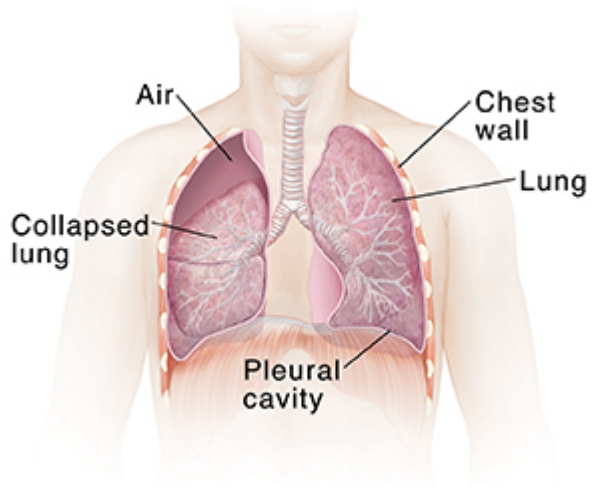


SIIM-ACR Pneumothorax Segmentation

1. Business Problem

1.1 Description

A pneumothorax or collapsed lungs is a medical condition that is responsible for making the people suddenly gasp for air, and feel helplessly breathless for no apparent reason. It can be a complete lung collapse or a collapse of a portion of the lung. It is usually diagnosed by a radiologist with several years of experience on a chest x- ray; which sometimes is very difficult to confirm. Our goal is to classify(if present segment) pneumothorax from a set of chest radiographic images.

This is a Kaggle problem. (https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/overview)

1.2 Problem Statement

Classify pneumothorax from a set of chest radiographic images and if present segment the regions of pneumnothorax.

1.3 Sources

- Source : https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/overview
-

1.4 Real world/Business Objectives and Constraints

- Classify pneumothorax and if present segment it.
- Maximize the overlap between the actual mask and predicted mask(Dice).
- No such latency concerns.

2. Machine Learning Problem

2.1 Data

2.1.1 Data Overview

- Data is present in under folder SIIM. It contains three folders and files,

- dicom-images-test
- dicom-images-train
- train-rle.csv

Note : I downloaded the data from https://www.kaggle.com/seesee/siim-train-test as the data is removed from the Cloud Healthcare API.

2.2 Mapping the real world problem to a Deep Learning Problem

2.2.1 Type of Deep Learning Problem

It is a basically a semantic image segmentation problem, where we have to segment areas of pneumothorax.

2.2.2 Performance Metric

Source: https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/data

Metric:

- Focal Loss + Dice Loss
    - Focal loss is very good for imbalanced data as it focuses more on hard examples than easy examples. Our data is very imbalanced as the area having pneumothorax is very small. Dice is best metric overall. So we are going for the combination of the two.

2.2.3 Necessary Imports

```
import numpy as np

def mask2rle(img, width, height):
    rle = []
    lastColor = 0;
    currentPixel = 0;
    runStart = -1;
    runLength = 0;

    for x in range(width):
        for y in range(height):
            currentColor = img[x][y]
            if currentColor != lastColor:
                if currentColor == 255:
                    runStart = currentPixel;
                    runLength = 1;
                else:
                    rle.append(str(runStart));
                    rle.append(str(runLength));
                    runStart = -1;
                    runLength = 0;
                    currentPixel = 0;
            elif runStart > -1:
                runLength += 1
            lastColor = currentColor;
            currentPixel+=1;

    return " ".join(rle)

def rle2mask(rle, width, height):
    mask= np.zeros(width* height)
    array = np.asarray([int(x) for x in rle.split()])
    starts = array[0::2]
    lengths = array[1::2]

    current_position = 0
```

```
    for index, start in enumerate(starts):
        current_position += start
        mask[current_position:current_position+lengths[index]] = 255
        current_position += lengths[index]

    return mask.reshape(width, height)
```

```
!pip install pydicom
```

```
Requirement already satisfied: pydicom in /usr/local/lib/python3.7/dist-packages (2.1.2)
```

```
import pydicom
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
import glob
from tqdm import tqdm
import seaborn as sns
```

```
import warnings
warnings.filterwarnings("ignore")
import gc
gc.collect()
import cv2
import tensorflow as tf
import keras
from PIL import Image, ImageDraw
from PIL import ImagePath
import imgaug.augmenters as iaa
from skimage import exposure
```

3. Exploratory Data Analysis

```
data_rle_train = pd.read_csv("./siim/train-rle.csv")
data_rle_train.head()
```

	ImageId	EncodedPixels
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1

```
#The second column contains a space infront of it so manually rewriting it
data_rle_train.columns = ['ImageId', 'EncodedPixels']
```

2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	1

- train-rle.csv contains the mask information for the images in the train dataset.
- The column EncodedPixels contains the mask information in rle format.
- Images having no mask i.e. no pneumnothorax have its EncodedPixels value as -1.

```
#https://www.kaggle.com/jesperdramsch/intro-chest-xray-dicom-viz-u-nets-full-data
def getInfoDICOM(path, toPrint = False, train = True):

    info = {}

    path = os.path.join(path)
    #reading the data using methods of pydicom
    data = pydicom.dcmread(path)
    info['path'] = path
    info['age'] = data.PatientAge
    info['sex']= data.PatientSex
    info['ImageId'] = data.SOPInstanceUID

    if train: #test doesn't have encoded pixels data, we have to predict them
        #SOPInstanceUID contains the storage type which resembles the image ids given in trai
        encodedPixels = data_rle_train[data_rle_train['ImageId'] == data.SOPInstanceUID]["Enc
        info['encodedPixels'] = encodedPixels

        info['lenOfEncodedPixels'] = len(encodedPixels)

        #this is for visualization purpose
        if '-1' in encodedPixels or len(encodedPixels) == 0:
            info['has_pneumothorax'] = 0
        else:
            info['has_pneumothorax'] = 1

    if toPrint:
        print("Path..............:", path)
        print("Patient's Name.....:", data.PatientName)
        print("Patient's Id.......:", data.PatientID)
        print("Patient's Age......:", data.PatientAge)
        print("Patient's Sex......:", data.PatientSex)

        print("Original X Ray")
        plt.figure(figsize=(10,10))
        plt.imshow(data.pixel_array, cmap=plt.cm.bone)
        plt.show()
```

```
    return info


file_name = './siim/dicom-images-train/1.2.276.0.7230010.3.1.2.8323329.11639.1517875234.49960
info = getInfoDICOM(file_name, True, True)
```

```
    Path..............: ./siim/dicom-images-train/1.2.276.0.7230010.3.1.2.8323329.11639.151
    Patient's Name.....: 9b2c32db-d3f8-4033-8083-4f7e906c5a11
    Patient's Id.......: 9b2c32db-d3f8-4033-8083-4f7e906c5a11
    Patient's Age......: 51
    Patient's Sex......: M
    Original X Ray
```

- Every dicom image contains information about the patients.
- Name and id of patients are not readable, they seem to be encrypted for privacy concerns.
- Patients age, sex are also available.
- There are also written text on the images.

3.1 Creating train and test dataset

```
#creation of train dataset
#https://stackoverflow.com/questions/33747968/getting-file-list-using-glob-in-python
filesList = glob.glob('./siim/dicom-images-train/*/*/*.dcm')
train = pd.DataFrame()
train_list = []
for file in tqdm(filesList):
    info = getInfoDICOM(file, False, True)
    train_list.append(info)
train = pd.DataFrame(train_list)
```

```
100%|██████████| 12089/12089 [00:24<00:00, 486.91it/s]
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12089 entries, 0 to 12088
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   path                12089 non-null  object
 1   age                 12089 non-null  object
 2   sex                 12089 non-null  object
 3   ImageId             12089 non-null  object
 4   encodedPixels       12089 non-null  object
 5   lenOfEncodedPixels  12089 non-null  int64 
 6   has_pneumothorax    12089 non-null  int64 
dtypes: int64(2), object(5)
memory usage: 661.2+ KB
```

```
train.head(5)
```

	path	age	sex	ImageId	enco
0	./siim/dicom-images-train/1.2.276.0.7230010.3....	30	F	1.2.276.0.7230010.3.1.4.8323329.4185.151787518...	[568 9 10
1	./siim/dicom-images-train/1.2.276.0.7230010.3....	46	M	1.2.276.0.7230010.3.1.4.8323329.14269.15178752...	
2	./siim/dicom-images-train/1.2.276.0.7230010.3....	66	M	1.2.276.0.7230010.3.1.4.8323329.3552.151787517...	
3	./siim/dicom-images-train/1.2.276.0.7230010.3....	19	F	1.2.276.0.7230010.3.1.4.8323329.5298.151787518...	
4	./siim/dicom-images-	72	F	1.2.276.0.7230010.3.1.4.8323329.2137.151787517	

```
#creation of test dataset
```

```
#creation of test dataset
filesList = glob.glob('./siim/dicom-images-test/*/*/*.dcm')
test = pd.DataFrame()
test_list = []
for file in tqdm(filesList):
    info = getInfoDICOM(file, False, False)
    test_list.append(info)
test = pd.DataFrame(test_list)
```

```
100%|██████████| 3205/3205 [00:02<00:00, 1291.75it/s]
```

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3205 entries, 0 to 3204
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   path     3205 non-null   object
 1   age      3205 non-null   object
 2   sex      3205 non-null   object
 3   ImageId  3205 non-null   object
dtypes: object(4)
memory usage: 100.3+ KB
```

```
test.head(5)
```

	path	age	sex	ImageId
0	./siim/dicom-images-test/_/_/ID_29804aff4.dcm	82	F	ID_29804aff4
1	./siim/dicom-images-test/_/_/ID_bf534a6ee.dcm	54	M	ID_bf534a6ee
2	./siim/dicom-images-test/_/_/ID_733f72db6.dcm	64	F	ID_733f72db6
3	./siim/dicom-images-test/_/_/ID_cde4075de.dcm	23	M	ID_cde4075de
4	./siim/dicom-images-test/_/_/ID_3b95d7640.dcm	40	F	ID_3b95d7640

3.2 Checking if there are any null values and replacing them

```
print("Columns having null values in train", train.columns[train.isnull().any()].tolist())
```

