Name: Anik Manik

Email address: iamanik4@gmail.com

Contact number: 9477672426

Anydesk address: 400 728 410

Years of Work Experience: 2.6 years

Date: 24th Jan 2021

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import datetime as dt
from datetime import datetime
from tqdm import tqdm
from glob import glob
import pandas as pd
import shutil
import glob2
```

In [ ]:
```
from tensorflow.keras import models, layers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import BatchNormalization, Activation, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import *
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
import datetime
from sklearn.model_selection import train_test_split
```

In [ ]:
```
# install libraries to read dicom images
!pip install -q tensorflow-io
!pip install pydicom
```

```
     |████████████████████████████████| 25.3MB 129kB/s
Collecting pydicom
  Downloading https://files.pythonhosted.org/packages/f4/15/df16546bc59bfca390cf072d473fb2c8acd423163
6f64356593a63137e55/pydicom-2.1.2-py3-none-any.whl (1.9MB)
     |████████████████████████████████| 1.9MB 8.1MB/s
Installing collected packages: pydicom
Successfully installed pydicom-2.1.2
```

In [ ]:
```
import pydicom as dicom
import tensorflow as tf
import tensorflow_io as tfio
```

In [ ]:
```
# moung google drive
from google.colab import drive
drive.mount('gdrive',force_remount=True)
```

```
Mounted at gdrive
```

Download dataset from Kaggle

https://www.kaggle.com/seesee/siim-train-test (https://www.kaggle.com/seesee/siim-train-test)

In [ ]:
```
# download the dataset from kaggle
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win6
4; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36" --header="Accept: te
xt/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applica
tion/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9" --header="Referer: http
s://www.kaggle.com/" "https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archive.zi
p?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccou
nt.com%2F20210317%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210317T142453Z&X-Goog-Expires=259199&
X-Goog-SignedHeaders=host&X-Goog-Signature=a442032ce439bae6f0d8ff2b9dba327817f23b7d6f26540be17c80d2bc7
3641b64d3736438a3a8070a9507ef3312b580745cc5088975a22f232baa0767c5a1b3f6323351a61da90939e3408f7c00001fd
15817d90dbff914a6633228febebd968d3e34f034aadf2f9d10127ff992b36ca0598c2b785782771da63443827668f3cb1d5e1
0296f75ad2b95a0d10af0e2d2d37e54edfe2489e195dd4cf30de4ce78e4e55505d3a92b0bc10ba658a2b084a7cb8ff44756212
f5691a92d97602027a98b9c6c092390f63b6286b2b4ac1d45b16284e2e9ae74522de68a78f7c99ee71f07f57449034f0dbfae0
3ad9db6233a0dfe53292b787b9a0538a146e7e1c9b93f" -c -O 'archive.zip'
```

```
--2021-03-19 03:02:12--  https://storage.googleapis.com/kaggle-data-sets/245622/651264/bundle/archiv
e.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gservice
account.com%2F20210317%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210317T142453Z&X-Goog-Expires=2
59199&X-Goog-SignedHeaders=host&X-Goog-Signature=a442032ce439bae6f0d8ff2b9dba327817f23b7d6f26540be17c
80d2bc73641b64d3736438a3a8070a9507ef3312b580745cc5088975a22f232baa0767c5a1b3f6323351a61da90939e3408f7
c00001fd15817d90dbff914a6633228febebd968d3e34f034aadf2f9d10127ff992b36ca0598c2b785782771da63443827668
f3cb1d5e10296f75ad2b95a0d10af0e2d2d37e54edfe2489e195dd4cf30de4ce78e4e55505d3a92b0bc10ba658a2b084a7cb8
ff44756212f5691a92d97602027a98b9c6c092390f63b6286b2b4ac1d45b16284e2e9ae74522de68a78f7c99ee71f07f57449
034f0dbfae03ad9db6233a0dfe53292b787b9a0538a146e7e1c9b93f
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.142.128, 74.125.195.128, 74.125.2
0.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.142.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2059765561 (1.9G) [application/zip]
Saving to: ‘archive.zip’

archive.zip         100%[===================>]   1.92G  95.5MB/s    in 22s

2021-03-19 03:02:34 (89.0 MB/s) - ‘archive.zip’ saved [2059765561/2059765561]
```

In [ ]:
```
# unzip the dataset
!unzip -qq 'archive.zip'
```

In [ ]:
```
# read the given train csv file
image_df = pd.read_csv('siim/train-rle.csv')
image_df.head()
```

Out[ ]:

	ImageId	EncodedPixels
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...

In [ ]:
```
# check the properties of the image dataframe
image_df.describe()
```

Out[ ]:

	ImageId	EncodedPixels
count	12954	12954
unique	12047	3577
top	1.2.276.0.7230010.3.1.4.8323329.1851.151787516...	-1
freq	10	9378

Out of 12954 image ids 12047 are unique. I need to drop the duplicate images.

In [ ]:
```
# drop the duplicate ImageIDs
image_df.drop_duplicates(subset ="ImageId", keep = 'first', inplace = True)
```

Move all the images in a single directory

In [ ]:
```
# create a directory for dicom images
images_dicom = 'siim/images_dicom/'
if not os.path.isdir(images_dicom):
    os.makedirs(images_dicom)

# move all train dicom images from 'dicom-images-train' to 'images_dicom' in a single directory
existing_path = 'siim/dicom-images-train/'
dicom_list = glob2.glob(os.path.join(existing_path, '**/*.dcm'))
for filename in tqdm(dicom_list):
  shutil.move(str(filename), images_dicom)
```

```
100%|██████████| 12089/12089 [00:00<00:00, 24788.20it/s]
```

Add the new image path to image_df

In [ ]:
```
# add full dicom path to image_df
image_df['dicom_path'] = images_dicom + image_df['ImageId']+'.dcm'
image_df.head()
```

Out[ ]:

	ImageId	EncodedPixels	dicom_path
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...

Add a new column in the dataframe "is_pneumothorax" to indicate if is has pneumothorax

In [ ]:
```
# remove extra space in EncodedPixels column
image_df.rename(columns = {' EncodedPixels':'EncodedPixels'}, inplace = True)

# add a column whether the image is with pneumothorax or without pneumothorax
# if EncodedPixels value is not "-1" then the image is with pneumothorax
image_df['is_pneumothorax'] = np.where(image_df['EncodedPixels']=='-1', 0, 1)
image_df.head()
```

Out[ ]:

	ImageId	EncodedPixels	dicom_path
0	1.2.276.0.7230010.3.1.4.8323329.6904.151787520...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
1	1.2.276.0.7230010.3.1.4.8323329.13666.15178752...	557374 2 1015 8 1009 14 1002 20 997 26 990 32 ...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
2	1.2.276.0.7230010.3.1.4.8323329.11028.15178752...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
3	1.2.276.0.7230010.3.1.4.8323329.10366.15178752...	514175 10 1008 29 994 30 993 32 991 33 990 34 ...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...
4	1.2.276.0.7230010.3.1.4.8323329.10016.15178752...	592184 33 976 58 956 73 941 88 926 102 917 109...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8323...

Split the dataset into train and validation

In [ ]:
```
from sklearn.model_selection import train_test_split
train_df, val_df = train_test_split(image_df, test_size=0.2, random_state=42, stratify=image_df['is_pn
eumothorax'], shuffle=True)
```

Define a function to read and decode dicom image

In [ ]:
```
# Define a function to read and decode dicom image
def decode_image(image_path, label, size=256):
  # read the image from image_path
  image = tf.io.read_file(image_path)
  # convert the image into a 3D tensor
  image = tfio.image.decode_dicom_image(image, dtype=tf.uint8,color_dim=True,scale='preserve')
  # convert image datatype to float32
  image = tf.image.convert_image_dtype(image, tf.float32)
  # squeeze the image from shape (1,1024,1024,1) to (1024,1024,1)
  image =tf.squeeze(image,[0])
  # using tf.tile convert image shape (1024,1024,1) tp (1024,1024,3)
  image=tf.tile(image, tf.constant([1,1,3], tf.int32))
  # resize the image
  image=tf.image.resize(image,size=[size,size])
  # return image and corresponding label
  return image, label
```

Define a function to augment image

```
In [ ]: # Define a function to augment image
def augment_image(image, label):
  a = np.random.uniform()
  if a<0.2:
      image = tf.image.random_flip_left_right(image)
  elif a<0.4:
      image = tf.image.random_flip_up_down(image)
  elif a<0.6:
      image = tf.image.random_brightness(image, 0.3)
  elif a<0.8:
      image = tf.image.random_contrast(image,lower=0.2,upper=0.3)
  else:
      image = tf.image.random_saturation(image, lower=2, upper=5)
  return image, label
```

Define train_generator and val_generator using tf.data

```
In [ ]: AUTOTUNE = tf.data.experimental.AUTOTUNE
def train_generator(image_path, image_label):
  # creating a dataset from tensor slices
  dataset = tf.data.Dataset.from_tensor_slices((image_path, image_label))
  # shuffle the dataset
  dataset = dataset.shuffle(len(image_path), seed=42)
  # decode image using decode_image function
  dataset = dataset.map(decode_image, num_parallel_calls=AUTOTUNE)
  # augment image using augment_image function
  dataset = dataset.map(augment_image, num_parallel_calls=AUTOTUNE)
  return dataset

def val_generator(image_path, image_label):
  # creating a dataset from tensor slices
  dataset = tf.data.Dataset.from_tensor_slices((image_path, image_label))
  # shuffle the dataset
  dataset = dataset.shuffle(len(image_path), seed=42)
  # decode image using decode_image function
  dataset = dataset.map(decode_image, num_parallel_calls=AUTOTUNE)
  return dataset
```

