Covid-19 CT scan identification project code

Installing missing modules

```
#@title Installing missing modules
#installing patool for unzipping & split-folder for splitting the folder into train,test,val
!pip install patool
!pip install split-folders
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting patool
  Downloading patool-1.12-py2.py3-none-any.whl (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.5/77.5 kB 4.9 MB/s eta 0:00:00
Installing collected packages: patool
Successfully installed patool-1.12
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

```
[2] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing modules

```
[3] #@title Importing modules
    import patoolib
    import cv2
    import glob
    import os
    import splitfolders as sf
```

```
[4] import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers
    from tensorflow.python.keras.layers import Dense, Flatten
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.optimizers import Adam
    from keras.callbacks import ModelCheckpoint, EarlyStopping, CSVLogger
    from keras.preprocessing.image import ImageDataGenerator
```

```
[61] import pandas as pd
     import numpy as np
     from keras.models import load_model
     import matplotlib.pyplot as plt
     from PIL import Image as imgshow
```

Unzipping files

```
[6]  #@title Unzipping files
     #set the path where the zip files of images are present
     input='/content/drive/MyDrive/CTscan/'
```

```
[7]  #unzipping folder containing CT scan images
     patoolib.extract_archive(input+"COVID.zip",outdir='/content/CTscan')
```

```
patool: Extracting /content/drive/MyDrive/CTscan/COVID.zip ...
patool: running /usr/bin/7z x -o/content/CTscan -- /content/drive/MyDrive/CTscan/COVID.zip
patool: ... /content/drive/MyDrive/CTscan/COVID.zip extracted to `/content/CTscan'.
'/content/CTscan'
```

```
[8]  patoolib.extract_archive(input+"nonCOVID.zip",outdir='/content/CTscan')
```

```
patool: Extracting /content/drive/MyDrive/CTscan/nonCOVID.zip ...
patool: running /usr/bin/7z x -o/content/CTscan -- /content/drive/MyDrive/CTscan/nonCOVID.zip
patool: ... /content/drive/MyDrive/CTscan/nonCOVID.zip extracted to `/content/CTscan'.
'/content/CTscan'
```

Resizing images

```
[9]  #@title Resizing images
     curr=os.getcwd()
```

```
[10] #Resizing images to 224x224 for Resnet
     inputfolder='/content/CTscan/COVID'
     length=len(inputfolder)

     path='/content/resize/covid'
     os.makedirs(path,mode=0o666,exist_ok=True)

     for img in glob.glob(inputfolder+'/*.*'):
         image=cv2.imread(img)
         imgRe=cv2.resize(image,(224,224))
         cv2.imwrite(path+img[length:],imgRe)
```

```
[11] inputfolder='/content/CTscan/nonCOVID'
     length=len(inputfolder)

     path='/content/resize/noncovid'
     os.makedirs(path,mode=0o666,exist_ok=True)

     for img in glob.glob(inputfolder+'/*.*'):
         image=cv2.imread(img)
         imgRe=cv2.resize(image,(224,224))
         cv2.imwrite(path+img[length:],imgRe)
```

```
[12] curr

     '/content'

[13] input_folder=curr+'/resize'
```

Splitting dataset

```
[14] #@title Splitting dataset
     #splitting folder into train,test,val
     sf.ratio(input_folder,output='/content/final',seed=50,ratio=(.7,.2,.1),group_prefix=None,move=False)

     Copying files: 2480 files [00:00, 2558.22 files/s]
```

Data preparation

```
#@title Data preparation
#Data agumentation
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=45,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,)

val_datagen = ImageDataGenerator(rescale=1./255)
```

```
[69] train_dir=curr+'/final/train'
     validation_dir=curr+'/final/val'
```

```
[17] #train & validation dataset generators
     train_generator = train_datagen.flow_from_directory(train_dir,
                                                         target_size=(224, 224),
                                                         batch_size=32,
                                                         class_mode='binary',
                                                        color_mode='grayscale',
                                                         shuffle=True,
                                                         seed=50)

     validation_generator = val_datagen.flow_from_directory(validation_dir,
                                                            target_size=(224, 224),
                                                            batch_size=32,
                                                            class_mode='binary',
                                                           color_mode='grayscale',
                                                            shuffle=True,
                                                            seed=50)
```

```
Found 1735 images belonging to 2 classes.
Found 495 images belonging to 2 classes.
```

