (ResNet, VGG, Inception, etc.) and a network of regional proposals. To offer a set of region proposals, these networks run once per picture. Proposals for a region are regions in the function map that contain the object.

Stage 2: In the second stage, the network predicts bounding boxes and object class for each of the proposed region obtained in stage1. Each proposed region can be of different size, whereas fully connected layers in the networks always require a fixed size vector to make predictions. These proposed regions' size is fixed by using either RoI pool (which is very similar to MaxPooling) or the RoIAlign method. The RoIAlign layer's output is then fed into Mask head, which consists of two convolution layers. It generates a mask for each RoI, thus segmenting an image in a pixel-to-pixel manner.

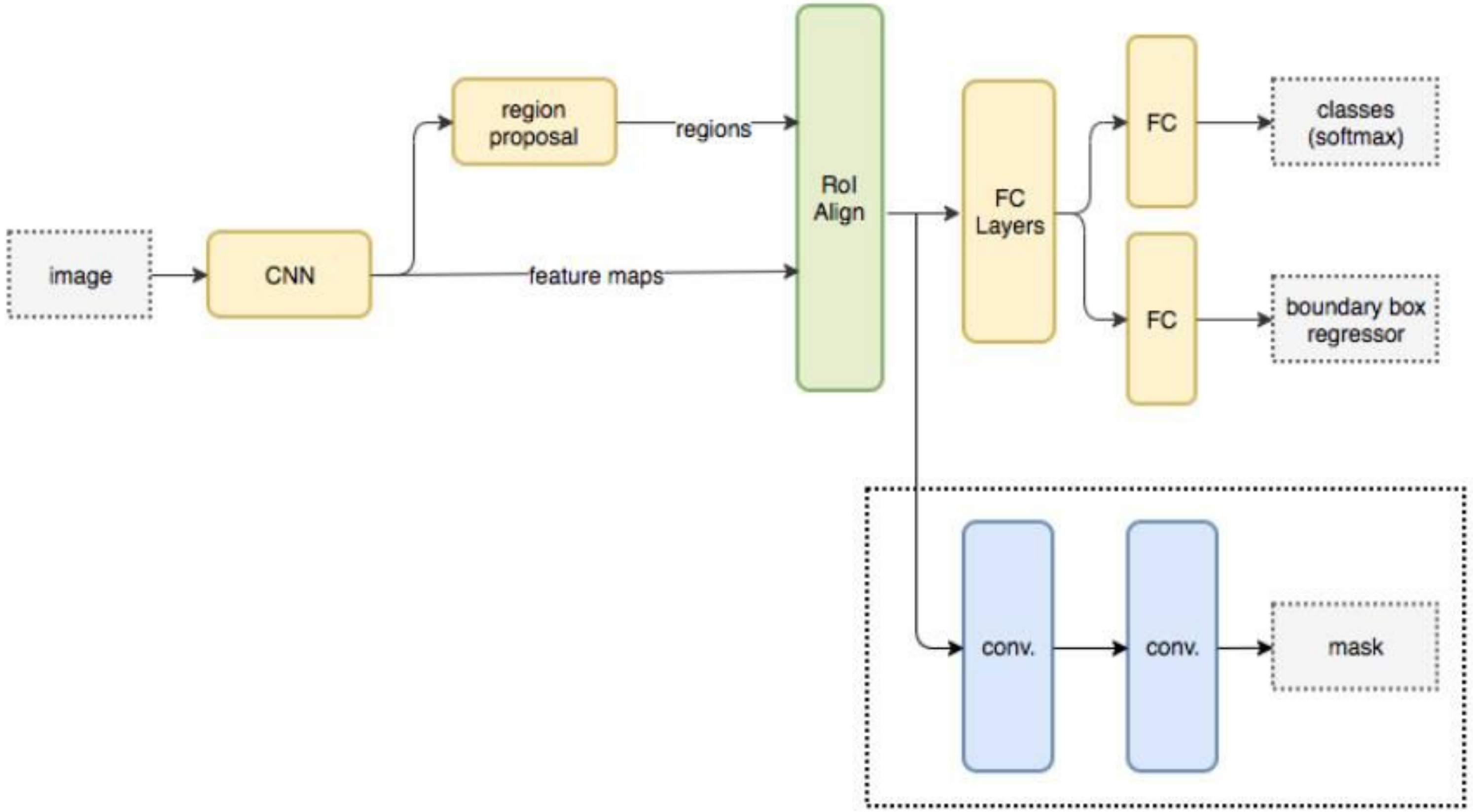
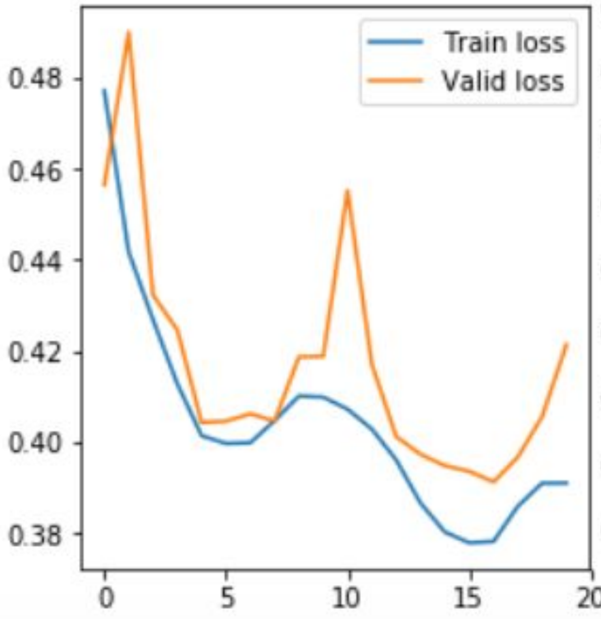