```
In [ ]: # separate image path and image label from train_df and val_df
train_image_path = train_df['dicom_path'].values
train_image_label = train_df['is_pneumothorax'].values
val_image_path = val_df['dicom_path'].values
val_image_label = val_df['is_pneumothorax'].values
```

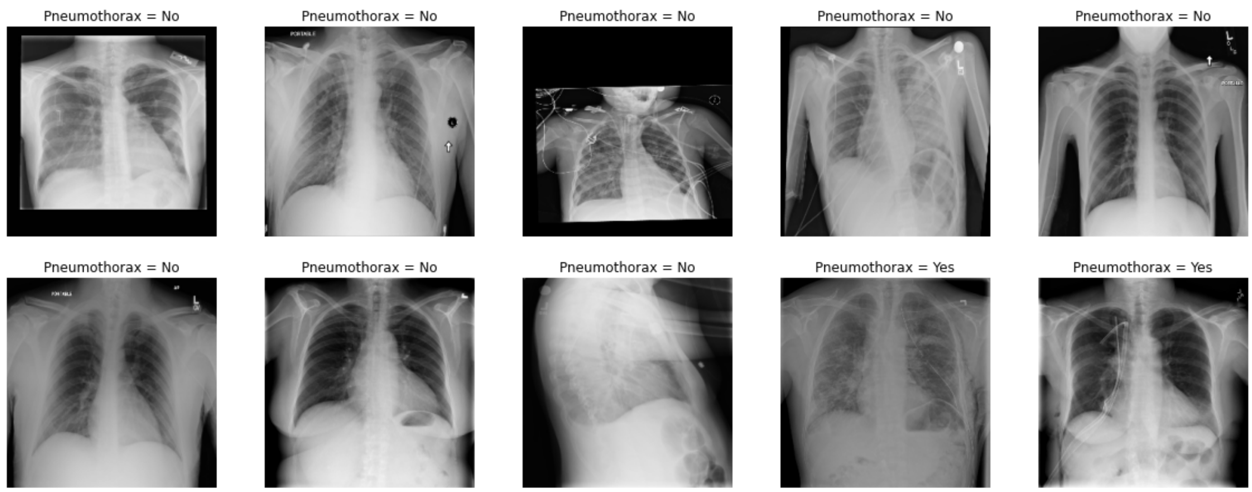
```
In [ ]: train_dataset = train_generator(train_image_path, train_image_label)
val_dataset  = val_generator(val_image_path, val_image_label)
train_dataset, val_dataset

Out[ ]: (<ParallelMapDataset shapes: ((256, 256, None), ()), types: (tf.float32, tf.int64)>,
 <ParallelMapDataset shapes: ((256, 256, None), ()), types: (tf.float32, tf.int64)>)
```

```
In [ ]: # batch the train and validation dataset
batch_size = 64
train_ds_batch = train_dataset.batch(batch_size, drop_remainder=True)
val_ds_batch = val_dataset.batch(batch_size, drop_remainder=True)
train_ds_batch, val_ds_batch

Out[ ]: (<BatchDataset shapes: ((64, 256, 256, None), (64,)), types: (tf.float32, tf.int64)>,
 <BatchDataset shapes: ((64, 256, 256, None), (64,)), types: (tf.float32, tf.int64)>)
```

In [ ]:

```
# plot some random images from train_dataset
plt.figure(figsize=(20,20))
count=0
for i,j in tqdm(train_dataset.take(10)):
  ax = plt.subplot(5,5,count+1)
  count=count+1
  if j==0:
    # if the image label=0, then print "Pneumothorax = No"
    plt.title("Pneumothorax = No")
  else:
    # otherwise(image label=1) print "Pneumothorax = Yes"
    plt.title("Pneumothorax = Yes")
  plt.imshow(i)
  plt.axis("off")
```

```
100%|██████████| 10/10 [00:00<00:00, 12.49it/s]
```



01. Create model for classification using pretrained VGG16 model

In [ ]:

```
# https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Model, load_model
from keras.applications.vgg16 import VGG16
tf.keras.backend.clear_session()
# use VGG16 model with imagenet weights
model_vgg16 = VGG16(weights = "imagenet", include_top=False, input_shape = (256,256,3))
# set all the layers trainable = False
for i in tqdm(model_vgg16.layers):
  i.trainable=False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weight
s_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 2s 0us/step

100%|██████████| 19/19 [00:00<00:00, 5988.26it/s]
```

In [ ]:

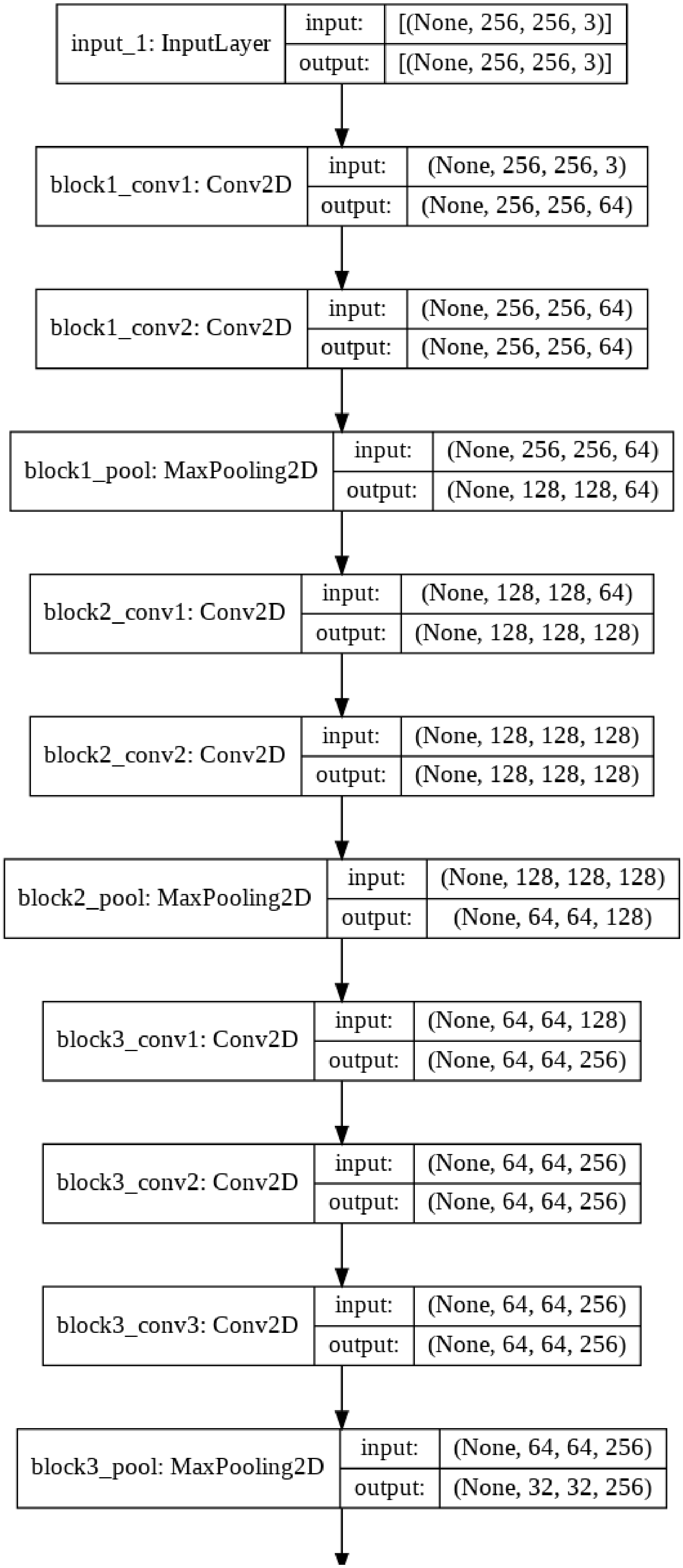
```
# define model architecture
model = model_vgg16.output
model = Conv2D(32, (3, 3))(model)
model = (Activation('relu'))(model)
model = (MaxPool2D(pool_size=(2, 2)))(model)
model = Flatten()(model)
model = Dense(256, activation="relu")(model)
# model= tf.keras.layers.Dropout(0.2)(model)
model = Dense(128, activation="relu")(model)
output_layer = Dense(1, activation="sigmoid")(model)
model_1 = Model(model_vgg16.input,output_layer)
```

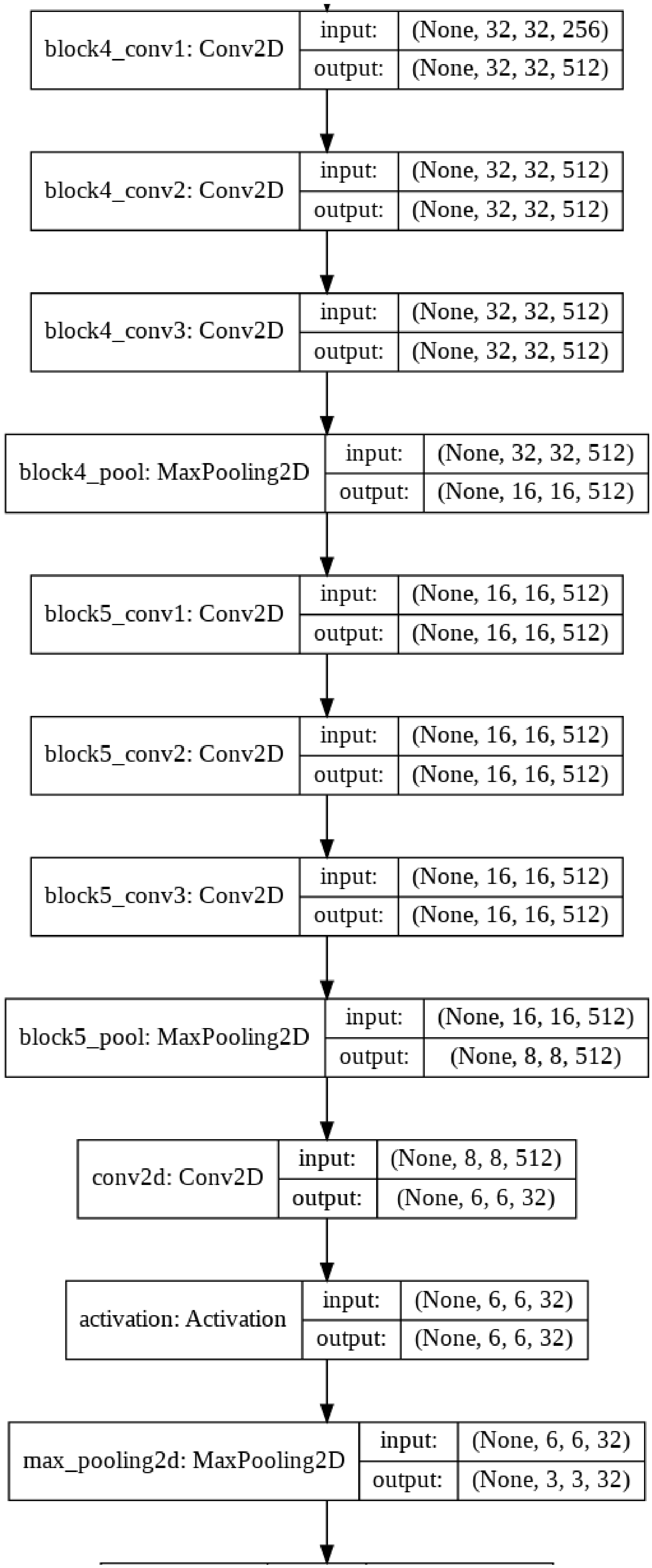
In [ ]:
```
# print model summary
model_1.summary()
```

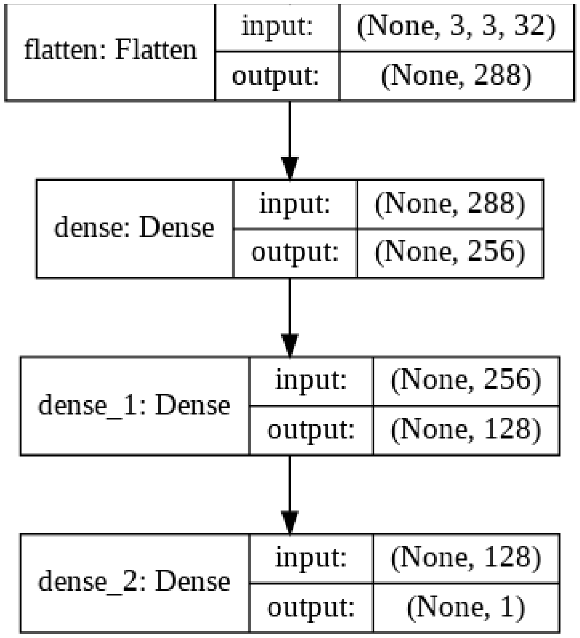
```
Model: "model"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)        [(None, 256, 256, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0
_________________________________________________________________
conv2d (Conv2D)             (None, 6, 6, 32)          147488
_________________________________________________________________
activation (Activation)     (None, 6, 6, 32)          0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 3, 32)         0
_________________________________________________________________
flatten (Flatten)           (None, 288)               0
_________________________________________________________________
dense (Dense)               (None, 256)               73984
_________________________________________________________________
dense_1 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)             (None, 1)                 129
=================================================================
Total params: 14,969,185
Trainable params: 254,497
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [ ]:
```
from tensorflow.keras.utils import plot_model
plot_model(model_1, 'model_1.png', show_shapes=True)
```