```
Columns having null values in train []
```

```
print("No of images having no mask", train[train['lenOfEncodedPixels'] == 0].shape[0])
```

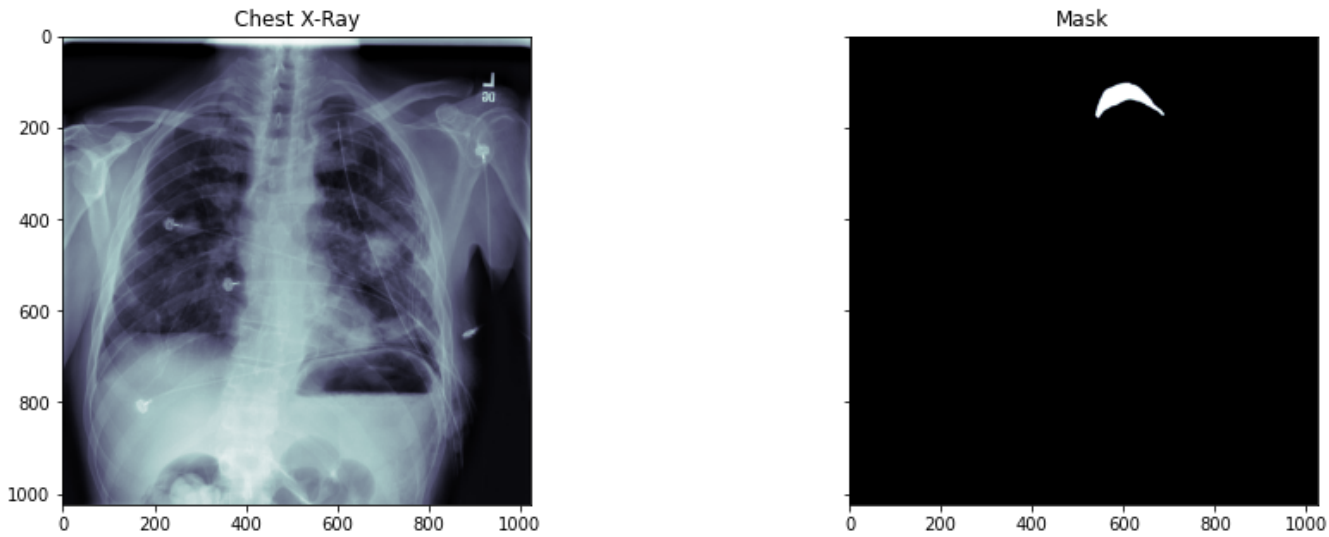
```
No of images having no mask 42
```

- There are 42 rows where the images doesn't have any mask.
- We can assume that these images don't have have pneumothorax.

```
train.loc[train.lenOfEncodedPixels == 0 , 'encodedPixels'] = np.array(-1)
```

3.3 Visualizing the mask

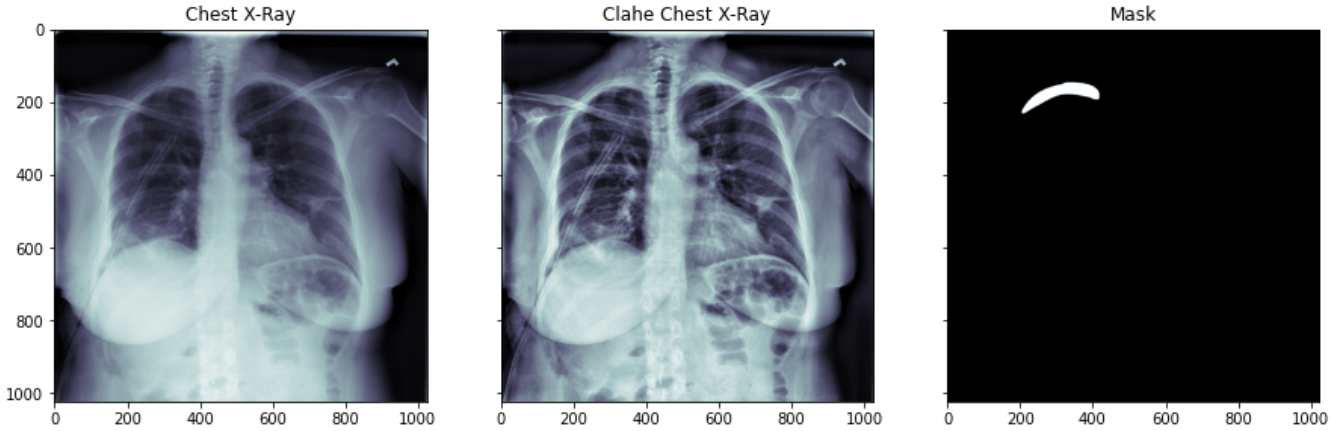
```
file_name = './siim/dicom-images-train/1.2.276.0.7230010.3.1.2.8323329.11639.1517875234.49960
path = os.path.join(file_name)
#reading the data using pydicom
data = pydicom.dcmread(path)


encodedPixels = data_rle_train[data_rle_train['ImageId'] == data.SOPInstanceUID]["EncodedPixe
mask = np.zeros((1024, 1024))
for pix in encodedPixels:
    mask = mask + rle2mask(pix, 1024, 1024).T
fig, ax = plt.subplots(nrows=1, ncols=2, sharey=True, figsize=(15,5))
ax[0].imshow(data.pixel_array, cmap=plt.cm.bone)
ax[0].set_title('Chest X-Ray')

ax[1].imshow(mask, cmap=plt.cm.bone)
ax[1].set_title('Mask')
plt.show()
```



- It is very difficult to make out from the picture what really constitutes of pneumothorax.

```
file_name = './siim/dicom-images-train/1.2.276.0.7230010.3.1.2.8323329.10005.1517875220.95895
```

```
path = os.path.join(file_name)
#reading the data using pydicom
data = pydicom.dcmread(path)

encodedPixels = data_rle_train[data_rle_train['ImageId'] == data.SOPInstanceUID]["EncodedPixe
mask = np.zeros((1024, 1024))
for pix in encodedPixels:
    mask = mask + rle2mask(pix, 1024, 1024).T

fig, ax = plt.subplots(nrows=1, ncols=3, sharey=True, figsize=(15,5))
ax[0].imshow(data.pixel_array, cmap=plt.cm.bone)
ax[0].set_title('Chest X-Ray')

#Using clahe to make it somewhat visible
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(16, 16))
clahe_pixel_array = clahe.apply(data.pixel_array)
ax[1].imshow(clahe_pixel_array, cmap=plt.cm.bone)
ax[1].set_title('Clahe Chest X-Ray')

ax[2].imshow(mask, cmap=plt.cm.bone)
ax[2].set_title('Mask')
plt.show()
```



3.3 Analysis on data

```
train['has_pneumothorax'].value_counts()
```

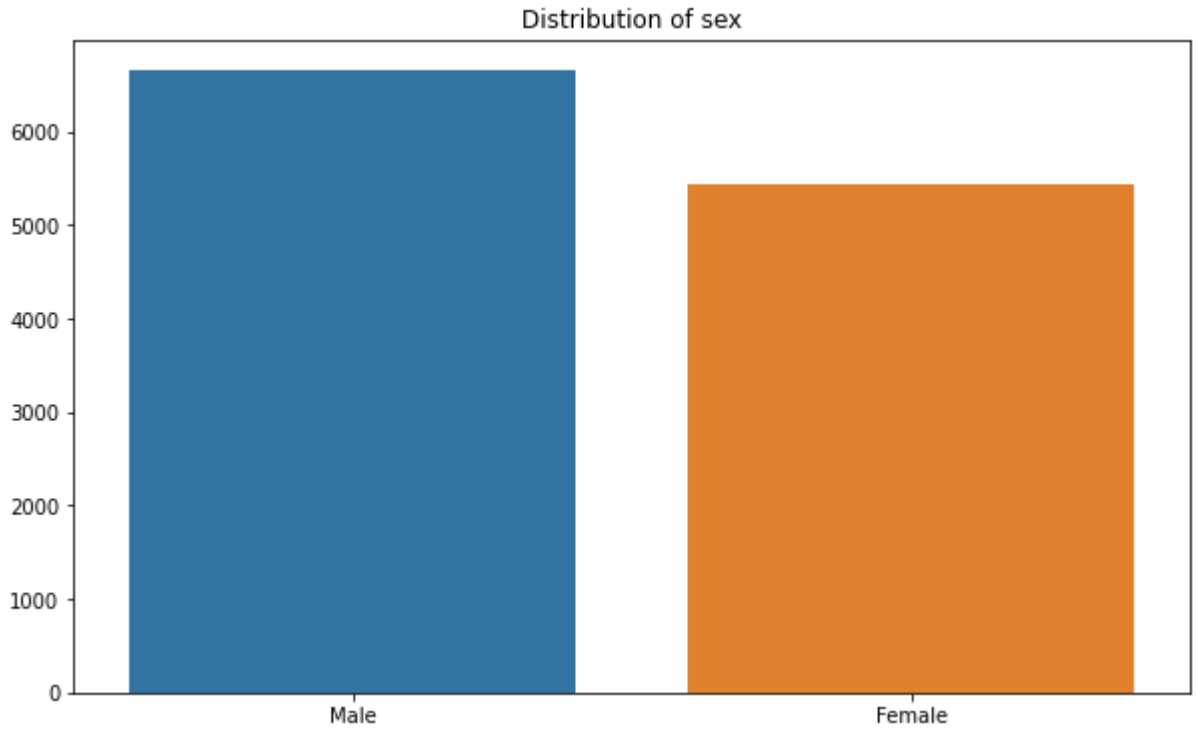
```
0    9420
1    2669
Name: has_pneumothorax, dtype: int64
```

- Out of 12089 images 2669 contains pneumothorax.