Model building

```
[18] #@title Model building
     #defining modelcheckpoints & early stopping
     filepath='saved_models/weights-improvement--{epoch:02d}--{val_acc:.2f}.hdf5'
     checkpoint=ModelCheckpoint(filepath,monitor='val_acc',verbose=1,save_best_only=True,mode='max')
     early_stop=EarlyStopping(monitor='val_loss',patience=10,verbose=1)
     log_csv=CSVLogger('my_logs.csv',separator=',',append=False)
     callbackslist=[checkpoint,early_stop,log_csv]
```

```
[19] #Defining Resnet50 model
     resnet_model = Sequential()

     pretrained_model= tf.keras.applications.ResNet50(include_top=False,
                       input_shape=(224,224,1),
                       pooling=None,classes=2,
                       weights=None)
     for layer in pretrained_model.layers:
            layer.trainable=False
```

```
[20] #Adding layers to resent model
     resnet_model.add(pretrained_model)
     resnet_model.add(layers.Conv2D(32, 3, activation='relu',padding='same',
                                    input_shape=(224,224,1),data_format='channels_last'))
     resnet_model.add(layers.MaxPooling2D(pool_size=2, strides=2,padding='same'))
     resnet_model.add(layers.Conv2D(64, 3, activation='relu',padding='same'))
     resnet_model.add(layers.MaxPooling2D(pool_size=2, strides=2,padding='same'))
     resnet_model.add(layers.Conv2D(128, 3, activation='relu',padding='same'))
     resnet_model.add(layers.MaxPooling2D(pool_size=2, strides=2,padding='same'))
     resnet_model.add(layers.Dropout(0.3))
     resnet_model.add(layers.Conv2D(128, 3, activation='relu',padding='same'))
     resnet_model.add(layers.MaxPooling2D(pool_size=2, strides=2,padding='same'))
     resnet_model.add(layers.Flatten())
     resnet_model.add(layers.Dense(512, activation='relu'))
     resnet_model.add(layers.Dropout(0.5))
     resnet_model.add(layers.Dense(1, activation='sigmoid'))
```

```
#model summary
resnet_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23581440

 conv2d (Conv2D)             (None, 7, 7, 32)          589856

 max_pooling2d (MaxPooling2D  (None, 4, 4, 32)         0
 )

 conv2d_1 (Conv2D)           (None, 4, 4, 64)          18496

 max_pooling2d_1 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 2, 2, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 128)        0
 2D)

 dropout (Dropout)           (None, 1, 1, 128)         0

 conv2d_3 (Conv2D)           (None, 1, 1, 128)         147584

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 128)        0
 2D)

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 512)               66048

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 24,477,793
Trainable params: 896,353
Non-trainable params: 23,581,440
_________________________________________________________________
```

```
[22] #compiling model
     resnet_model.compile(loss='binary_crossentropy',
                  optimizer=Adam(learning_rate=0.0001),
                  metrics=['acc','Precision','Recall','AUC'])
```

Model fitting

```
[23] #@title Model fitting
     #model fitting
     batch_size=32
     history=resnet_model.fit(train_generator,
                        steps_per_epoch=1735//batch_size,
                        epochs=100,
                        validation_data=validation_generator,
                        validation_steps=495//batch_size,
                        callbacks=callbackslist)

     Epoch 1/100
```

```
Epoch 24/100
54/54 [==============================] - ETA: 0s - loss: 0.6010 - acc: 0.6682 - precision: 0.6310 - recall: 0.8017
Epoch 24: val_acc did not improve from 0.70833
54/54 [==============================] - 14s 256ms/step - loss: 0.6010 - acc: 0.6682 - precision: 0.6310 - recall:
Epoch 25/100
54/54 [==============================] - ETA: 0s - loss: 0.5960 - acc: 0.6782 - precision: 0.6368 - recall: 0.8164
Epoch 25: val_acc did not improve from 0.70833
54/54 [==============================] - 13s 235ms/step - loss: 0.5960 - acc: 0.6782 - precision: 0.6368 - recall:
Epoch 26/100
54/54 [==============================] - ETA: 0s - loss: 0.5951 - acc: 0.6712 - precision: 0.6279 - recall: 0.8337
Epoch 26: val_acc did not improve from 0.70833
54/54 [==============================] - 13s 248ms/step - loss: 0.5951 - acc: 0.6712 - precision: 0.6279 - recall:
Epoch 27/100
54/54 [==============================] - ETA: 0s - loss: 0.5885 - acc: 0.6788 - precision: 0.6343 - recall: 0.8384
Epoch 27: val_acc did not improve from 0.70833
54/54 [==============================] - 13s 236ms/step - loss: 0.5885 - acc: 0.6788 - precision: 0.6343 - recall:
Epoch 27: early stopping
```