Out[ ]:


input_1: InputLayer
input:
[(None, 256, 256, 3)]
output:
[(None, 256, 256, 3)]
block1_conv1: Conv2D
input:
(None, 256, 256, 3)
output:
(None, 256, 256, 64)
block1_conv2: Conv2D
input:
(None, 256, 256, 64)
output:
(None, 256, 256, 64)
block1_pool: MaxPooling2D
input:
(None, 256, 256, 64)
output:
(None, 128, 128, 64)
block2_conv1: Conv2D
input:
(None, 128, 128, 64)
output:
(None, 128, 128, 128)
block2_conv2: Conv2D
input:
(None, 128, 128, 128)
output:
(None, 128, 128, 128)
block2_pool: MaxPooling2D
input:
(None, 128, 128, 128)
output:
(None, 64, 64, 128)
block3_conv1: Conv2D
input:
(None, 64, 64, 128)
output:
(None, 64, 64, 256)
block3_conv2: Conv2D
input:
(None, 64, 64, 256)
output:
(None, 64, 64, 256)
block3_conv3: Conv2D
input:
(None, 64, 64, 256)
output:
(None, 64, 64, 256)
block3_pool: MaxPooling2D
input:
(None, 64, 64, 256)
output:
(None, 32, 32, 256)


block4_conv1: Conv2D
input:
(None, 32, 32, 256)
output:
(None, 32, 32, 512)
block4_conv2: Conv2D
input:
(None, 32, 32, 512)
output:
(None, 32, 32, 512)
block4_conv3: Conv2D
input:
(None, 32, 32, 512)
output:
(None, 32, 32, 512)
block4_pool: MaxPooling2D
input:
(None, 32, 32, 512)
output:
(None, 16, 16, 512)
block5_conv1: Conv2D
input:
(None, 16, 16, 512)
output:
(None, 16, 16, 512)
block5_conv2: Conv2D
input:
(None, 16, 16, 512)
output:
(None, 16, 16, 512)
block5_conv3: Conv2D
input:
(None, 16, 16, 512)
output:
(None, 16, 16, 512)
block5_pool: MaxPooling2D
input:
(None, 16, 16, 512)
output:
(None, 8, 8, 512)
conv2d: Conv2D
input:
(None, 8, 8, 512)
output:
(None, 6, 6, 32)
activation: Activation
input:
(None, 6, 6, 32)
output:
(None, 6, 6, 32)
max_pooling2d: MaxPooling2D
input:
(None, 6, 6, 32)
output:
(None, 3, 3, 32)



Define callbacks to monitor the model performance

In [ ]:

```
# set filepath to save best models
filepath="gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/weights-{epoch:02d}-{val_rec
all:.4f}.hdf5"
model_checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_recall',  verbose=1, save_best_only
=True, mode='max')

# earlystop
from tensorflow.keras.callbacks import EarlyStopping
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.001, mode='max', patience=4, verbose=1,
restore_best_weights=True)
```

In [ ]:

```
# callback to stop training when desired recall is reached
# https://towardsdatascience.com/neural-network-with-tensorflow-how-to-stop-training-using-callback-5c
8d575c18a9
class myCallback(tf.keras.callbacks.Callback):
    def __init__(self, threshold):
      super(myCallback, self).__init__()
      self.threshold = threshold

    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_recall') > self.threshold):
          print("Trainning Stopped. Val Recall = {} crossed threshold = {}".format(logs.get('val_recal
l'), self.threshold))
          self.model.stop_training = True
```

Train model_1

In [ ]:

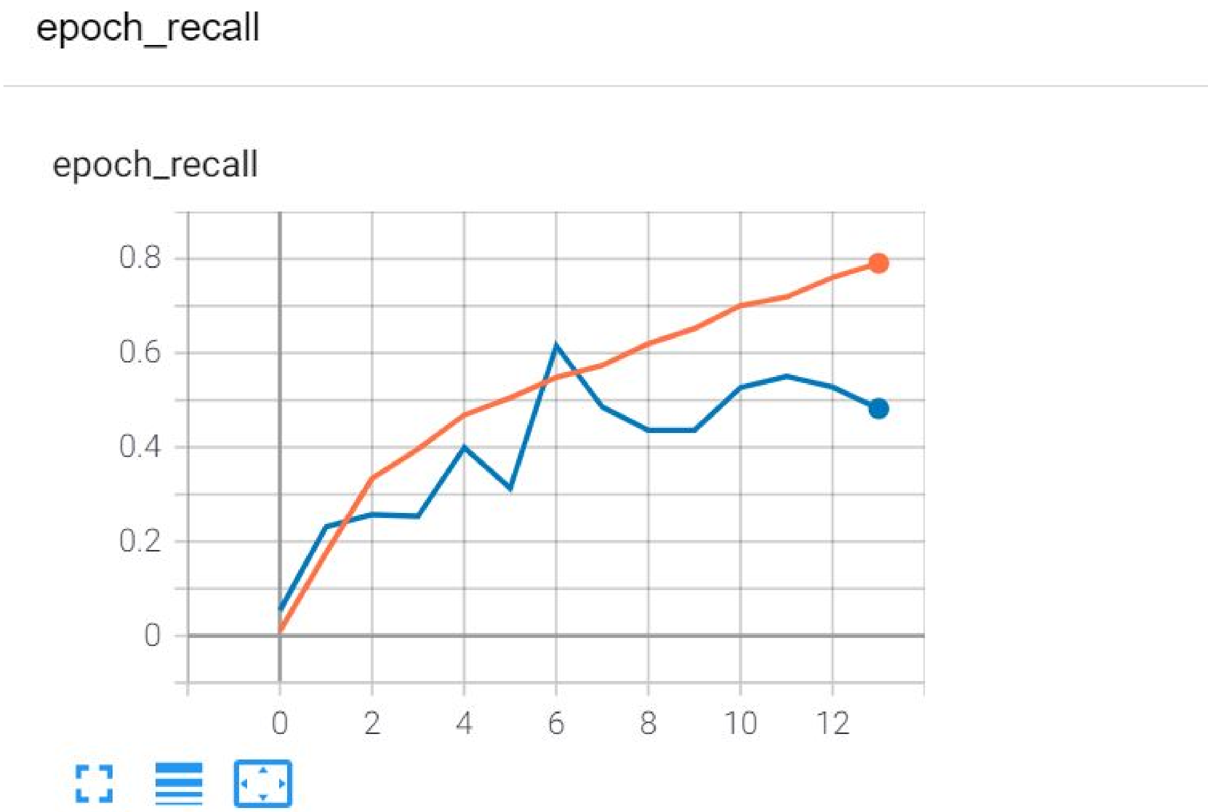
```
%rm -rf ./log
%load_ext tensorboard
%tensorboard --logdir='./log'
%reload_ext tensorboard
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir='./log')
adam = tf.keras.optimizers.Adam(lr=0.0001)
precision = tf.keras.metrics.Precision(name='precision')
recall = tf.keras.metrics.Recall(name='recall')
callback_list = [model_checkpoint, myCallback(threshold=0.99),tensorboard_callback]
model_1.compile(loss = "binary_crossentropy", optimizer=adam, metrics=["accuracy", recall, precision])
model_1.fit(train_ds_batch, epochs=20,verbose=1,validation_data=val_ds_batch,batch_size=64,callbacks=c
allback_list)
```

```
Epoch 1/20
150/150 [==============================] - 475s 3s/step - loss: 0.5308 - accuracy: 0.7685 - recall:
0.0379 - precision: 0.3349 - val_loss: 0.4614 - val_accuracy: 0.7855 - val_recall: 0.0534 - val_preci
sion: 0.7000

Epoch 00001: val_recall improved from -inf to 0.05344, saving model to gdrive/My Drive/Colab Notebook
s/cs2_pneumothorax/classification/weights-01-0.0534.hdf5
Epoch 2/20
150/150 [==============================] - 445s 3s/step - loss: 0.4417 - accuracy: 0.7891 - recall:
0.1236 - precision: 0.6561 - val_loss: 0.4172 - val_accuracy: 0.8074 - val_recall: 0.2314 - val_preci
sion: 0.6914

Epoch 00002: val_recall improved from 0.05344 to 0.23136, saving model to gdrive/My Drive/Colab Noteb
ooks/cs2_pneumothorax/classification/weights-02-0.2314.hdf5
Epoch 3/20
150/150 [==============================] - 441s 3s/step - loss: 0.4080 - accuracy: 0.8109 - recall:
0.3071 - precision: 0.6681 - val_loss: 0.4063 - val_accuracy: 0.8053 - val_recall: 0.2571 - val_preci
sion: 0.6553

Epoch 00003: val_recall improved from 0.23136 to 0.25714, saving model to gdrive/My Drive/Colab Noteb
ooks/cs2_pneumothorax/classification/weights-03-0.2571.hdf5
Epoch 4/20
150/150 [==============================] - 441s 3s/step - loss: 0.3811 - accuracy: 0.8258 - recall:
0.3631 - precision: 0.7003 - val_loss: 0.4124 - val_accuracy: 0.8117 - val_recall: 0.2538 - val_preci
sion: 0.7204

Epoch 00004: val_recall did not improve from 0.25714
Epoch 5/20
150/150 [==============================] - 444s 3s/step - loss: 0.3547 - accuracy: 0.8427 - recall:
0.4591 - precision: 0.7186 - val_loss: 0.3868 - val_accuracy: 0.8247 - val_recall: 0.3992 - val_preci
sion: 0.6710

Epoch 00005: val_recall improved from 0.25714 to 0.39923, saving model to gdrive/My Drive/Colab Noteb
ooks/cs2_pneumothorax/classification/weights-05-0.3992.hdf5
Epoch 6/20
150/150 [==============================] - 448s 3s/step - loss: 0.3529 - accuracy: 0.8441 - recall:
0.5064 - precision: 0.7449 - val_loss: 0.4056 - val_accuracy: 0.8218 - val_recall: 0.3131 - val_preci
sion: 0.7333

Epoch 00006: val_recall did not improve from 0.39923
Epoch 7/20
150/150 [==============================] - 442s 3s/step - loss: 0.3282 - accuracy: 0.8571 - recall:
0.5442 - precision: 0.7548 - val_loss: 0.3968 - val_accuracy: 0.8197 - val_recall: 0.6152 - val_preci
sion: 0.5894

Epoch 00007: val_recall improved from 0.39923 to 0.61524, saving model to gdrive/My Drive/Colab Noteb
ooks/cs2_pneumothorax/classification/weights-07-0.6152.hdf5
Epoch 8/20
150/150 [==============================] - 441s 3s/step - loss: 0.3049 - accuracy: 0.8690 - recall:
0.5767 - precision: 0.7828 - val_loss: 0.3850 - val_accuracy: 0.8307 - val_recall: 0.4856 - val_preci
sion: 0.6554

Epoch 00008: val_recall did not improve from 0.61524
Epoch 9/20
150/150 [==============================] - 443s 3s/step - loss: 0.2827 - accuracy: 0.8847 - recall:
0.6098 - precision: 0.8132 - val_loss: 0.3916 - val_accuracy: 0.8336 - val_recall: 0.4359 - val_preci
sion: 0.6972

Epoch 00009: val_recall did not improve from 0.61524
Epoch 10/20
150/150 [==============================] - 441s 3s/step - loss: 0.2746 - accuracy: 0.8842 - recall:
0.6384 - precision: 0.8047 - val_loss: 0.3975 - val_accuracy: 0.8349 - val_recall: 0.4364 - val_preci
sion: 0.7099

Epoch 00010: val_recall did not improve from 0.61524
Epoch 11/20
150/150 [==============================] - 440s 3s/step - loss: 0.2382 - accuracy: 0.9031 - recall:
0.6988 - precision: 0.8382 - val_loss: 0.3929 - val_accuracy: 0.8336 - val_recall: 0.5269 - val_preci
sion: 0.6493

Epoch 00011: val_recall did not improve from 0.61524
```

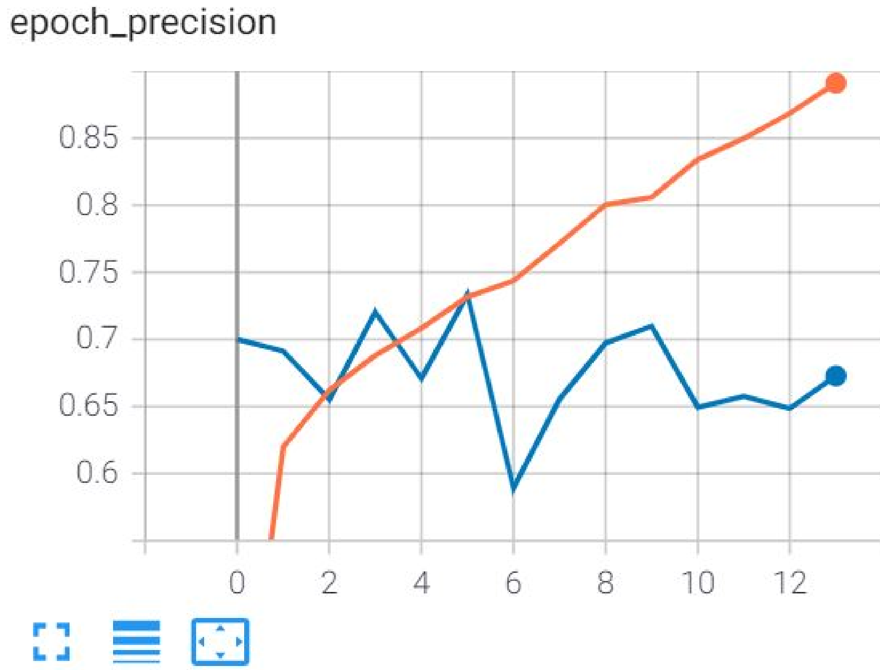
```
Epoch 12/20
150/150 [==============================] - 441s 3s/step - loss: 0.2280 - accuracy: 0.9110 - recall:
0.7155 - precision: 0.8500 - val_loss: 0.3969 - val_accuracy: 0.8361 - val_recall: 0.5503 - val_preci
sion: 0.6576

Epoch 00012: val_recall did not improve from 0.61524
Epoch 13/20
150/150 [==============================] - 438s 3s/step - loss: 0.1982 - accuracy: 0.9230 - recall:
0.7514 - precision: 0.8705 - val_loss: 0.4078 - val_accuracy: 0.8332 - val_recall: 0.5278 - val_preci
sion: 0.6486

Epoch 00013: val_recall did not improve from 0.61524
Epoch 14/20
150/150 [==============================] - 439s 3s/step - loss: 0.1746 - accuracy: 0.9406 - recall:
0.8127 - precision: 0.9131 - val_loss: 0.4323 - val_accuracy: 0.8332 - val_recall: 0.4819 - val_preci
sion: 0.6729

Epoch 00014: val_recall did not improve from 0.61524
Epoch 15/20
150/150 [==============================] - 438s 3s/step - loss: 0.1572 - accuracy: 0.9465 - recall:
0.8267 - precision: 0.9281 - val_loss: 0.4445 - val_accuracy: 0.8336 - val_recall: 0.4905 - val_preci
sion: 0.6745

Epoch 00015: val_recall did not improve from 0.61524
Epoch 16/20
150/150 [==============================] - 439s 3s/step - loss: 0.1464 - accuracy: 0.9508 - recall:
0.8517 - precision: 0.9264 - val_loss: 0.4455 - val_accuracy: 0.8281 - val_recall: 0.5798 - val_preci
sion: 0.6212

Epoch 00016: val_recall did not improve from 0.61524
Epoch 17/20
150/150 [==============================] - 436s 3s/step - loss: 0.1165 - accuracy: 0.9628 - recall:
0.8816 - precision: 0.9457 - val_loss: 0.4737 - val_accuracy: 0.8247 - val_recall: 0.5190 - val_preci
sion: 0.6276

Epoch 00017: val_recall did not improve from 0.61524
Epoch 18/20
150/150 [==============================] - 438s 3s/step - loss: 0.0981 - accuracy: 0.9693 - recall:
0.9029 - precision: 0.9550 - val_loss: 0.4906 - val_accuracy: 0.8264 - val_recall: 0.5173 - val_preci
sion: 0.6270

Epoch 00018: val_recall did not improve from 0.61524
Epoch 19/20
150/150 [==============================] - 435s 3s/step - loss: 0.0962 - accuracy: 0.9727 - recall:
0.9172 - precision: 0.9580 - val_loss: 0.5069 - val_accuracy: 0.8193 - val_recall: 0.5886 - val_preci
sion: 0.5931

Epoch 00019: val_recall did not improve from 0.61524
Epoch 20/20
150/150 [==============================] - 434s 3s/step - loss: 0.0720 - accuracy: 0.9823 - recall:
0.9458 - precision: 0.9726 - val_loss: 0.5372 - val_accuracy: 0.8235 - val_recall: 0.5066 - val_preci
sion: 0.6282

Epoch 00020: val_recall did not improve from 0.61524
```

```
Out[ ]: <tensorflow.python.keras.callbacks.History at 0x7fc289e897d0>
```