```
train['sex'].value_counts()
```

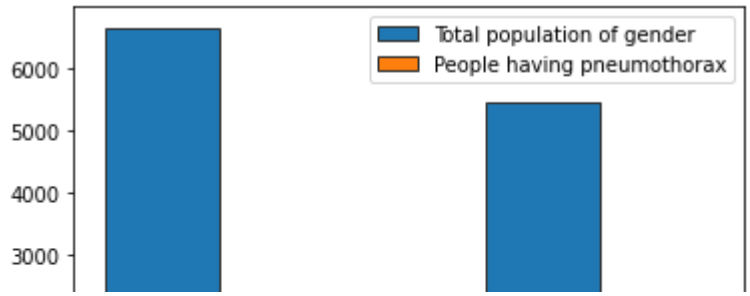
```
M    6650
F    5439
Name: sex, dtype: int64
```

```
x = ["Male" , "Female"]
y = [train['sex'].value_counts()[0] , train['sex'].value_counts()[1]]
plt.figure(figsize=(10, 6))
plt.title("Distribution of sex")
sns.barplot(x,y)
plt.show()
```



- More data is present for the male population

```
x = np.array(['Male', 'Female'])
plt.bar(x, [len(train[(train['sex'] == 'M')]), len(train[(train['sex'] == 'F')])], width=-0.3
plt.bar(x, [len(train[(train['has_pneumothorax']==1) & (train['sex'] == 'M')]), len(train[(tr
plt.legend()
plt.show()
```



- Out of total population of a particular gender many of them don't have pneumothorax

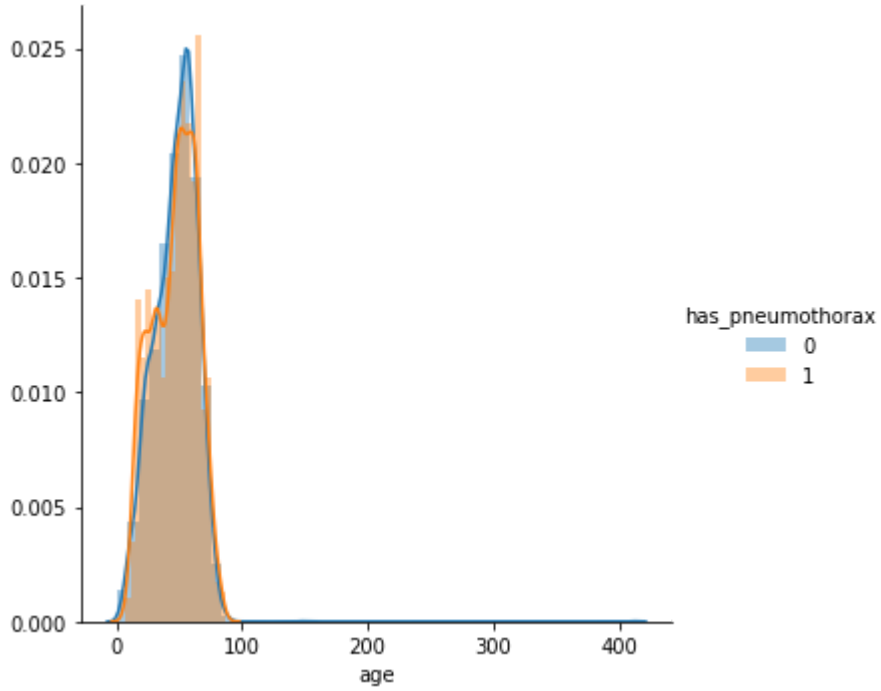
```
print("Out of total male population {} % of them have pneumothorax".format((len(train[(train[
```

```
Out of total male population 22.39097744360902 % of them have pneumothorax
```

```
print("Out of total female population {} % of them have pneumothorax".format((len(train[(trai
```

```
Out of total female population 21.695164552307407 % of them have pneumothorax
```

```
sns.FacetGrid(train, hue="has_pneumothorax", size=5) \
 .map(sns.distplot, "age") \
 .add_legend();
plt.show();
```



- It seems that pneumothorax is not depended on age.

```
print("Patient having maximum age {}".format(train['age'].max()))
```

```
Patient having maximum age 94
```

```
print("Patient having minimum age {}".format(train['age'].min()))
```

```
 Patient having minimum age 1
```

4. Data Preparation

```
def computeMasks(train, path):
    mask_paths = []
    for index, row in tqdm(train.iterrows()):
        pixels = row['encodedPixels']
        mask = np.zeros((1024, 1024))
        if row['has_pneumothorax'] == 1:
            for pix in pixels:
                mask = mask + rle2mask(pix, 1024, 1024).T
        img = Image.fromarray(mask).convert('L')
        path2save = path + str(index+1) + ".jpg"
        img.save(path2save)
        mask_paths.append(path2save)
    train["mask"] = mask_paths
    return train


path = "siim/dicom-mask-train/"
try:
    os.makedirs(path)
except:
    pass
train = computeMasks(train, path)
train.head(2)
```

```
 12089it [04:41, 42.89it/s]
```

	path	age	sex	ImageId	enco
0	./siim/dicom-images-train/1.2.276.0.7230010.3....	63	M	1.2.276.0.7230010.3.1.4.8323329.11956.15178752...	
1	./siim/dicom-images-train/1.2.276.0.7230010.3....	19	M	1.2.276.0.7230010.3.1.4.8323329.3534.151787517...	

```
def convertImagesToJpeg(train, path):
    images_paths = []
    for index, row in tqdm(train.iterrows()):
        path2dicom = row['path']
        ds = pydicom.read_file(path2dicom) # read dicom image
        img = ds.pixel_array # get image array
```

```
        img_mem = Image.fromarray(img) # Creates an image memory from an object exporting the
        path2save = path + str(index + 1)  + ".jpg"
        img_mem.save(path2save)
        images_paths.append(path2save)
    train["images_paths"] = images_paths
    return train


path = "siim/dicom-images-train-jpg/"
try:
    os.makedirs(path)
except:
    pass
train = convertImagesToJpeg(train, path)
train.head(2)
```

```
12089it [04:32, 44.42it/s]
```

	path	age	sex	ImageId	enco
0	./siim/dicom-images-train/1.2.276.0.7230010.3....	63	M	1.2.276.0.7230010.3.1.4.8323329.11956.15178752...	
1	./siim/dicom-images-train/1.2.276.0.7230010.3....	19	M	1.2.276.0.7230010.3.1.4.8323329.3534.151787517...	

```
path = "siim/dicom-images-test-jpg/"
try:
    os.makedirs(path)
except:
    pass
test = convertImagesToJpeg(test, path)
test.head(2)
```

```
3205it [01:12, 44.49it/s]
```

	path	age	sex	ImageId	images_paths
0	./siim/dicom-images-test/_/_/ID_213b7e68f.dcm	29	M	ID_213b7e68f	siim/dicom-images-test-jpg/1.jpg
1	./siim/dicom-images-	47	M	ID_f0bbc61c9	siim/dicom-images-test-

5. Data Pipeline

```
import tensorflow as tf
# tf.compat.v1.enable_eager_execution()
from tensorflow import keras
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import image
```

```
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import UpSampling2D
from tensorflow.keras.layers import MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.layers import concatenate
from tensorflow.keras.layers import Multiply
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormal
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import plot_model
from tensorflow.keras.initializers import glorot_uniform, he_normal
from tensorflow.keras.losses import binary_crossentropy
K.set_image_data_format('channels_last')


img_size = 256


def load_data(data):
    images = data['images_paths']
    masks = data['mask']
    return images, masks



def read_image(datapoint):
    image = tf.keras.preprocessing.image.load_img(datapoint)
    image = tf.keras.preprocessing.image.img_to_array(image, dtype='float32')
    image = tf.image.resize(image, [img_size, img_size])
    image = image / 255.0
    image = exposure.equalize_adapthist(image)     # contrast correction
    return image

def read_mask(datapoint):
    mask = tf.keras.preprocessing.image.load_img(datapoint, color_mode='grayscale')
    mask = tf.keras.preprocessing.image.img_to_array(mask, dtype = 'float32')
    mask = tf.image.resize(mask, [img_size, img_size])
    mask = mask / 255.0
    return mask

def preprocess(image, image_mask):
    def f(image, image_mask):

        image = image.decode()
        image_mask = image_mask.decode()

        image = read_image(image)
        image_mask = read_mask(image_mask)

        a = np.random.uniform()
        if a<0.50:
            image = tf.image.flip_left_right(image)
            image_mask = tf.image.flip_left_right(image_mask)
        else:
```

```
            else:
                image = tf.image.flip_up_down(image)
                image_mask = tf.image.flip_up_down(image_mask)

            return image, image_mask
        images, masks = tf.numpy_function(f, [image, image_mask], [tf.float32, tf.float32])
        images.set_shape([img_size, img_size, 3])
        masks.set_shape([img_size, img_size, 1])

        return images, masks

    def tf_dataset(x, y, batch=8):
        dataset = tf.data.Dataset.from_tensor_slices((x, y))
        dataset = dataset.shuffle(buffer_size=15000)
        dataset = dataset.map(preprocess)
        dataset = dataset.batch(batch)
        dataset = dataset.prefetch(2)
        return dataset


    def display(display_list):
        plt.figure(figsize=(8, 8))

        title = ['Input Image', 'True Mask', 'Predicted Mask']

        for i in range(len(display_list)):
            plt.subplot(1, len(display_list), i+1)
            plt.title(title[i])
            plt.imshow(tf.keras.preprocessing.image.array_to_img(display_list[i]))
            plt.axis('off')
        plt.show()


    def load_data_for_predict(data):
        images = data['images_paths']
        image_id = data['ImageId']
        return images, image_id

    def preprocess_predict(image, image_id):
        def f(image, image_id):

            image = image.decode()
            image_id = image_id.decode()

            image = read_image(image)

            return image, image_id

        images, imageid = tf.numpy_function(f, [image, image_id], [tf.float32, tf.string])
        images.set_shape([img_size, img_size, 3])

        return images, image_id
```

```
def tf_dataset_predict(x, y, batch=8):
    dataset = tf.data.Dataset.from_tensor_slices((x, y))
    dataset = dataset.shuffle(buffer_size=15000)
    dataset = dataset.map(preprocess_predict)
    dataset = dataset.batch(batch)
    dataset = dataset.prefetch(2)
    return dataset
```

6. Model

1. Here i will be taking only the pneumothorax data as the dataset is severly imbalanced.