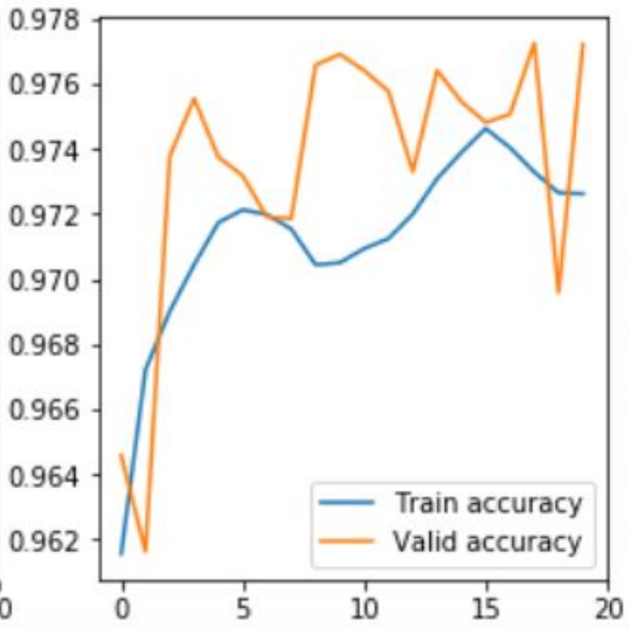
image
CNN
region proposal
regions
feature maps
RoI Align
FC Layers
FC
FC
classes (softmax)
boundary box regressor
conv.
conv.
mask