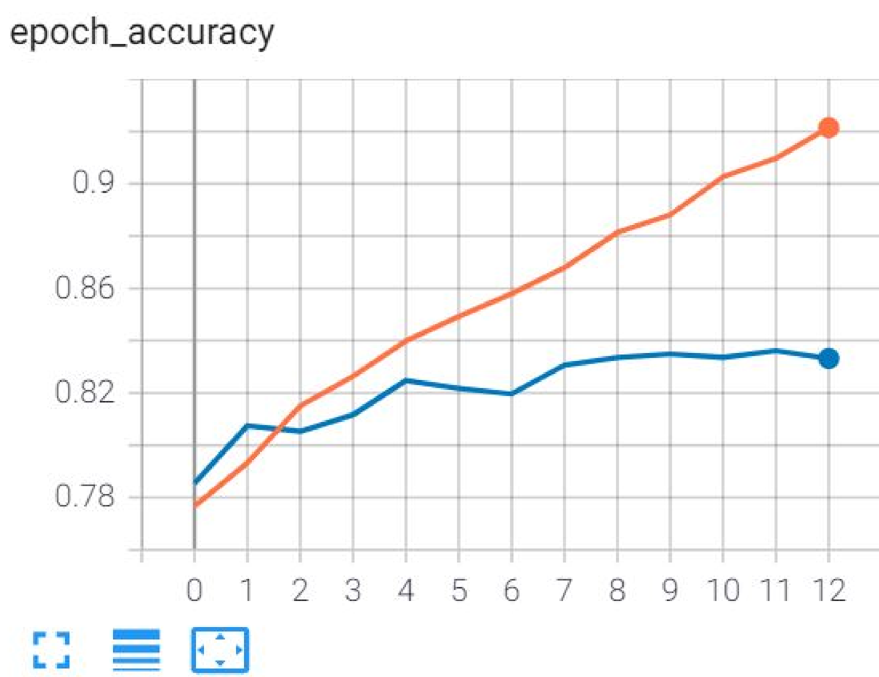
Model 1: Scalars



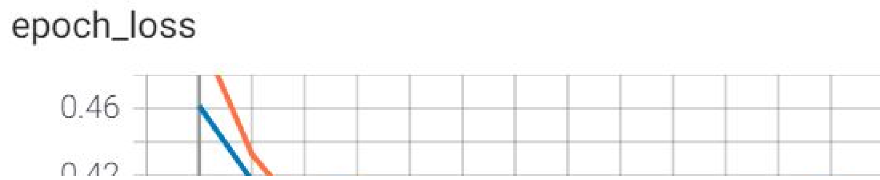
epoch_precision



epoch_accuracy



epoch_loss



Load best weights to model_1

In [ ]:
```
# Load best weights to model_1
from keras.models import load_model
model_1 = load_model("gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/weights-07-0.615
2.hdf5")
```

Define a function to plot confusion matrix

In [39]:
```
from sklearn.metrics import confusion_matrix
# define function to plot confusion matrix
def conf_matrix(test_y,predict_y):
  labels = [0,1]
  plt.figure(figsize=(8,6))
  C = confusion_matrix(test_y, predict_y)
  sns.heatmap(C, annot=True, fmt='d')
  plt.xlabel('Predicted Class')
  plt.ylabel('Original Class')
  plt.title('Confusion matrix')
  plt.show()
```

In [ ]:
```
# classify the images based on the probability score
# initialize a empty list to store the predicted label
val_pred_list = []
for file in tqdm(val_image_path):
  # preprocess the image
  size = 256
  image = tf.io.read_file(file)
  image = tfio.image.decode_dicom_image(image, dtype=tf.uint8,color_dim=True,scale='preserve')
  image = tf.image.convert_image_dtype(image, tf.float32)
  image =tf.squeeze(image,[0])
  image=tf.tile(image, tf.constant([1,1,3], tf.int32))
  image=tf.image.resize(image,size=[size,size])
  image = tf.expand_dims(image,axis=0)
  # predict the probability score of the image
  pred = model_1.predict(image)
  # if the probabiliy score is greater than 0.5 then give class label=1 else 0
  if pred[0]>0.5:
    val_pred_list.append(1)
  else:
    val_pred_list.append(0)
```

```
100%|██████████| 2410/2410 [02:44<00:00, 14.64it/s]
```

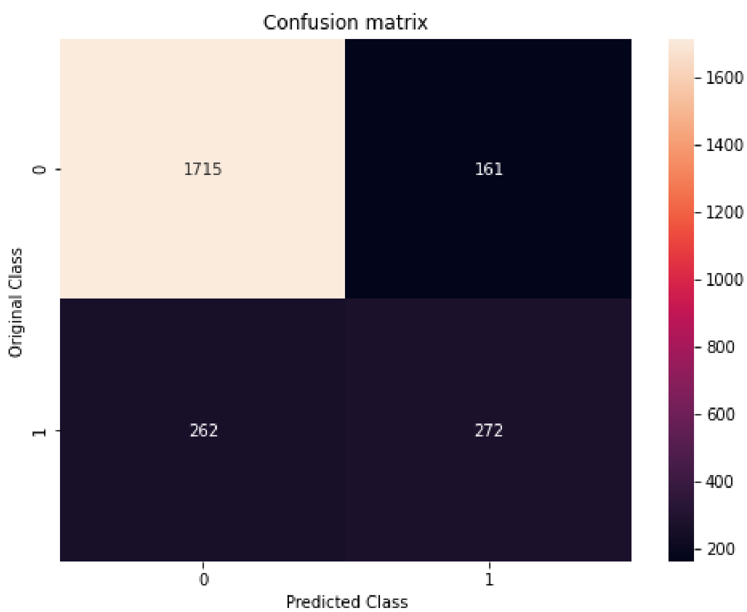
In [ ]:
```python
# add a new column in val_df dataframe with the predicted classes
val_df['pred_label'] = val_pred_list
val_df.head()
```

Out[ ]:

	ImageId	EncodedPixels	dicom_
10812	1.2.276.0.7230010.3.1.4.8323329.11636.15178752...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
7110	1.2.276.0.7230010.3.1.4.8323329.4471.151787518...	278724 1 1020 6 1016 9 1014 11 1011 13 1010 13...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
5130	1.2.276.0.7230010.3.1.4.8323329.5233.151787518...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
5131	1.2.276.0.7230010.3.1.4.8323329.11260.15178752...	611609 30 992 33 989 36 987 40 982 44 978 49 9...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
5297	1.2.276.0.7230010.3.1.4.8323329.14511.15178752...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:

In [ ]:
```python
# store the dataframe in csv format for future use
val_df.to_csv('gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/val_df_vgg16.csv', inde
x=False)
```

In [ ]:
```python
# plot confusion matrix
conf_matrix(val_df['is_pneumothorax'], val_df['pred_label'])
```



recall: 0.5442 val_recall: 0.6152 precision: 0.7548 val_precision: 0.5894

In [5]:
```
#reference :https://stackoverflow.com/questions/8356501/python-format-tabular-output
from beautifultable import BeautifulTable
table = BeautifulTable()a
table.column_headers= ["Model", "Train Recall", "Validation Recall", "Train Precision", "Validation Pr
ecision"]
table.append_row(["Model_1 \n VGG16", "0.5442", "0.6152", "0.7548", "0.5894"])
print(table)
```

```
+----------+-------------+----------------+--------------+--------------------+
|  Model   | Train Recall | Validation Rec | Train Precis | Validation Precisi |
|          |              |      all       |     ion      |         on         |
+----------+-------------+----------------+--------------+--------------------+
| Model_1  |    0.544     |     0.615      |    0.755     |       0.589        |
|   VGG16  |              |                |              |                    |
+----------+-------------+----------------+--------------+--------------------+

C:\Anaconda3\lib\site-packages\beautifultable\utils.py:113: FutureWarning: 'BeautifulTable.column_hea
ders' has been deprecated in 'v1.0.0' and will be removed in 'v1.2.0'. Use 'BTColumnCollection.heade
r' instead.
  warnings.warn(message, FutureWarning)
C:\Anaconda3\lib\site-packages\beautifultable\utils.py:113: FutureWarning: 'BeautifulTable.append_ro
w' has been deprecated in 'v1.0.0' and will be removed in 'v1.2.0'. Use 'BTRowCollection.append' inst
ead.
  warnings.warn(message, FutureWarning)
```

02. Create model for classification using pretrained VGG19 model

In [ ]:
```
from keras.applications.vgg19 import VGG19
tf.keras.backend.clear_session()
# use VGG19 model with imagenet weights
model_vgg19 = VGG19(weights = "imagenet", include_top=False, input_shape = (256,256,3))
# set all the layers trainable = False
for i in tqdm(model_vgg19.layers):
  i.trainable=False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weight
s_tf_dim_ordering_tf_kernels_notop.h5
80142336/80134624 [==============================] - 1s 0us/step

100%|██████████| 22/22 [00:00<00:00, 7124.36it/s]
```