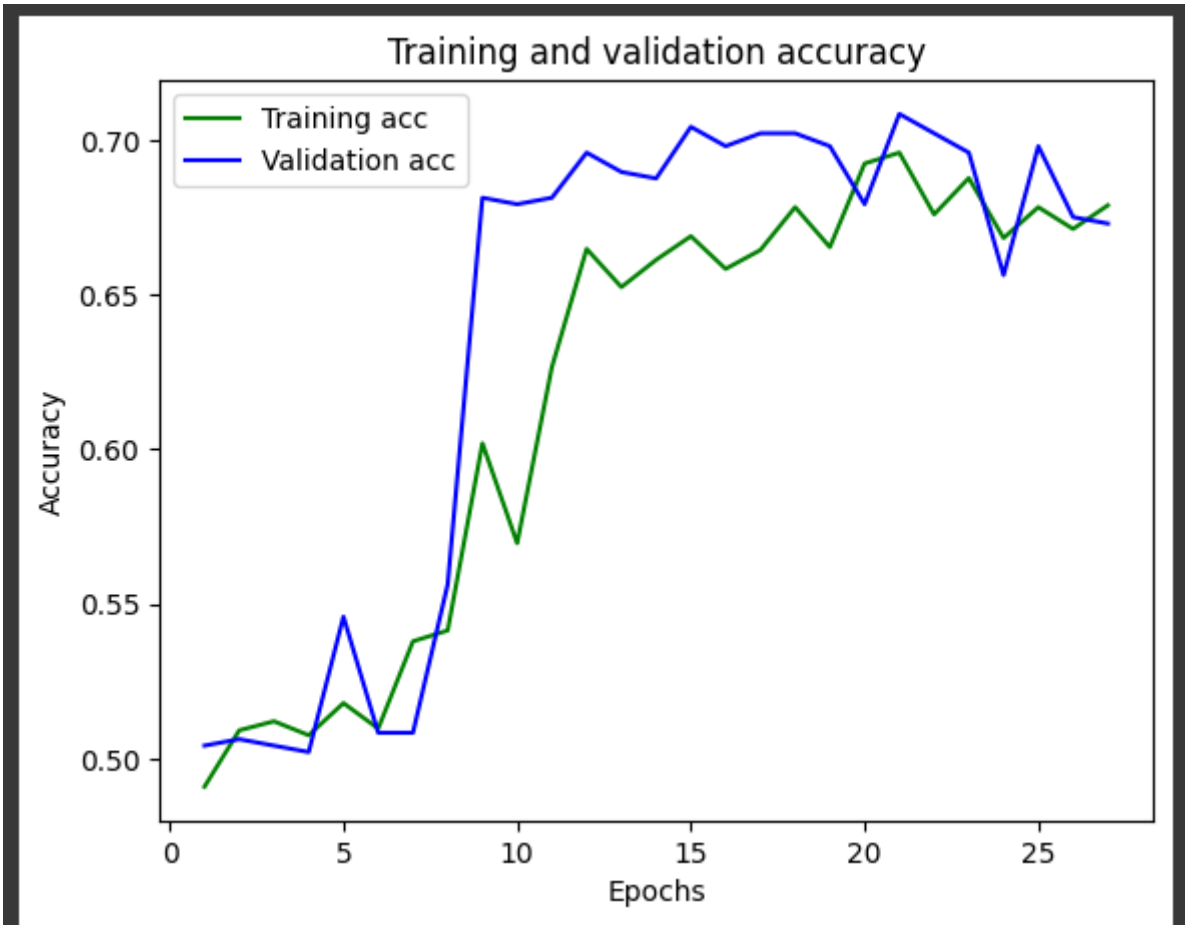
```
[24] hist=pd.DataFrame(history.history)
     hist.tail()
```