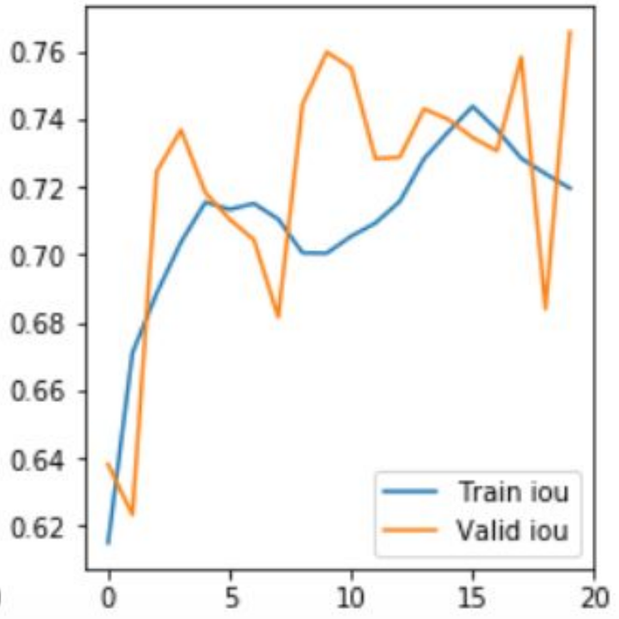
Mask

EVALUATION

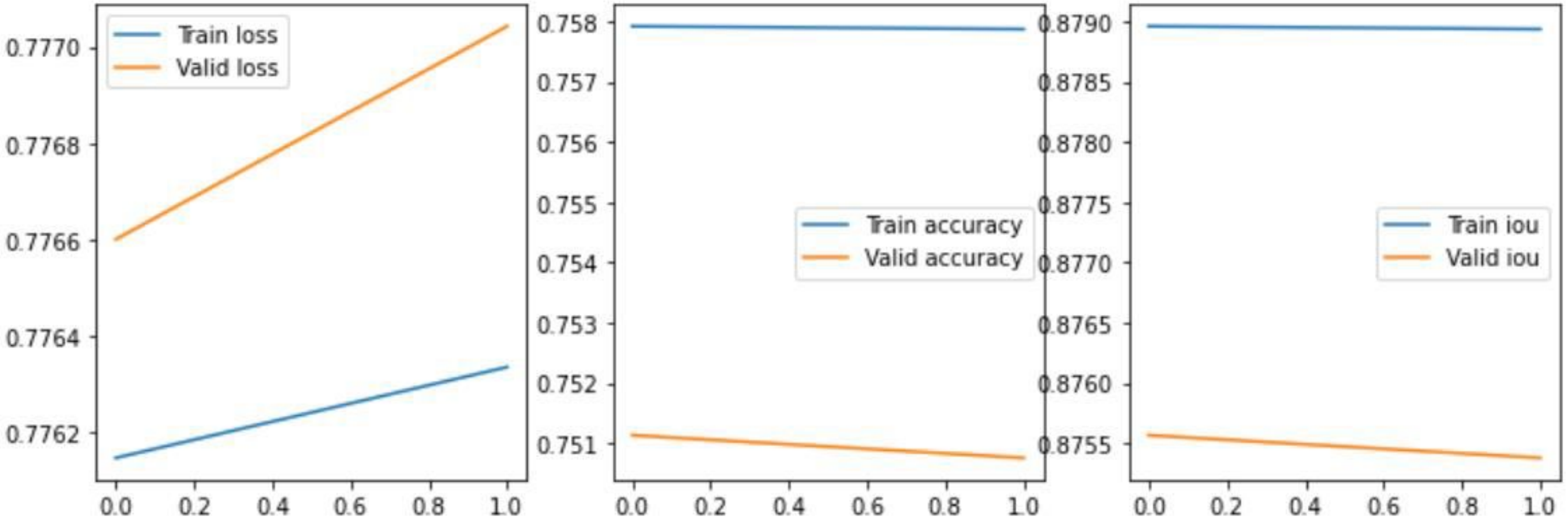
CNN CUSTOM MODEL WITH RESNET BLOCKS



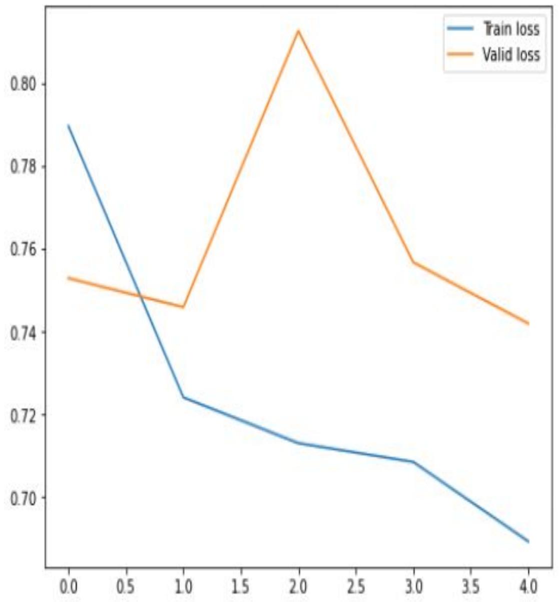


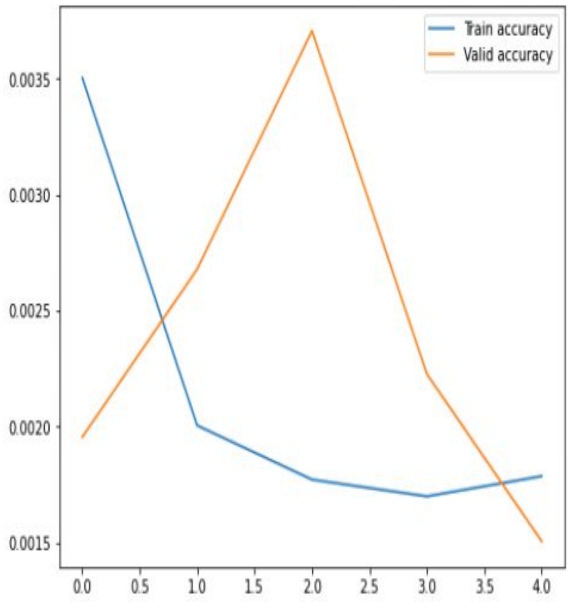


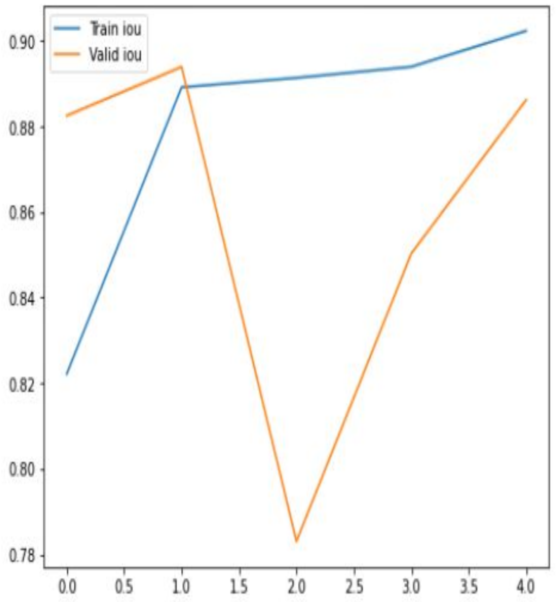
DENSENET 121



U-NET WITH MOBILENET BACKBONE







MODEL EVALUATION

The performance of a model for an object recognition task is often evaluated using the mAP and IOU.
We are predicting bounding boxes so we can determine how well the predicted and actual bounding boxes overlap. This can be calculated by dividing the area of the overlap by the total area of both bounding boxes, or the intersection divided by the union, referred to as “intersection over union,” or IoU. A perfect bounding box prediction will have an IoU of 1. It is standard to assume a positive prediction of a bounding box if the IoU is greater than 0.5, e.g. they overlap by 50% or more.