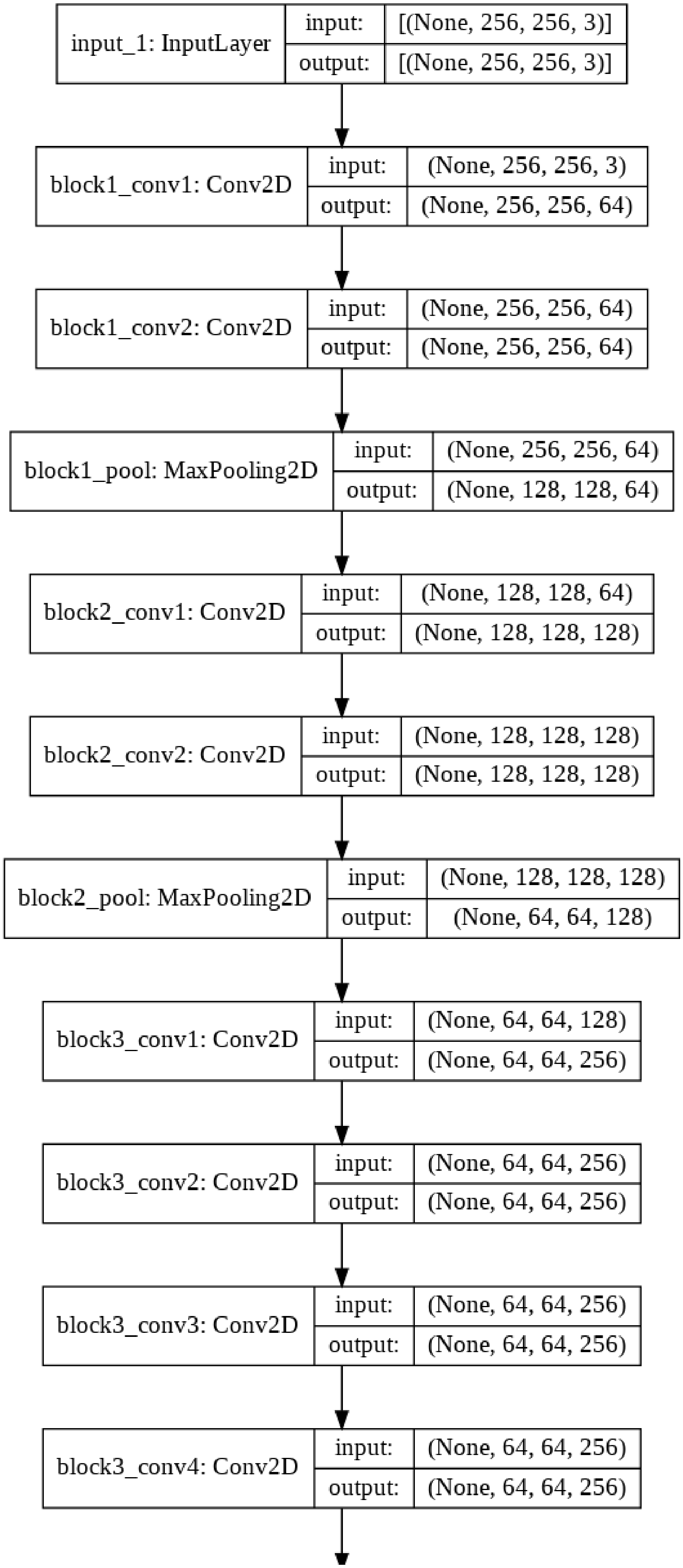
In [ ]:
```
# define model architecture
model = model_vgg19.output
model = Conv2D(32, (3, 3))(model)
model = (Activation('relu'))(model)
model = (MaxPool2D(pool_size=(2, 2)))(model)
model = Flatten()(model)
model = Dense(256, activation="relu")(model)
# model= tf.keras.layers.Dropout(0.2)(model)
model = Dense(128, activation="relu")(model)
output_layer = Dense(1, activation="sigmoid")(model)
model_2 = Model(model_vgg19.input,output_layer)
```

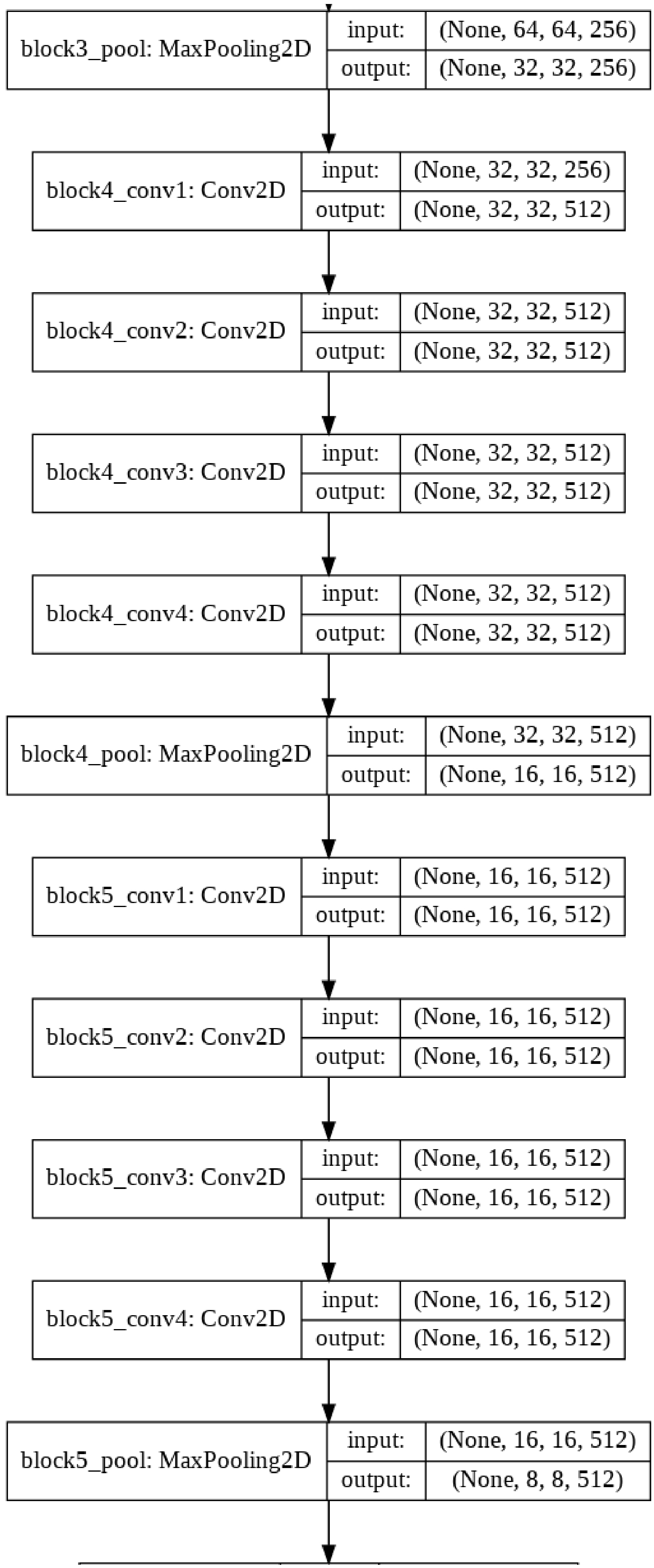
In [ ]:
```python
# print model summary
model_2.summary()
```