```
only_pnemothorax = train[train['has_pneumothorax'] == 1]
```

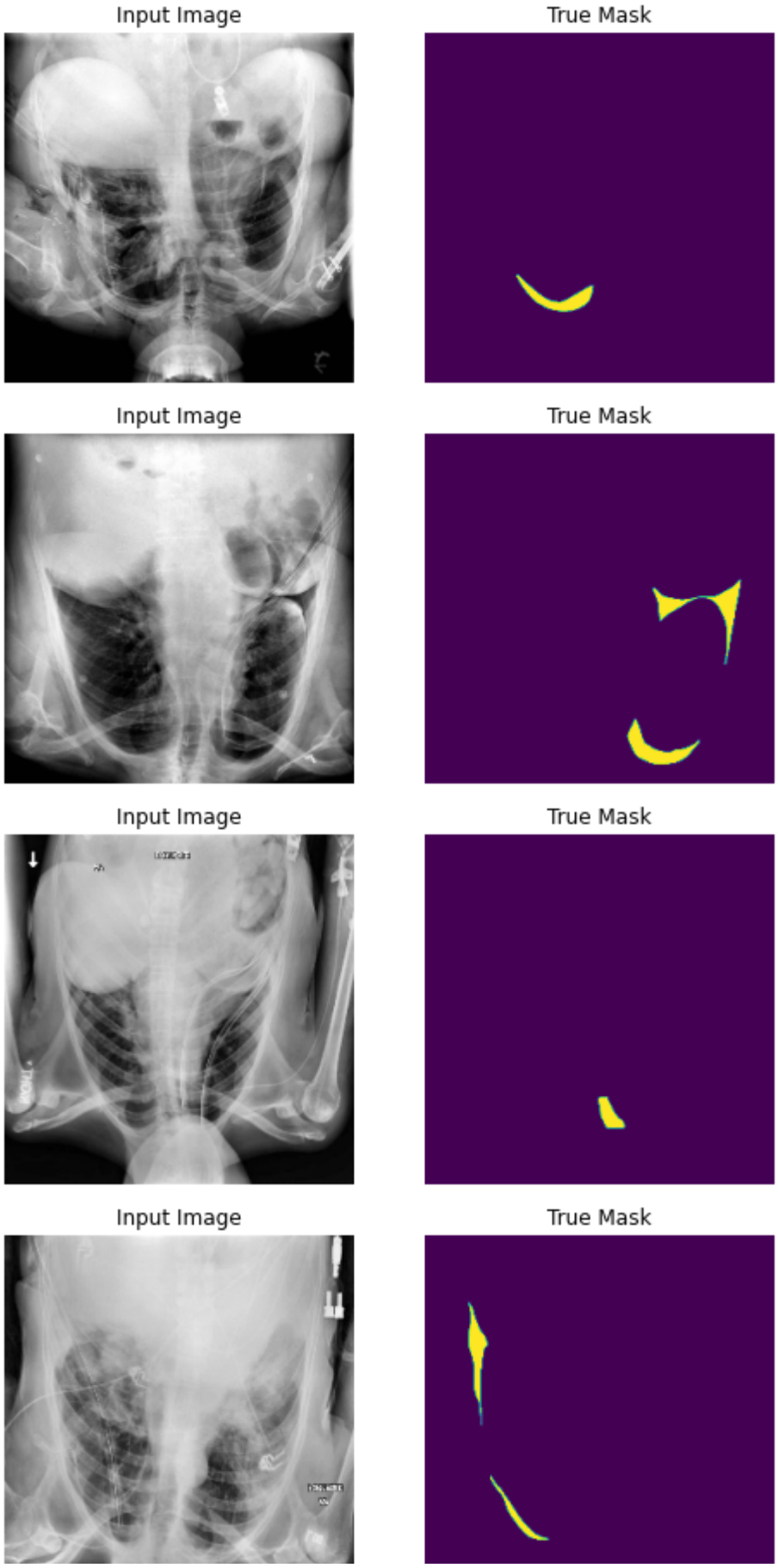
```
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split(only_pnemothorax, test_size=0.10, random_state=42)
```

```
X_train.head(2)
```

	path	age	sex	ImageId
835	./siim/dicom-images-train/1.2.276.0.7230010.3....	73	F	1.2.276.0.7230010.3.1.4.8323329.11383.15178752...
10336	./siim/dicom-images-train/1.2.276.0.7230010.3....	42	F	1.2.276.0.7230010.3.1.4.8323329.1460.151787516...

```
images, masks = load_data(X_train)
train_dataset = tf_dataset(images, masks, 8)

images, masks = load_data(X_test)
test_dataset = tf_dataset(images, masks, 8)
```

```
for image, mask in train_dataset.take(1):
    sample_image, sample_mask = image, mask
for i in range(4):
    display([sample_image[i], sample_mask[i]])
```




```
class BasicBlock(tf.keras.layers.Layer):

    def __init__(self, filters, stride=1):
```

```
        def __init__(self, filters, stride 1):
            super(BasicBlock, self).__init__()
            self.conv1_basic = Conv2D(filters=filters, kernel_size=(3, 3),strides=stride, padding
            self.batchNorm1_basic = BatchNormalization()
            self.conv2_basic = Conv2D(filters=filters,kernel_size=(3, 3),strides=1,padding="same"
            self.batchNorm2_basic = BatchNormalization()
            if stride != 1:
                self.downsample_basic = tf.keras.Sequential()
                self.downsample_basic.add(Conv2D(filters=filters, kernel_size=(1, 1), strides=str
                self.downsample_basic.add(BatchNormalization())
            else:
                self.downsample_basic = lambda x: x

        def call(self, inputs, training=None, **kwargs):
            residual = self.downsample_basic(inputs)

            x = self.conv1_basic(inputs)
            x = self.batchNorm1_basic(x, training=training)
            x = tf.nn.relu(x)
            x = self.conv2_basic(x)
            x = self.batchNorm2_basic(x, training=training)

            output = tf.nn.relu(tf.keras.layers.add([residual, x]))

            return output

    def make_basic_block_layer(filter_num, blocks, stride=1):
        res_block = tf.keras.Sequential()
        res_block.add(BasicBlock(filter_num, stride=stride))

        for _ in range(1, blocks):
            res_block.add(BasicBlock(filter_num, stride=1))

        return res_block



    class UNETBlock(tf.keras.Model):
        def __init__(self):
            super().__init__()
            self.conv1_unet = Conv2D(16, 3, activation = 'relu', padding = 'same', kernel_initial
            self.conv1_1_unet = Conv2D(16, 3, activation = 'relu', padding = 'same', kernel_initi
            self.conv1_2_unet = Conv2D(16, 3, activation = 'relu', padding = 'same', kernel_initi
            self.conv1_3_unet = Conv2D(16, 3, activation = 'relu', padding = 'same', kernel_initi

            self.pool1_unet = MaxPooling2D(pool_size=(2, 2))

            self.conv2_unet = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initial
            self.conv2_1_unet = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initi
            self.conv2_2_unet = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initi
            self.conv2_3_unet = Conv2D(32, 3, activation = 'relu', padding = 'same', kernel_initi

            self.conv3_unet = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initial
```

```
        self.conv3_unet = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initial
        self.conv3_1_unet = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initi
        self.conv3_2_unet = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initi
        self.conv3_3_unet = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initi

        self.conv4_unet = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initia
        self.conv4_1_unet = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_init
        self.conv4_2_unet = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_init
        self.conv4_3_unet = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_init

        self.drop1_unet = Dropout(0.2)
        self.conv5_unet = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initia
        self.conv5_1_unet = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_init

        self.up1_unet = UpSampling2D(size = (2,2))
        self.conv6_unet = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initia
        self.conv6_1_unet = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_init

        self.conv7_unet = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initial
        self.conv8_unet = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initial
        self.conv9_unet = Conv2D(16, 2, activation = 'relu', padding = 'same', kernel_initial
        self.conv10_unet = Conv2D(2, 3, activation = 'relu', padding = 'same', kernel_initial
        self.conv11_unet = Conv2D(1, 1, activation = 'sigmoid')

    def call(self, inputs):
        x = self.conv1_unet(inputs)
        conv_1 = self.conv1_1_unet(x)
        x = self.pool1_unet(conv_1)
        x = self.conv2_unet(x)
        conv_2 = self.conv2_1_unet(x)
        x = self.pool1_unet(x)
        x = self.conv3_unet(x)
        conv_3 = self.conv3_1_unet(x)
        x = self.pool1_unet(conv_3)
        x = self.conv4_unet(x)
        x = self.conv4_1_unet(x)
        drop_1 = self.drop1_unet(x)
        x = self.pool1_unet(drop_1)

        x = self.conv5_unet(x)
        x = self.conv5_1_unet(x)
        drop_2 = self.drop1_unet(x)

        x = self.conv6_unet(self.up1_unet(drop_2))
        x = concatenate([drop_1, x], axis = -1)

        x = self.conv4_2_unet(x)
        x = self.conv4_3_unet(x)

        x = self.conv7_unet(self.up1_unet(x))
        x = concatenate([conv_3, x], axis = -1)
        x = self.conv3_2_unet(x)
```

```
        x = self.conv3_3_unet(x)

        x = self.conv8_unet(self.up1_unet(x))
        x = concatenate([conv_2, x], axis = -1)
        x = self.conv2_2_unet(x)
        x = self.conv2_3_unet(x)

        x = self.conv9_unet(self.up1_unet(x))
        x = self.conv1_2_unet(x)
        x = self.conv1_3_unet(x)
        x = self.conv10_unet(x)
        output = self.conv11_unet(x)

        return output



class UnetResnet34(tf.keras.Model):
    def __init__(self, input_shape, layer_params):
        super().__init__()

        self.conv1_resnet = tf.keras.layers.Conv2D(filters=32,
                                           kernel_size=(7, 7),
                                           strides=2,
                                           padding="same")
        self.bn1_resnet = tf.keras.layers.BatchNormalization()
        self.pool1_resnet = tf.keras.layers.MaxPool2D(pool_size=(3, 3),
                                              strides=2,
                                              padding="same")

        self.layer1_resnet = make_basic_block_layer(filter_num=32,
                                            blocks=layer_params[0])
        self.layer2_resnet = make_basic_block_layer(filter_num=64,
                                            blocks=layer_params[1],
                                            stride=2)
        self.layer3_resnet = make_basic_block_layer(filter_num=128,
                                            blocks=layer_params[2],
                                            stride=2)
        self.layer4_resnet = make_basic_block_layer(filter_num=256,
                                            blocks=layer_params[3],
                                            stride=2)

        self.avgpool_resnet = tf.keras.layers.AveragePooling2D()

        self.unetBlock_resnet = UNETBlock()

        self.upsample_resnet = tf.keras.layers.UpSampling2D(size = (64, 64))
        self.conv1_resize = tf.keras.layers.Conv2D(filters=1,
                                           kernel_size=(3, 3),
                                           strides=1,
                                           padding="same")
```

```
    def call(self, inputs, training=None, mask=None):
        x = self.conv1_resnet(inputs)
        x = self.bn1_resnet(x, training=training)
        x = tf.nn.relu(x)
        x = self.pool1_resnet(x)
        x = self.layer1_resnet(x, training=training)
        x = self.layer2_resnet(x, training=training)
        x = self.layer3_resnet(x, training=training)
        x = self.layer4_resnet(x, training=training)
        x = self.avgpool_resnet(x)
        x = self.upsample_resnet(x)
        x = self.conv1_resize(x)
        output = self.unetBlock_resnet(x)

        return output


model = UnetResnet34((256, 256, 3),[3, 4, 6, 3])


smooth = 1e-5
def dice_coef(y_true, y_pred):
    y_true_f = tf.keras.layers.Flatten()(y_true)
    y_pred_f = tf.keras.layers.Flatten()(y_pred)
    intersection = tf.reduce_sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (tf.reduce_sum(y_true_f) + tf.reduce_sum(y_pred_f)


def combined_loss(y_true, y_pred):
    return 1.0 - dice_coef(y_true, y_pred) + binary_crossentropy(y_true, y_pred)


callbacks = [
    tf.keras.callbacks.ModelCheckpoint('./best_model1.h5', save_weights_only=True, save_best_
                                       mode='min'),
]


optim = tf.keras.optimizers.Adam(0.001)

metrics=[dice_coef]
loss = combined_loss

model.compile(optim, loss , metrics)


history = model.fit(train_dataset, steps_per_epoch=len(train_dataset), epochs=70,\
                              validation_data=test_dataset,callbacks=callbacks, )
```