CNN MODEL WITH RESNET BLOCKS

Vs

MASK-RCNN

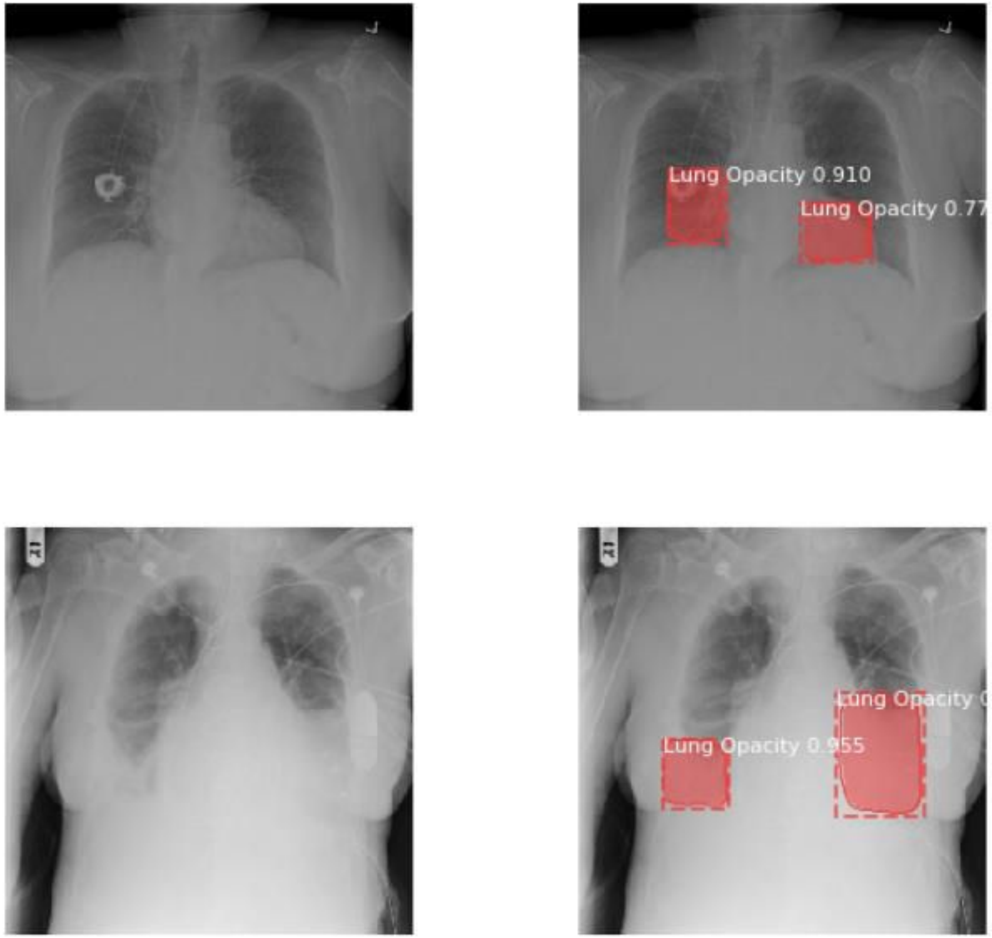
MASK RCNN HYPERPARAMETERS

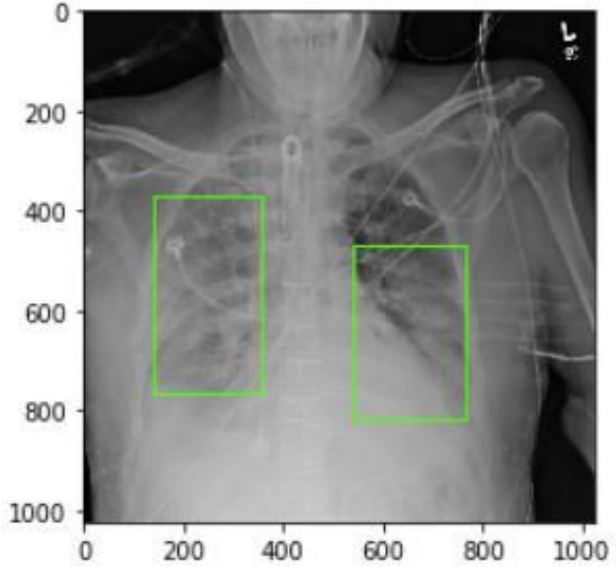
1) Set Higher Learning rate to train heads faster with 2 epochs.
2) Excluded last layers from training and trained on COCO datasets.
3) Original Image size is 1024x1024 with 3 channels which was scaled down to 256x256x3 to feed in the neural network.
4) This model consists of Time Distributed convolutional, Batch normalization layers and ResNet as a backbone architecture.
5) Ran the model in Inference mode setting GPU_COUNT=1 and IMAGES_PER_GPU=1.
6) Image Augmentation technique used to sharpen, blur and scale images.

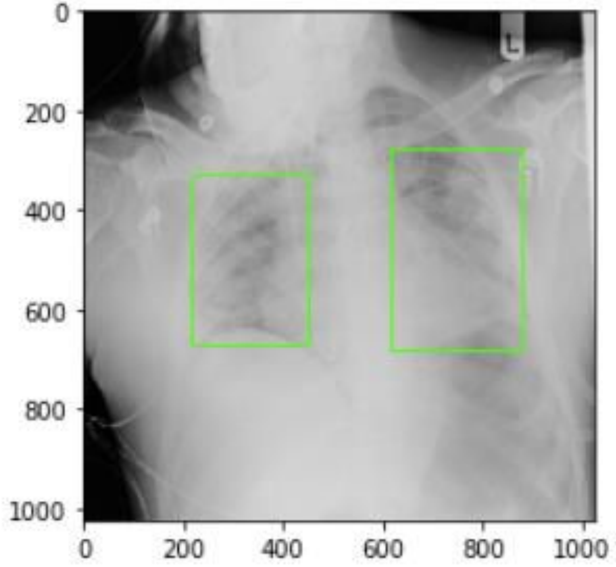
CNN RESNET CUSTOM MODEL HYPERPARAMETERS

1) Created a Data generator that takes input size, filenames and batch size.
2) Custom model with ResNet as a backbone architecture setting number of channels, image size, number of blocks and depth of a network.
3) Original Image size is 1024x1024 with 3 channels which is transformed into 256x256x3 to feed in the network for training.
4) Optimizers used are Adam, RMSProp and IOU as a metric to calculate bounding boxes on images.
5) callbacks used in Model hyperparameters for early stopping and reduce learning rate on plateau

VISUALIZATIONS OF RESULTS


Lung Opacity 0.910
Lung Opacity 0.77
Lung Opacity
Lung Opacity 0.955


0
200
400
600
800
1000
0
200
400
600
800
1000


0
200
400
600
800
1000
0
200
400
600
800
1000

BENCHMARK

At the beginning of the model building process we set out to achieve the IOU of 0.75 or more with the limited hardware and system resources available to run the model. From the above section it is evident that we achieved that benchmark with less epochs. But with the better resources and processors our benchmark would have been better. We were able to achieve the benchmark by trying different hyperparameters through one at a time method (OFAT). After
reaching the benchmark we continued on and found out that computation cost is increasing which is not feasible with the system resources and GPU time.

LIMITATIONS

• As our model has huge number of parameters with large number of layers of Resnet as a backbone it is tend to take longer to train.
• Bounding boxes can be improved to the exact size of the infected area. So, the limitation is its very difficult to do the segmentation as the bounding boxes will include certain portion of uninfected area.
• The implemented model RPN_Class Loss and RPN_BBox Loss has significant improvement per epoch, but Total Loss can be improved by changing some hyperparameters.
• This model is computationally expensive. Which can be a huge drawback as the initial cost of implementation maybe higher.

CONCLUSION AND CLOSING REFLECTIONS

THANK YOU