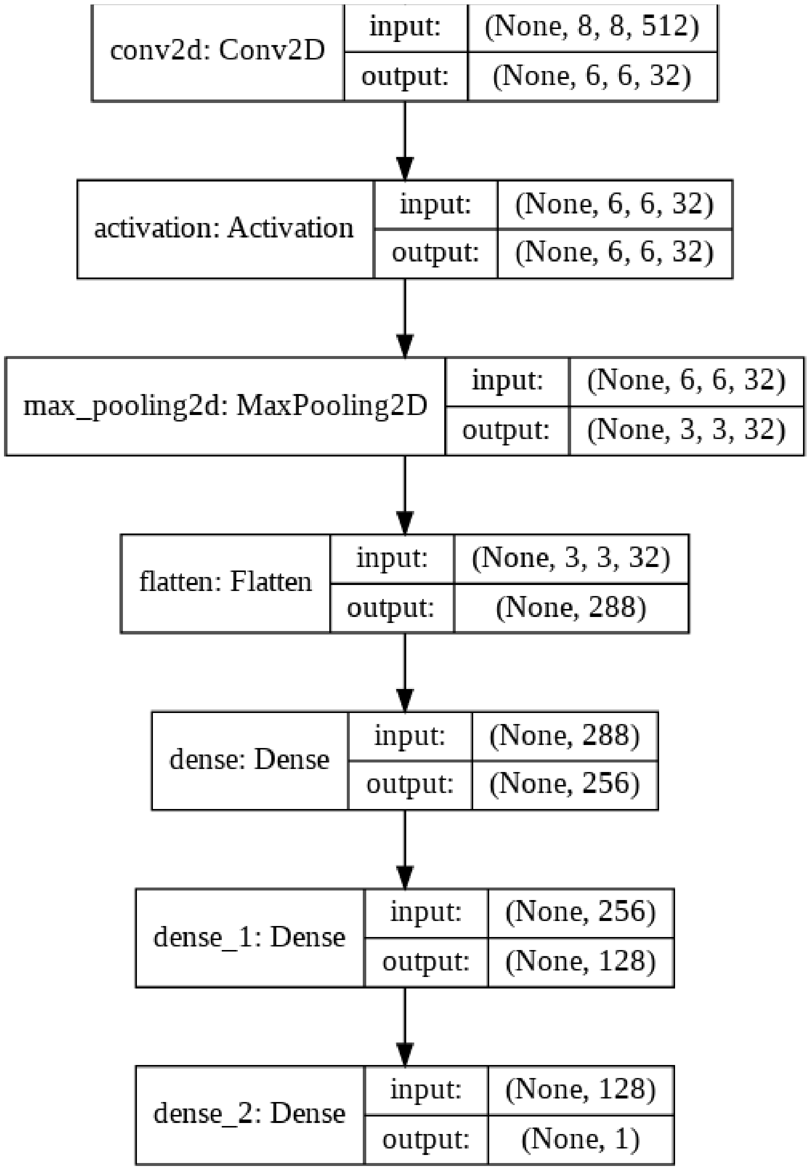
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 256, 256, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 256, 256, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 128, 128, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 128, 128, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 128, 128, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 64, 64, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 64, 64, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 64, 64, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 64, 64, 256)       590080    
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 64, 64, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 32, 32, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 32, 32, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 32, 32, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 32, 32, 512)       2359808   
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 32, 32, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 16, 16, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 16, 16, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 8, 8, 512)         0         
_________________________________________________________________
conv2d (Conv2D)              (None, 6, 6, 32)          147488    
_________________________________________________________________
activation (Activation)      (None, 6, 6, 32)          0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 3, 32)          0         
_________________________________________________________________
flatten (Flatten)            (None, 288)               0         
_________________________________________________________________
dense (Dense)                (None, 256)               73984     
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896     
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129       
=================================================================
Total params: 20,278,881
Trainable params: 254,497
Non-trainable params: 20,024,384
_________________________________________________________________
```

In [ ]:
```
from tensorflow.keras.utils import plot_model
plot_model(model_2, 'model_2.png', show_shapes=True)
```

Out[ ]:


input_1: InputLayer
input:
[(None, 256, 256, 3)]
output:
[(None, 256, 256, 3)]
block1_conv1: Conv2D
input:
(None, 256, 256, 3)
output:
(None, 256, 256, 64)
block1_conv2: Conv2D
input:
(None, 256, 256, 64)
output:
(None, 256, 256, 64)
block1_pool: MaxPooling2D
input:
(None, 256, 256, 64)
output:
(None, 128, 128, 64)
block2_conv1: Conv2D
input:
(None, 128, 128, 64)
output:
(None, 128, 128, 128)
block2_conv2: Conv2D
input:
(None, 128, 128, 128)
output:
(None, 128, 128, 128)
block2_pool: MaxPooling2D
input:
(None, 128, 128, 128)
output:
(None, 64, 64, 128)
block3_conv1: Conv2D
input:
(None, 64, 64, 128)
output:
(None, 64, 64, 256)
block3_conv2: Conv2D
input:
(None, 64, 64, 256)
output:
(None, 64, 64, 256)
block3_conv3: Conv2D
input:
(None, 64, 64, 256)
output:
(None, 64, 64, 256)
block3_conv4: Conv2D
input:
(None, 64, 64, 256)
output:
(None, 64, 64, 256)


block3_pool: MaxPooling2D
input: (None, 64, 64, 256)
output: (None, 32, 32, 256)
block4_conv1: Conv2D
input: (None, 32, 32, 256)
output: (None, 32, 32, 512)
block4_conv2: Conv2D
input: (None, 32, 32, 512)
output: (None, 32, 32, 512)
block4_conv3: Conv2D
input: (None, 32, 32, 512)
output: (None, 32, 32, 512)
block4_conv4: Conv2D
input: (None, 32, 32, 512)
output: (None, 32, 32, 512)
block4_pool: MaxPooling2D
input: (None, 32, 32, 512)
output: (None, 16, 16, 512)
block5_conv1: Conv2D
input: (None, 16, 16, 512)
output: (None, 16, 16, 512)
block5_conv2: Conv2D
input: (None, 16, 16, 512)
output: (None, 16, 16, 512)
block5_conv3: Conv2D
input: (None, 16, 16, 512)
output: (None, 16, 16, 512)
block5_conv4: Conv2D
input: (None, 16, 16, 512)
output: (None, 16, 16, 512)
block5_pool: MaxPooling2D
input: (None, 16, 16, 512)
output: (None, 8, 8, 512)



Train model_2

In [29]:

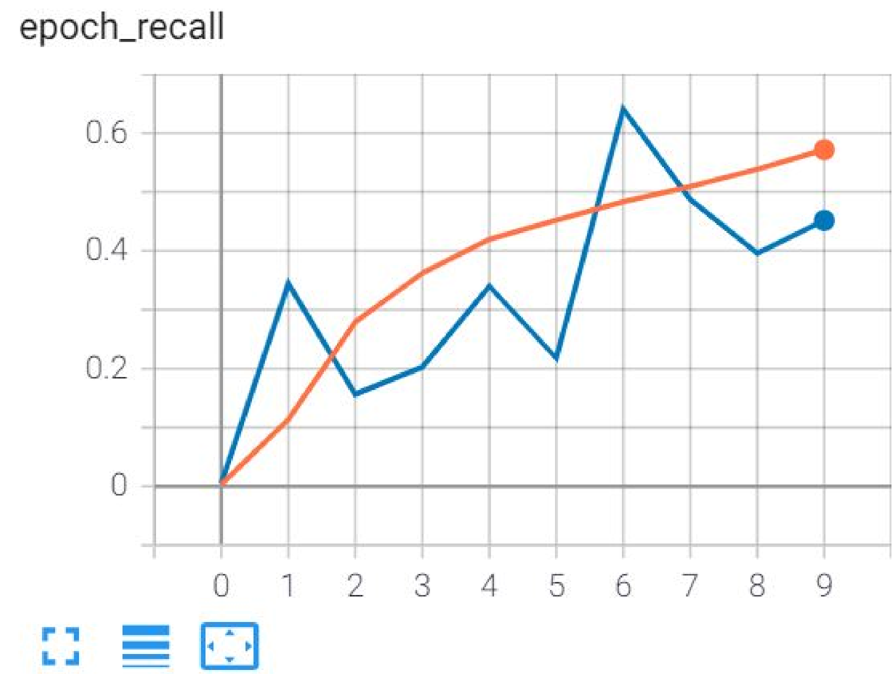
```
%rm -rf ./log
%load_ext tensorboard
%tensorboard --logdir='./log'
%reload_ext tensorboard
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir='./log')
adam = tf.keras.optimizers.Adam(lr=0.0001)
precision = tf.keras.metrics.Precision(name='precision')
recall = tf.keras.metrics.Recall(name='recall')
callback_list = [model_checkpoint, myCallback(threshold=0.99),tensorboard_callback] #earlystop,
model_2.compile(loss = "binary_crossentropy", optimizer=adam, metrics=["accuracy", recall, precision])
model_2.fit(train_ds_batch, epochs=10,verbose=1,validation_data=val_ds_batch,batch_size=64,callbacks=c
allback_list)
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard

Reusing TensorBoard on port 6006 (pid 320), started 0:00:54 ago. (Use '!kill 320' to kill it.)

Epoch 1/10
150/150 [==============================] - 491s 3s/step - loss: 0.5302 - accuracy: 0.7746 - recall:
0.0166 - precision: 0.3684 - val_loss: 0.4828 - val_accuracy: 0.7791 - val_recall: 0.0057 - val_preci
sion: 0.6000

Epoch 00001: val_recall improved from -inf to 0.00573, saving model to gdrive/My Drive/Colab Notebook
s/cs2_pneumothorax/classification/weights-01-0.0057.hdf5
Epoch 2/10
150/150 [==============================] - 452s 3s/step - loss: 0.4639 - accuracy: 0.7823 - recall:
0.0594 - precision: 0.7059 - val_loss: 0.4331 - val_accuracy: 0.8083 - val_recall: 0.3442 - val_preci
sion: 0.6186

Epoch 00002: val_recall improved from 0.00573 to 0.34417, saving model to gdrive/My Drive/Colab Noteb
ooks/cs2_pneumothorax/classification/weights-02-0.3442.hdf5
Epoch 3/10
150/150 [==============================] - 451s 3s/step - loss: 0.4255 - accuracy: 0.8055 - recall:
0.2674 - precision: 0.6630 - val_loss: 0.4279 - val_accuracy: 0.7960 - val_recall: 0.1562 - val_preci
sion: 0.6721

Epoch 00003: val_recall did not improve from 0.34417
Epoch 4/10
150/150 [==============================] - 445s 3s/step - loss: 0.4002 - accuracy: 0.8227 - recall:
0.3507 - precision: 0.6885 - val_loss: 0.4161 - val_accuracy: 0.8007 - val_recall: 0.2027 - val_preci
sion: 0.6772

Epoch 00004: val_recall did not improve from 0.34417
Epoch 5/10
150/150 [==============================] - 442s 3s/step - loss: 0.3764 - accuracy: 0.8348 - recall:
0.4093 - precision: 0.7109 - val_loss: 0.3960 - val_accuracy: 0.8222 - val_recall: 0.3397 - val_preci
sion: 0.6969

Epoch 00005: val_recall did not improve from 0.34417
Epoch 6/10
150/150 [==============================] - 441s 3s/step - loss: 0.3739 - accuracy: 0.8336 - recall:
0.4510 - precision: 0.7351 - val_loss: 0.4268 - val_accuracy: 0.8053 - val_recall: 0.2182 - val_preci
sion: 0.7012

Epoch 00006: val_recall did not improve from 0.34417
Epoch 7/10
150/150 [==============================] - 440s 3s/step - loss: 0.3527 - accuracy: 0.8437 - recall:
0.4903 - precision: 0.7308 - val_loss: 0.4230 - val_accuracy: 0.8041 - val_recall: 0.6400 - val_preci
sion: 0.5499

Epoch 00007: val_recall improved from 0.34417 to 0.64000, saving model to gdrive/My Drive/Colab Noteb
ooks/cs2_pneumothorax/classification/weights-07-0.6400.hdf5
Epoch 8/10
150/150 [==============================] - 442s 3s/step - loss: 0.3377 - accuracy: 0.8550 - recall:
0.5098 - precision: 0.7672 - val_loss: 0.3907 - val_accuracy: 0.8218 - val_recall: 0.4875 - val_preci
sion: 0.6210

Epoch 00008: val_recall did not improve from 0.64000
Epoch 9/10
150/150 [==============================] - 441s 3s/step - loss: 0.3220 - accuracy: 0.8597 - recall:
0.5257 - precision: 0.7526 - val_loss: 0.3933 - val_accuracy: 0.8231 - val_recall: 0.3958 - val_preci
sion: 0.6677