```
 Epoch 1/70
 301/301 [==============================] - 197s 636ms/step - loss: 1.1040 - dice_coef
 Epoch 2/70
```

```
301/301 [==============================] - 191s 634ms/step - loss: 1.0159 - dice_coef
Epoch 3/70
301/301 [==============================] - 194s 644ms/step - loss: 1.0069 - dice_coef
Epoch 4/70
301/301 [==============================] - 196s 653ms/step - loss: 0.9938 - dice_coef
Epoch 5/70
301/301 [==============================] - 193s 642ms/step - loss: 0.9861 - dice_coef
Epoch 6/70
301/301 [==============================] - 192s 637ms/step - loss: 0.9702 - dice_coef
Epoch 7/70
301/301 [==============================] - 190s 632ms/step - loss: 0.9632 - dice_coef
Epoch 8/70
301/301 [==============================] - 191s 635ms/step - loss: 0.9496 - dice_coef
Epoch 9/70
301/301 [==============================] - 195s 646ms/step - loss: 0.9507 - dice_coef
Epoch 10/70
301/301 [==============================] - 194s 644ms/step - loss: 0.9444 - dice_coef
Epoch 11/70
301/301 [==============================] - 190s 630ms/step - loss: 0.9353 - dice_coef
Epoch 12/70
301/301 [==============================] - 185s 616ms/step - loss: 0.9476 - dice_coef
Epoch 13/70
301/301 [==============================] - 184s 612ms/step - loss: 0.9250 - dice_coef
Epoch 14/70
301/301 [==============================] - 188s 626ms/step - loss: 0.9093 - dice_coef
Epoch 15/70
301/301 [==============================] - 188s 625ms/step - loss: 0.9082 - dice_coef
Epoch 16/70
301/301 [==============================] - 191s 634ms/step - loss: 0.9111 - dice_coef
Epoch 17/70
301/301 [==============================] - 190s 630ms/step - loss: 0.9334 - dice_coef
Epoch 18/70
301/301 [==============================] - 192s 638ms/step - loss: 0.9224 - dice_coef
Epoch 19/70
301/301 [==============================] - 193s 640ms/step - loss: 0.9029 - dice_coef
Epoch 20/70
301/301 [==============================] - 187s 621ms/step - loss: 0.8799 - dice_coef
Epoch 21/70
301/301 [==============================] - 182s 603ms/step - loss: 0.8886 - dice_coef
Epoch 22/70
301/301 [==============================] - 191s 633ms/step - loss: 0.8869 - dice_coef
Epoch 23/70
301/301 [==============================] - 189s 627ms/step - loss: 0.8787 - dice_coef
Epoch 24/70
301/301 [==============================] - 189s 626ms/step - loss: 0.8697 - dice_coef
Epoch 25/70
301/301 [==============================] - 189s 627ms/step - loss: 0.8644 - dice_coef
Epoch 26/70
301/301 [==============================] - 192s 638ms/step - loss: 0.8476 - dice_coef
Epoch 27/70
301/301 [==============================] - 191s 634ms/step - loss: 0.8551 - dice_coef
Epoch 28/70
301/301 [==============================] - 191s 633ms/step - loss: 0.8360 - dice_coef
Epoch 29/70
301/301 [==============================] - 189s 627ms/step - loss: 0.8368 - dice_coef
```

```
#Saving this model as it is showing better results
```

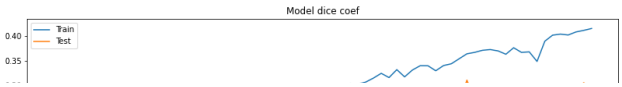
```
#saving this model as it is showing better results
model.save("pneumothorax_predictor")
```

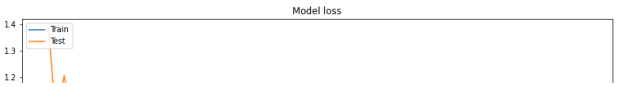
```
WARNING:tensorflow:Skipping full serialization of Keras layer <tensorflow.python.keras.l
WARNING:absl:Found untraced functions such as conv2d_125_layer_call_fn, conv2d_125_layer
WARNING:absl:Found untraced functions such as conv2d_125_layer_call_fn, conv2d_125_layer
INFO:tensorflow:Assets written to: pneumothorax_predictor/assets
INFO:tensorflow:Assets written to: pneumothorax_predictor/assets
```

```
#zipping so that it can be downloaded in colab
!zip -r /content/pneumothorax_predictor.zip /content/pneumothorax_predictor/
```

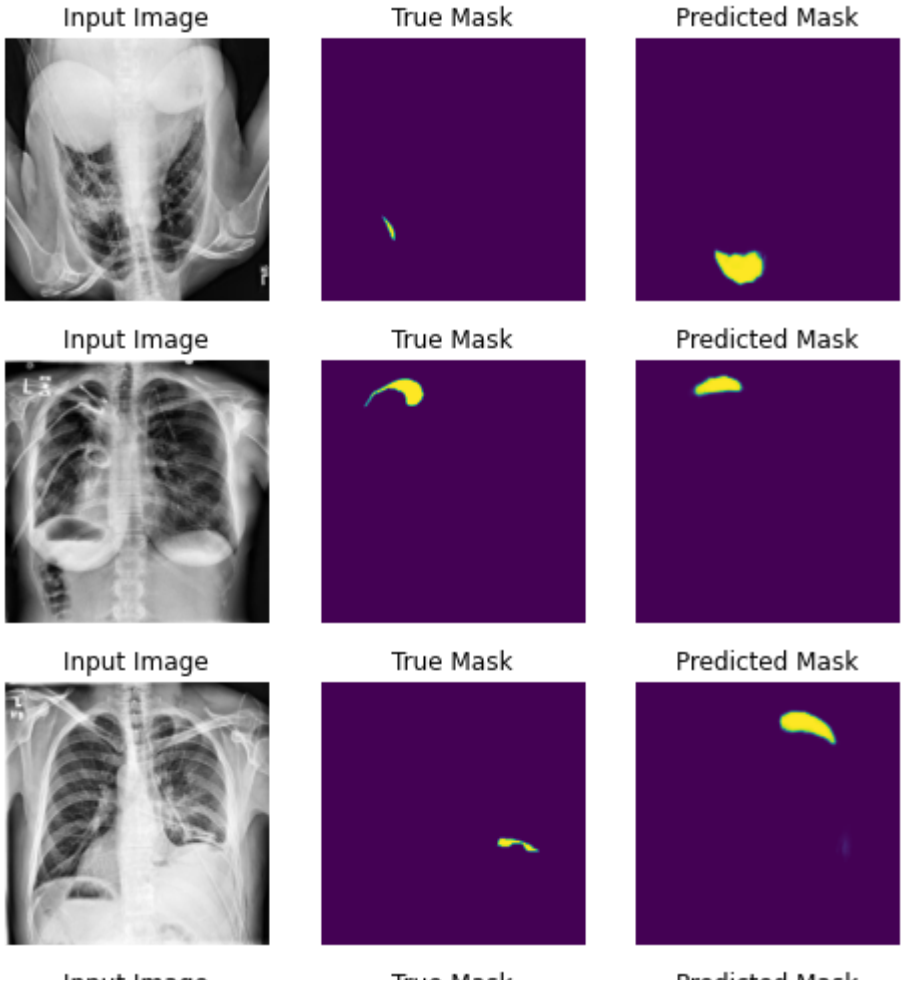
```
  adding: content/pneumothorax_predictor/ (stored 0%)
  adding: content/pneumothorax_predictor/variables/ (stored 0%)
  adding: content/pneumothorax_predictor/variables/variables.data-00000-of-00001 (deflat
  adding: content/pneumothorax_predictor/variables/variables.index (deflated 80%)
  adding: content/pneumothorax_predictor/assets/ (stored 0%)
  adding: content/pneumothorax_predictor/saved_model.pb (deflated 91%)
```

```
# Plot training & validation iou_score values
plt.figure(figsize=(30, 5))
plt.subplot(121)
plt.plot(history.history['dice_coef'])
plt.plot(history.history['val_dice_coef'])
plt.title('Model dice coef')
plt.ylabel('dice coef')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')

# Plot training & validation loss values
plt.subplot(122)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```






```
for image, mask in train_dataset.take(1):
    sample_image, sample_mask = image, mask
    pred_masks = model.predict(sample_image)
    for i in range(8):
        display([sample_image[i], sample_mask[i], pred_masks[i]])
```




```
images , imageid = load_data_for_predict(test)
test_sub = tf_dataset_predict(images, imageid)
```

```
submission = []
for image, imageid in test_sub:
    pred_masks = model.predict(image)
    for i in range(len(pred_masks)):
        prediction = {}
        prediction['ImageId'] = imageid[i].numpy().decode()
        pred_masks[i] = (pred_masks[i] > .5).astype(int)

        if pred_masks[i].sum() < 1:
            prediction['EncodedPixels'] =  -1
        else:
            prediction['EncodedPixels'] = mask2rle(pred_masks[i] * 255, 256, 256)

        submission.append(prediction)
```

```
submission_df = pd.DataFrame(submission)
submission_df = submission_df[['ImageId','EncodedPixels']]
submission_df.head()
```

	ImageId	EncodedPixels
0	ID_62d18c79f	11935 20 233 25 228 30 225 31 224 33 221 37 21...
1	ID_5c658c92c	5217 10 245 13 241 16 239 18 237 19 236 20 236...
2	ID_19cceaa1a	10138 15 239 18 237 21 233 25 230 29 226 31 22...
3	ID_88371969e	10142 16 237 21 234 24 231 27 228 29 227 30 22...
4	ID_474682dd8	43724 2 254 2 254 3 252 4 252 4 253 4 252 4 25...

2. Here i am taking data equally from both these two datasets and also i having am the model using the weights i learned from my previous model.

```
pneumothorax_data = train[train['has_pneumothorax'] == 1]
non_pneumothorax_data = train[train['has_pneumothorax'] != 1]
non_pneumothorax_data = non_pneumothorax_data.sample(3000)
new_dataset = pd.concat([pneumothorax_data,non_pneumothorax_data],axis = 0)
new_dataset.head(2)
```

	path	age	sex	ImageId	enc
7	./siim/dicom-images-train/1.2.276.0.7230010.3....	52	F	1.2.276.0.7230010.3.1.4.8323329.2421.151787517...	[33 1
12	./siim/dicom-images-train/1.2.276.0.7230010.3....	31	F	1.2.276.0.7230010.3.1.4.8323329.32430.15178751...	[8 50 72

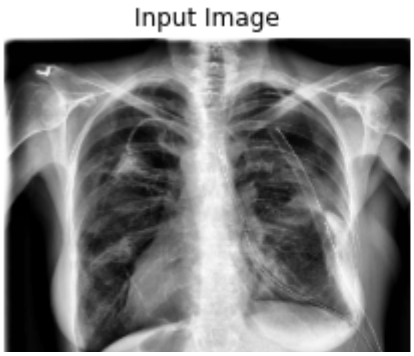
```
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split(new_dataset, test_size=0.10, random_state=42)
```

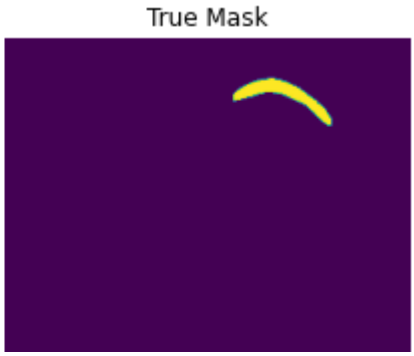
```
X_train.head(2)
```

	path	age	sex	ImageId	e
8281	./siim/dicom-images-train/1.2.276.0.7230010.3....	61	M	1.2.276.0.7230010.3.1.4.8323329.6177.151787519...	[1
5902	./siim/dicom-images-train/1.2.276.0.7230010.3....	49	F	1.2.276.0.7230010.3.1.4.8323329.1615.151787516...	

```
images, masks = load_data(X_train)
```

```
    images, masks = load_data(X_train)
    train_dataset = tf_dataset(images, masks, 8)

    images, masks = load_data(X_test)
    test_dataset = tf_dataset(images, masks, 8)


    for image, mask in train_dataset.take(1):
        sample_image, sample_mask = image, mask
    for i in range(4):
        display([sample_image[i], sample_mask[i]])
```






```
model2 = UnetResnet34((256, 256, 3),[3, 4, 6, 3])
```

```
smooth = 1e-5
def dice_coef(y_true, y_pred):
    y_true_f = tf.keras.layers.Flatten()(y_true)
    y_pred_f = tf.keras.layers.Flatten()(y_pred)
    intersection = tf.reduce_sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (tf.reduce_sum(y_true_f) + tf.reduce_sum(y_pred_f)


def combined_loss(y_true, y_pred):
    return 1.0 - dice_coef(y_true, y_pred) + binary_crossentropy(y_true, y_pred)
```

```
callbacks = [
    tf.keras.callbacks.ModelCheckpoint('./best_model2.h5', save_weights_only=True, save_best_
                                       mode='min'),
   tf.keras.callbacks.ReduceLROnPlateau(patience = 10, min_lr= 0.000001)
]
```

```
optim = tf.keras.optimizers.Adam(0.0001)

metrics=[dice_coef]
loss = combined_loss

model2.compile(optim, loss , metrics)
```

```
model2.fit(train_dataset, steps_per_epoch=len(train_dataset), epochs=1,\
                            validation_data=test_dataset,callbacks=callbacks, )
```

```
638/638 [==============================] - 411s 623ms/step - loss: 1.9094 - dice_coef: 0
<tensorflow.python.keras.callbacks.History at 0x7f31459fa750>
```

```
#Loading weights from the previous model
model2.load_weights("best_model1.h5")
```

```
history2 = model2.fit(train_dataset, steps_per_epoch=len(train_dataset), epochs=25,\
                            validation_data=test_dataset,callbacks=callbacks, )
```

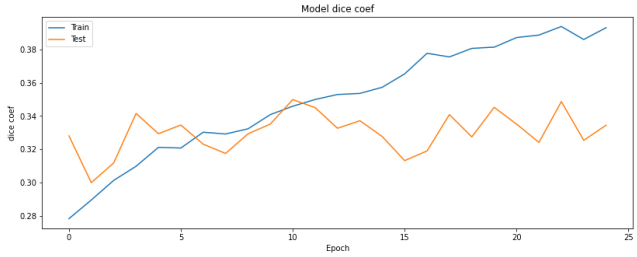
```
                    validation_data=test_dataset,callbacks=callbacks, )
```

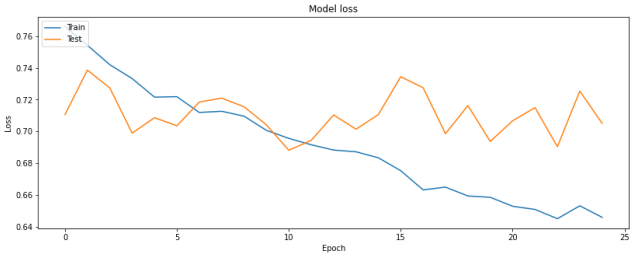
```
Epoch 1/25
638/638 [==============================] - 408s 640ms/step - loss: 0.7660 - dice_coef: 0
Epoch 2/25
638/638 [==============================] - 408s 639ms/step - loss: 0.7542 - dice_coef: 0
Epoch 3/25
638/638 [==============================] - 409s 641ms/step - loss: 0.7420 - dice_coef: 0
Epoch 4/25
638/638 [==============================] - 410s 642ms/step - loss: 0.7332 - dice_coef: 0
Epoch 5/25
638/638 [==============================] - 407s 638ms/step - loss: 0.7216 - dice_coef: 0
Epoch 6/25
638/638 [==============================] - 405s 634ms/step - loss: 0.7218 - dice_coef: 0
Epoch 7/25
638/638 [==============================] - 405s 634ms/step - loss: 0.7119 - dice_coef: 0
Epoch 8/25
638/638 [==============================] - 398s 623ms/step - loss: 0.7127 - dice_coef: 0
Epoch 9/25
638/638 [==============================] - 403s 632ms/step - loss: 0.7096 - dice_coef: 0
Epoch 10/25
638/638 [==============================] - 401s 629ms/step - loss: 0.7006 - dice_coef: 0
Epoch 11/25
638/638 [==============================] - 399s 625ms/step - loss: 0.6956 - dice_coef: 0
Epoch 12/25
638/638 [==============================] - 407s 638ms/step - loss: 0.6916 - dice_coef: 0
Epoch 13/25
638/638 [==============================] - 409s 641ms/step - loss: 0.6882 - dice_coef: 0
Epoch 14/25
638/638 [==============================] - 406s 637ms/step - loss: 0.6871 - dice_coef: 0
Epoch 15/25
638/638 [==============================] - 392s 615ms/step - loss: 0.6834 - dice_coef: 0
Epoch 16/25
638/638 [==============================] - 390s 612ms/step - loss: 0.6752 - dice_coef: 0
Epoch 17/25
638/638 [==============================] - 389s 610ms/step - loss: 0.6632 - dice_coef: 0
Epoch 18/25
638/638 [==============================] - 389s 610ms/step - loss: 0.6649 - dice_coef: 0
Epoch 19/25
638/638 [==============================] - 387s 606ms/step - loss: 0.6593 - dice_coef: 0
Epoch 20/25
638/638 [==============================] - 385s 603ms/step - loss: 0.6585 - dice_coef: 0
Epoch 21/25
638/638 [==============================] - 389s 609ms/step - loss: 0.6528 - dice_coef: 0
Epoch 22/25
638/638 [==============================] - 387s 607ms/step - loss: 0.6508 - dice_coef: 0
Epoch 23/25
638/638 [==============================] - 386s 605ms/step - loss: 0.6450 - dice_coef: 0
Epoch 24/25
638/638 [==============================] - 386s 604ms/step - loss: 0.6531 - dice_coef: 0
Epoch 25/25
638/638 [==============================] - 385s 604ms/step - loss: 0.6458 - dice_coef: 0
```

```
model2.summary()
```

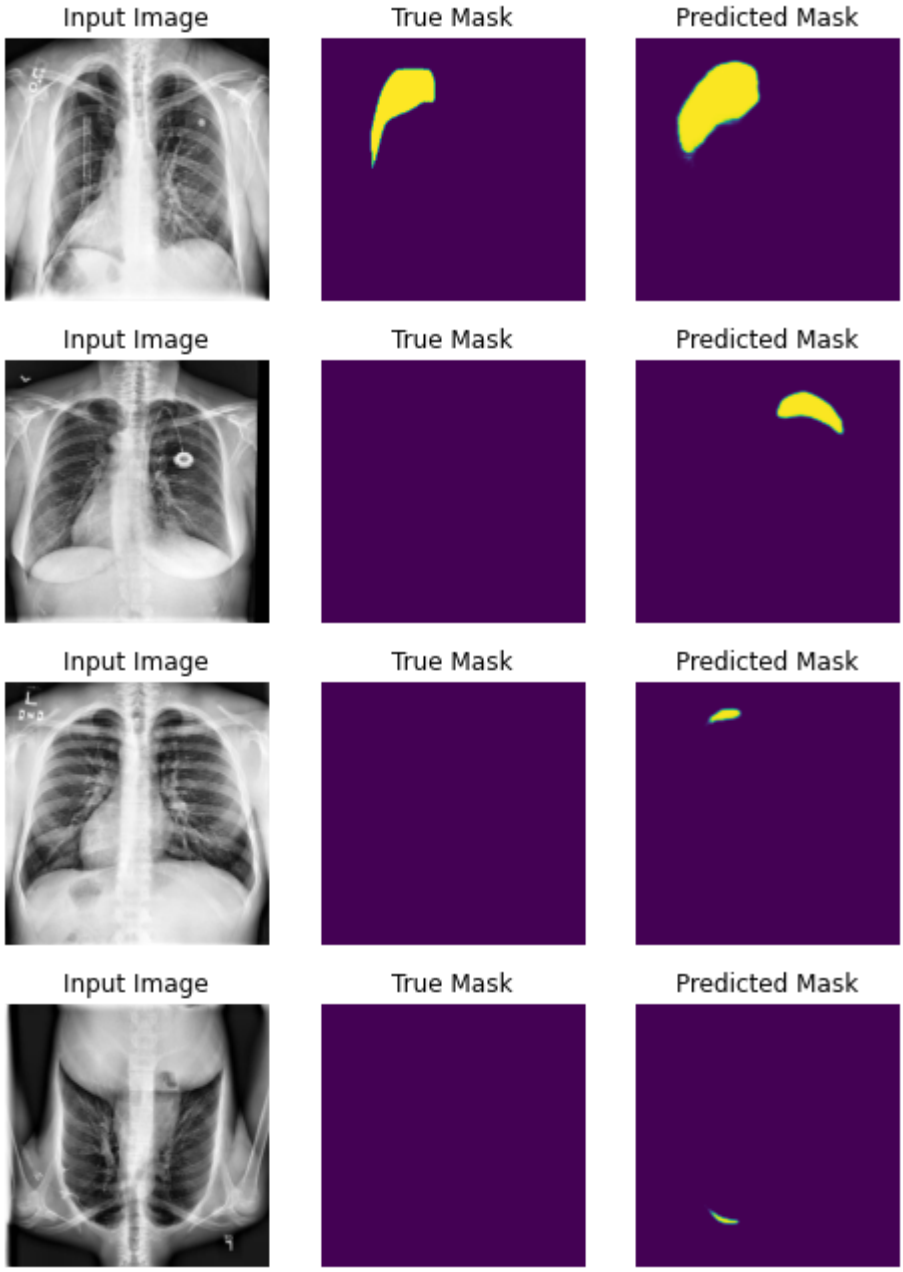
```
Model: "unet_resnet34_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_62 (Conv2D)           multiple                  4736
_________________________________________________________________
batch_normalization_36 (Batc multiple                  128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 multiple                  0
_________________________________________________________________
sequential_7 (Sequential)    (None, 64, 64, 32)        56256
_________________________________________________________________
sequential_8 (Sequential)    (None, 32, 32, 64)        281408
_________________________________________________________________
sequential_10 (Sequential)   (None, 16, 16, 128)       1712256
_________________________________________________________________
sequential_12 (Sequential)   (None, 8, 8, 256)         3285760
_________________________________________________________________
average_pooling2d_1 (Average multiple                  0
_________________________________________________________________
unet_block_1 (UNETBlock)     multiple                  1956229
_________________________________________________________________
up_sampling2d_3 (UpSampling2 multiple                  0
_________________________________________________________________
conv2d_123 (Conv2D)          multiple                  2305
=================================================================
Total params: 7,299,078
Trainable params: 7,290,566
Non-trainable params: 8,512
_________________________________________________________________
```

```
# Plot training & validation iou_score values
plt.figure(figsize=(30, 5))
plt.subplot(121)
plt.plot(history2.history['dice_coef'])
plt.plot(history2.history['val_dice_coef'])
plt.title('Model dice coef')
plt.ylabel('dice coef')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')

# Plot training & validation loss values
plt.subplot(122)
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```