Epoch 00009: val_recall did not improve from 0.64000
Epoch 10/10
150/150 [==============================] - 440s 3s/step - loss: 0.3124 - accuracy: 0.8606 - recall:
0.5535 - precision: 0.7587 - val_loss: 0.3934 - val_accuracy: 0.8226 - val_recall: 0.4516 - val_preci
sion: 0.6450

Epoch 00010: val_recall did not improve from 0.64000
```

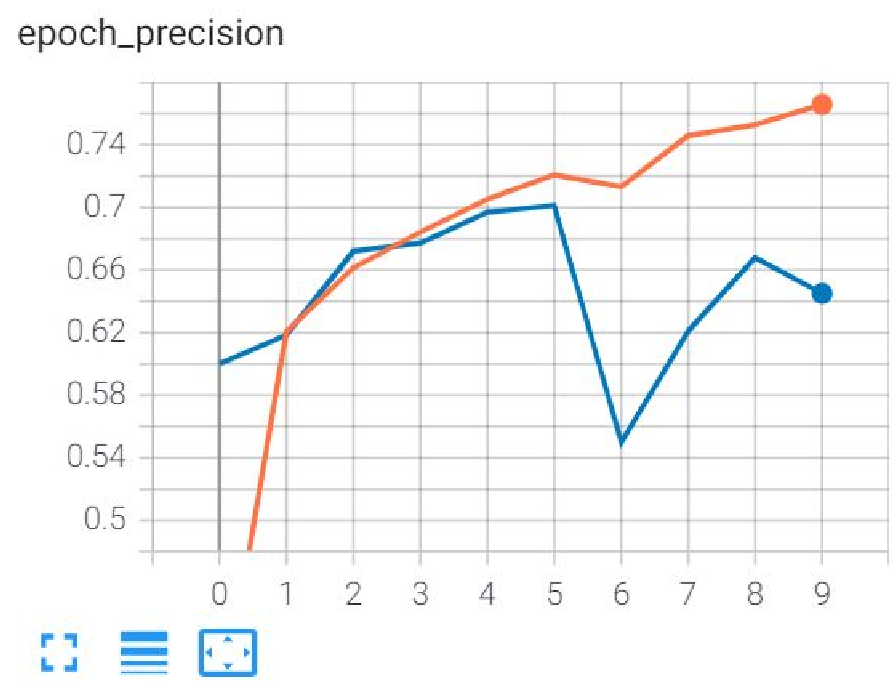
```
Out[29]: <tensorflow.python.keras.callbacks.History at 0x7fd7f62de2d0>
```

Model 2: Scalars

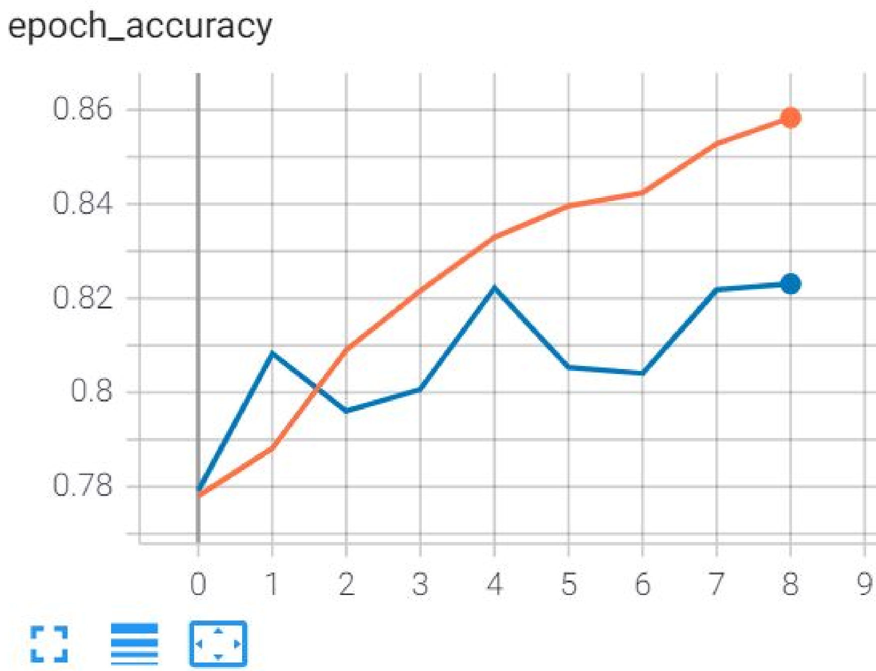
epoch_recall



epoch_precision



epoch_accuracy



epoch_loss

In [34]:

```
# Load best weights to model_2
from keras.models import load_model
model_2 = load_model("gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/weights-07-0.640
0.hdf5")
```

In [35]:

```
# classify the images based on the probability score
# initialize a empty list to store the predicted label
val_pred_list = []
for file in tqdm(val_image_path):
  # preprocess the image
  size = 256
  image = tf.io.read_file(file)
  image = tfio.image.decode_dicom_image(image, dtype=tf.uint8,color_dim=True,scale='preserve')
  image = tf.image.convert_image_dtype(image, tf.float32)
  image =tf.squeeze(image,[0])
  image=tf.tile(image, tf.constant([1,1,3], tf.int32))
  image=tf.image.resize(image,size=[size,size])
  image = tf.expand_dims(image,axis=0)
  # predict the probability score of the image
  pred = model_2.predict(image)
  # if the probabiliy score is greater than 0.5 then give class label=1 else 0
  if pred[0]>0.5:
    val_pred_list.append(1)
  else:
    val_pred_list.append(0)
```

```
100%|██████████| 2410/2410 [03:22<00:00, 11.90it/s]
```

In [36]:

```
# add a new column in val_df dataframe with the predicted classes
val_df['pred_label'] = val_pred_list
val_df.head()
```

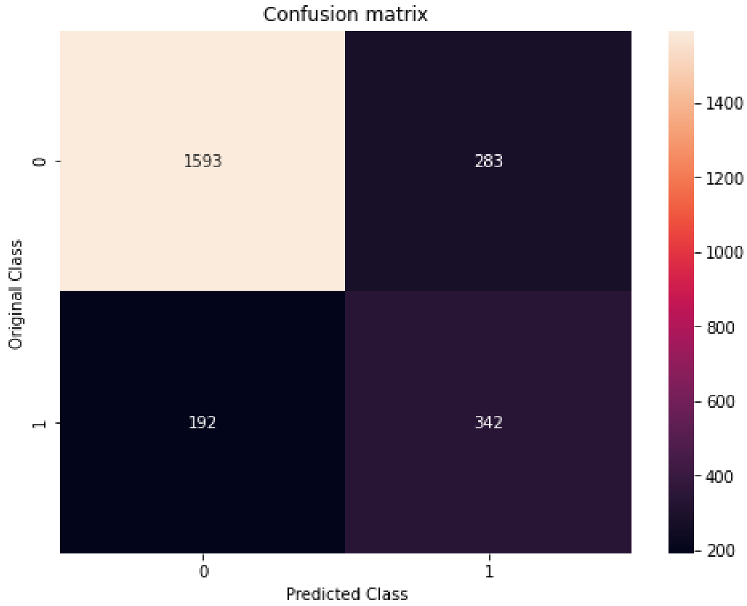
Out[36]:

	ImageId	EncodedPixels	dicom_
10812	1.2.276.0.7230010.3.1.4.8323329.11636.15178752...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
7110	1.2.276.0.7230010.3.1.4.8323329.4471.151787518...	278724 1 1020 6 1016 9 1014 11 1011 13 1010 13...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
5130	1.2.276.0.7230010.3.1.4.8323329.5233.151787518...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
5131	1.2.276.0.7230010.3.1.4.8323329.11260.15178752...	611609 30 992 33 989 36 987 40 982 44 978 49 9...	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:
5297	1.2.276.0.7230010.3.1.4.8323329.14511.15178752...	-1	siim/images_dicom/1.2.276.0.7230010.3.1.4.8:

In [37]:

```
# store the dataframe in csv format for future use
val_df.to_csv('gdrive/My Drive/Colab Notebooks/cs2_pneumothorax/classification/val_df_vgg19.csv', inde
x=False)
```

In [40]:
```
# plot confusion matrix
conf_matrix(val_df['is_pneumothorax'], val_df['pred_label'])
```



recall: 0.4903 val_recall: 0.6400 precision: 0.7308 val_precision: 0.5499

In [6]:
```
table.append_row(["Model_2 \n VGG19", "0.4903", "0.6400", "0.7308", "0.5499"])
print(table)
```

```
+----------+-------------+---------------+-------------+-------------------+
|  Model   | Train Recall | Validation Rec | Train Precis | Validation Precisi |
|          |             |      all      |     ion     |        on         |
+----------+-------------+---------------+-------------+-------------------+
| Model_1  |    0.544    |     0.615     |    0.755    |       0.589       |
|   VGG16  |             |               |             |                   |
+----------+-------------+---------------+-------------+-------------------+
| Model_2  |     0.49    |      0.64     |    0.731    |        0.55       |
|   VGG19  |             |               |             |                   |
+----------+-------------+---------------+-------------+-------------------+

C:\Anaconda3\lib\site-packages\beautifultable\utils.py:113: FutureWarning: 'BeautifulTable.append_ro
w' has been deprecated in 'v1.0.0' and will be removed in 'v1.2.0'. Use 'BTRowCollection.append' inst
ead.
  warnings.warn(message, FutureWarning)
```

Conclusion:

I have taken "recall" as the metric to measure the performance of the model. Recall is defined as, out of total positive points how many of them are predicted correctly i.e. True Positive/ Total Positive. In medical domain recall is very important because if a positive patient is detected as negative this is more dangerous than a negative patient detected as positive.

1. For model_1, built using VGG16 backbone val_recall = 0.61 val_precision = 0.59
2. For model_2, built using VGG19 backbone val_recall = 0.64 val_precision = 0.55

Model_2 gives higher recall than model_1. Even though the precision of model_1 is higher, in terms of recall model_2 performs better for pneumothorax classification. I will use model_2 for final prediction.