```
for image, mask in train_dataset.take(1):
    sample_image, sample_mask = image, mask
    pred_masks = model2.predict(sample_image)
    for i in range(8):
        display([sample_image[i], sample_mask[i], pred_masks[i]])
```




```
images , imageid = load_data_for_predict(test)
test_sub = tf_dataset_predict(images, imageid)
```

```
submission = []
for image, imageid in test_sub:
    pred_masks = model2.predict(image)
    for i in range(len(pred_masks)):
        prediction = {}
        prediction['ImageId'] = imageid[i].numpy().decode()
        pred_masks[i] = (pred_masks[i] > .5).astype(int)

        if pred_masks[i].sum() < 1:
            prediction['EncodedPixels'] =  -1
        else:
            prediction['EncodedPixels'] = mask2rle(pred_masks[i] * 255, 256, 256)
```

```
        submission.append(prediction)
```

```
submission_df = pd.DataFrame(submission)
submission_df = submission_df[['ImageId','EncodedPixels']]
submission_df.head()
```

	ImageId	EncodedPixels
0	ID_73c53cd54	-1
1	ID_b4480fd55	7568 14 239 22 233 26 229 30 225 33 223 35 221...
2	ID_8dbfc31c5	-1
3	ID_2251f0828	4700 10 243 15 238 21 231 26 228 29 225 32 223...
4	ID_636060bd7	-1

Classifying whether a patient has pneumothorax.

```
from sklearn import metrics
```

```
fileNames = train['images_paths']
labels = train['has_pneumothorax']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(fileNames, labels, test_size=0.10, strati
```

```
X_train.head(2)
```

```
     8511    siim/dicom-images-train-jpg/8512.jpg
     9211    siim/dicom-images-train-jpg/9212.jpg
     Name: images_paths, dtype: object
```

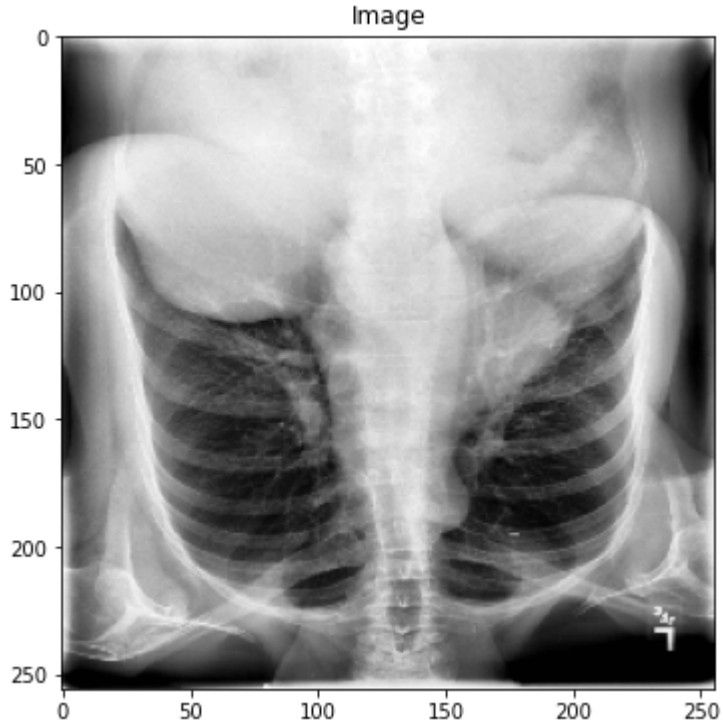
```
y_train.head(2)
```

```
     8511    1
     9211    0
     Name: has_pneumothorax, dtype: int64
```

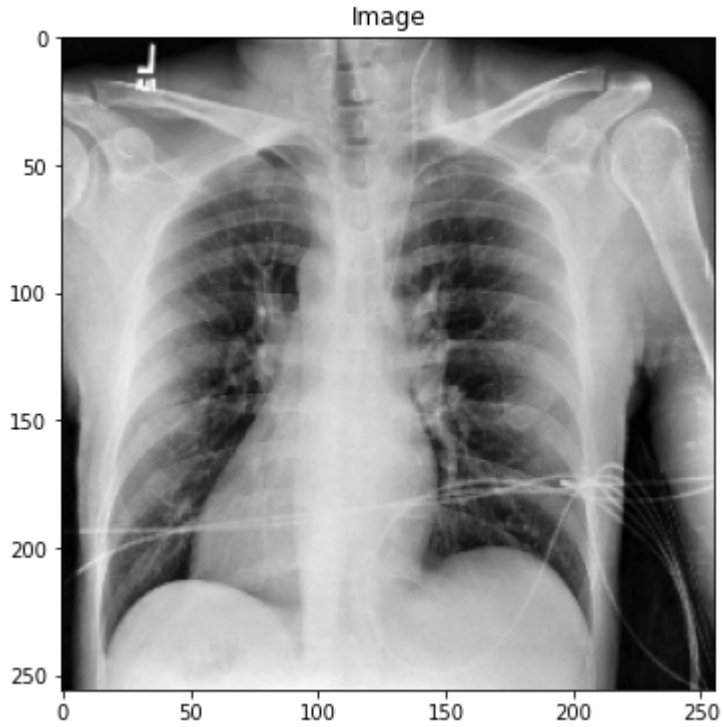
```
img_size = 256
```

```
def read_image(datapoint):
    image = tf.keras.preprocessing.image.load_img(datapoint)
    image = tf.keras.preprocessing.image.img_to_array(image, dtype='float32')
    image = tf.image.resize(image, [img_size, img_size])
    image = image / 255.0
```

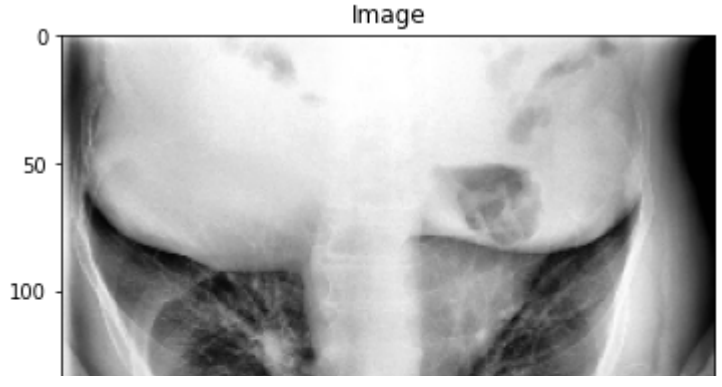
```
        image = image / 255.0
        image = exposure.equalize_adapthist(image)     # contrast correction
        return image

    def read_label(datapoint):
        datapoint = tf.reshape(datapoint, [1])
        return datapoint

    def preprocess(image, label):
        def f(image, label):

            image = image.decode()
            image = read_image(image)

            a = np.random.uniform()
            if a<0.50:
                image = tf.image.flip_left_right(image)
            else:
                image = tf.image.flip_up_down(image)

            label = read_label(label)
            return image, label
        images, labels = tf.numpy_function(f, [image, label], [tf.float32, tf.int64])
        images.set_shape([img_size, img_size, 3])
        labels.set_shape([1])
        return images, labels

    def tf_dataset(x, y, batch=8):
        dataset = tf.data.Dataset.from_tensor_slices((x, y))
        dataset = dataset.shuffle(buffer_size=15000)
        dataset = dataset.map(preprocess)
        dataset = dataset.batch(batch)
        dataset = dataset.prefetch(2)
        return dataset


    train_data = tf_dataset(X_train.values, y_train.values)
    test_data = tf_dataset(X_test.values, y_test.values)


    for image, result in train_data.take(1):
        sample_image, sample_result = image, result
    for i in range(4):
        plt.figure(figsize=(6, 6))
        plt.title("Image")
        plt.imshow(tf.keras.preprocessing.image.array_to_img(sample_image[i]))
        plt.show()
        plt.close()
        print(sample_result[i])
```

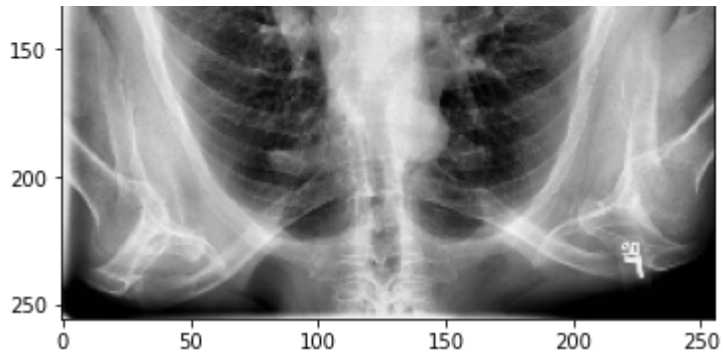



```
tf.Tensor([0], shape=(1,), dtype=int64)
```

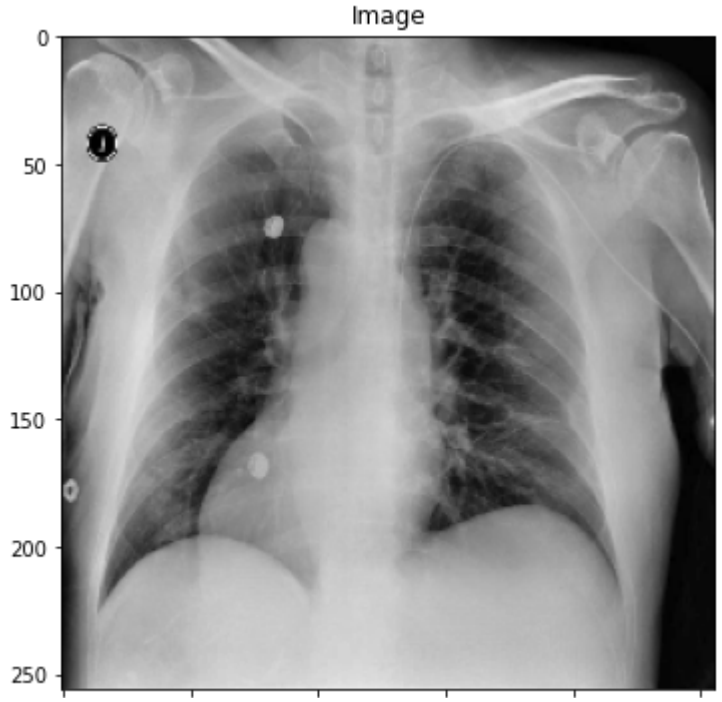



```
tf.Tensor([0], shape=(1,), dtype=int64)
```






```
tf.Tensor([0], shape=(1,), dtype=int64)
```




```
X_test.head(2)
```

```
2699    siim/dicom-images-train-jpg/2700.jpg
2124    siim/dicom-images-train-jpg/2125.jpg
Name: images_paths, dtype: object
```

```
class History_Callback(tf.keras.callbacks.Callback):

    def on_epoch_end(self, epoch, logs={}):

        val_predict = []
        for i in range(len(X_test)):
            image = tf.keras.preprocessing.image.load_img(X_test.iloc[i])
            image = tf.keras.preprocessing.image.img_to_array(image, dtype='float32')
            image = tf.image.resize(image, [img_size, img_size])
            image = image / 255.0
            image = exposure.equalize_adapthist(image)
            image = tf.expand_dims(image, axis = 0)
            predict = self.model.predict(image)
            val_predict.append(np.squeeze(predict, axis = -1).round())

        val_targ = y_test
```

```
            f1_score = metrics.f1_score(val_targ, val_predict)
            auc = metrics.roc_auc_score(val_targ, val_predict)
            recall = metrics.recall_score(val_targ, val_predict)

            print("recall:  {recall} - f1Score: {f1score} - AUC: {AUC}".format(f1score = f1_score


    history_Model1 = History_Callback()


    class Resnet34(tf.keras.Model):
        def __init__(self, input_shape, layer_params):
            super().__init__()

            self.conv1_resnet = tf.keras.layers.Conv2D(filters=32,
                                                       kernel_size=(7, 7),
                                                       strides=2,
                                                       padding="same")
            self.bn1_resnet = tf.keras.layers.BatchNormalization()
            self.pool1_resnet = tf.keras.layers.MaxPool2D(pool_size=(3, 3),
                                                          strides=2,
                                                          padding="same")

            self.layer1_resnet = make_basic_block_layer(filter_num=32,
                                                        blocks=layer_params[0])
            self.layer2_resnet = make_basic_block_layer(filter_num=64,
                                                        blocks=layer_params[1],
                                                        stride=2)
            self.layer3_resnet = make_basic_block_layer(filter_num=128,
                                                        blocks=layer_params[2],
                                                        stride=2)
            self.layer4_resnet = make_basic_block_layer(filter_num=256,
                                                        blocks=layer_params[3],
                                                        stride=2)

            self.avgpool_resnet = tf.keras.layers.AveragePooling2D()
            self.flatten = tf.keras.layers.Flatten()

            self.dense1 = tf.keras.layers.Dense(300, activation = 'relu')
            self.dense2 = tf.keras.layers.Dense(50, activation = 'relu')
            self.final = tf.keras.layers.Dense(1, activation= 'sigmoid')


        def call(self, inputs, training=None, mask=None):
            x = self.conv1_resnet(inputs)
            x = self.bn1_resnet(x, training=training)
            x = tf.nn.relu(x)
            x = self.pool1_resnet(x)
            x = self.layer1_resnet(x, training=training)
            x = self.layer2_resnet(x, training=training)
            x = self.layer3_resnet(x, training=training)
            x = self.layer4_resnet(x, training=training)
```

```
        x = self.layer4_resnet(x, training=training)
        x = self.avgpool_resnet(x)
        x = self.flatten(x)
        x = self.dense1(x)
        x = self.dense2(x)
        output = self.final(x)
        return output


model3 = Resnet34((256, 256, 3),[3, 4, 6, 3])


callbacks = [
    tf.keras.callbacks.ModelCheckpoint('./best_model3.h5', save_weights_only=True, save_best_
                                       mode='min'),
             history_Model1
]
optimizer = tf.keras.optimizers.Adam(0.001)


model3.compile(optimizer, loss = 'binary_crossentropy' , metrics=['accuracy'])


history3 = model3.fit(train_data, steps_per_epoch=len(train_data), epochs=25,\
                              validation_data=test_data,callbacks=callbacks, )
```

```
Epoch 1/25
1360/1360 [==============================] - ETA: 0s - loss: 0.7193 - accuracy: 0.753
1360/1360 [==============================] - 636s 464ms/step - loss: 0.7192 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 2/25
1360/1360 [==============================] - 622s 457ms/step - loss: 0.5321 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 3/25
1360/1360 [==============================] - 628s 462ms/step - loss: 0.5216 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 4/25
1360/1360 [==============================] - 623s 458ms/step - loss: 0.5238 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 5/25
1360/1360 [==============================] - 614s 452ms/step - loss: 0.5141 - accuracy
recall:  0.02247191011235955 - f1Score: 0.0410958904109589 - AUC: 0.5011510293661586
Epoch 6/25
1360/1360 [==============================] - 613s 451ms/step - loss: 0.5160 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 7/25
1360/1360 [==============================] - 642s 472ms/step - loss: 0.5149 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 8/25
1360/1360 [==============================] - 640s 471ms/step - loss: 0.5180 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 9/25
1360/1360 [==============================] - 648s 477ms/step - loss: 0.4952 - accuracy
recall:  0.5543071161048689 - f1Score: 0.33789954337899547 - AUC: 0.5324614136787614
Epoch 10/25
1360/1360 [==============================] - 652s 479ms/step - loss: 0.4936 - accuracy
```

```
recall:  0.0449438202247191 - f1Score: 0.07894736842105263 - AUC: 0.5092022710465421
Epoch 11/25
1360/1360 [==============================] - 643s 473ms/step - loss: 0.4767 - accuracy
recall:  1.0 - f1Score: 0.3617886178861789 - AUC: 0.5
Epoch 12/25
1360/1360 [==============================] - 640s 471ms/step - loss: 0.4812 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.4994692144373673
Epoch 13/25
1360/1360 [==============================] - 641s 472ms/step - loss: 0.4668 - accuracy
recall:  0.08239700374531835 - f1Score: 0.1423948220064725 - AUC: 0.5305827906200052
Epoch 14/25
1360/1360 [==============================] - 647s 476ms/step - loss: 0.4446 - accuracy
recall:  0.6292134831460674 - f1Score: 0.4132841328413284 - AUC: 0.6139697988978745
Epoch 15/25
1360/1360 [==============================] - 643s 473ms/step - loss: 0.4382 - accuracy
recall:  0.0 - f1Score: 0.0 - AUC: 0.5
Epoch 16/25
1360/1360 [==============================] - 646s 475ms/step - loss: 0.4338 - accuracy
recall:  0.11610486891385768 - f1Score: 0.1962025316455696 - AUC: 0.5484982943295403
Epoch 17/25
1360/1360 [==============================] - 647s 476ms/step - loss: 0.4111 - accuracy
recall:  0.19101123595505617 - f1Score: 0.2982456140350877 - AUC: 0.5827667644743434
Epoch 18/25
1360/1360 [==============================] - 639s 470ms/step - loss: 0.4198 - accuracy
recall:  0.25842696629213485 - f1Score: 0.375 - AUC: 0.6122283451418211
Epoch 19/25
1360/1360 [==============================] - 647s 476ms/step - loss: 0.4114 - accuracy
recall:  0.449438202247191 - f1Score: 0.4519774011299435 - AUC: 0.6482859801044873
```

```
model3.summary()
```

```
Model: "resnet34_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_612 (Conv2D)          multiple                  4736
_________________________________________________________________
batch_normalization_612 (Bat multiple                  128
_________________________________________________________________
max_pooling2d_17 (MaxPooling multiple                  0
_________________________________________________________________
sequential_119 (Sequential)  (None, 64, 64, 32)        56256
_________________________________________________________________
sequential_120 (Sequential)  (None, 32, 32, 64)        281408
_________________________________________________________________
sequential_122 (Sequential)  (None, 16, 16, 128)       1712256
_________________________________________________________________
sequential_124 (Sequential)  (None, 8, 8, 256)         3285760
_________________________________________________________________
average_pooling2d_17 (Averag multiple                  0
_________________________________________________________________
flatten_17 (Flatten)         multiple                  0
_________________________________________________________________
dense_51 (Dense)             multiple                  1229100
_________________________________________________________________
dense_52 (Dense)             multiple                  15050
```

```
_________________________________________________________________
dense_53 (Dense)             multiple                  51        
=================================================================
Total params: 6,584,745
Trainable params: 6,576,233
Non-trainable params: 8,512
_________________________________________________________________
```

```
# Plot training & validation iou_score values
plt.figure(figsize=(30, 5))
plt.subplot(121)
plt.plot(history3.history['loss'])
plt.plot(history3.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')

# Plot training & validation loss values
plt.subplot(122)
plt.plot(history3.history['accuracy'])
plt.plot(history3.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```