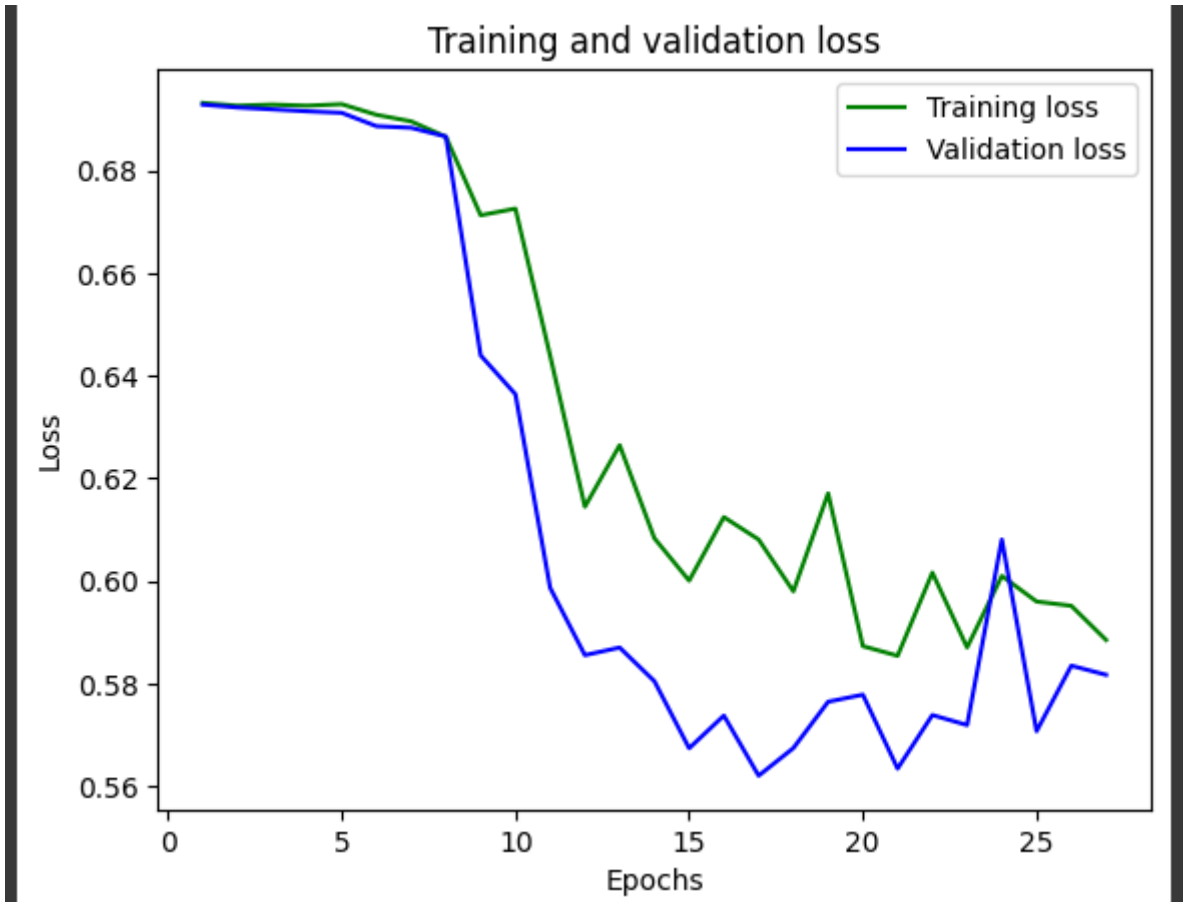
	loss	acc	precision	recall	auc	val_loss	val_acc	val_precision	val_recall	val_auc
22	0.587021	0.687610	0.644860	0.819477	0.737783	0.571899	0.695833	0.636103	0.921162	0.779319
23	0.600956	0.668233	0.631041	0.801653	0.712500	0.608055	0.656250	0.593909	0.979079	0.776845
24	0.595966	0.678215	0.636784	0.816351	0.716158	0.570702	0.697917	0.633929	0.906383	0.777195
25	0.595135	0.671169	0.627886	0.833726	0.722451	0.583466	0.675000	0.608696	0.949153	0.777334
26	0.588484	0.678802	0.634255	0.838443	0.733276	0.581687	0.672917	0.610959	0.936975	0.778379

Visualizing accuracy

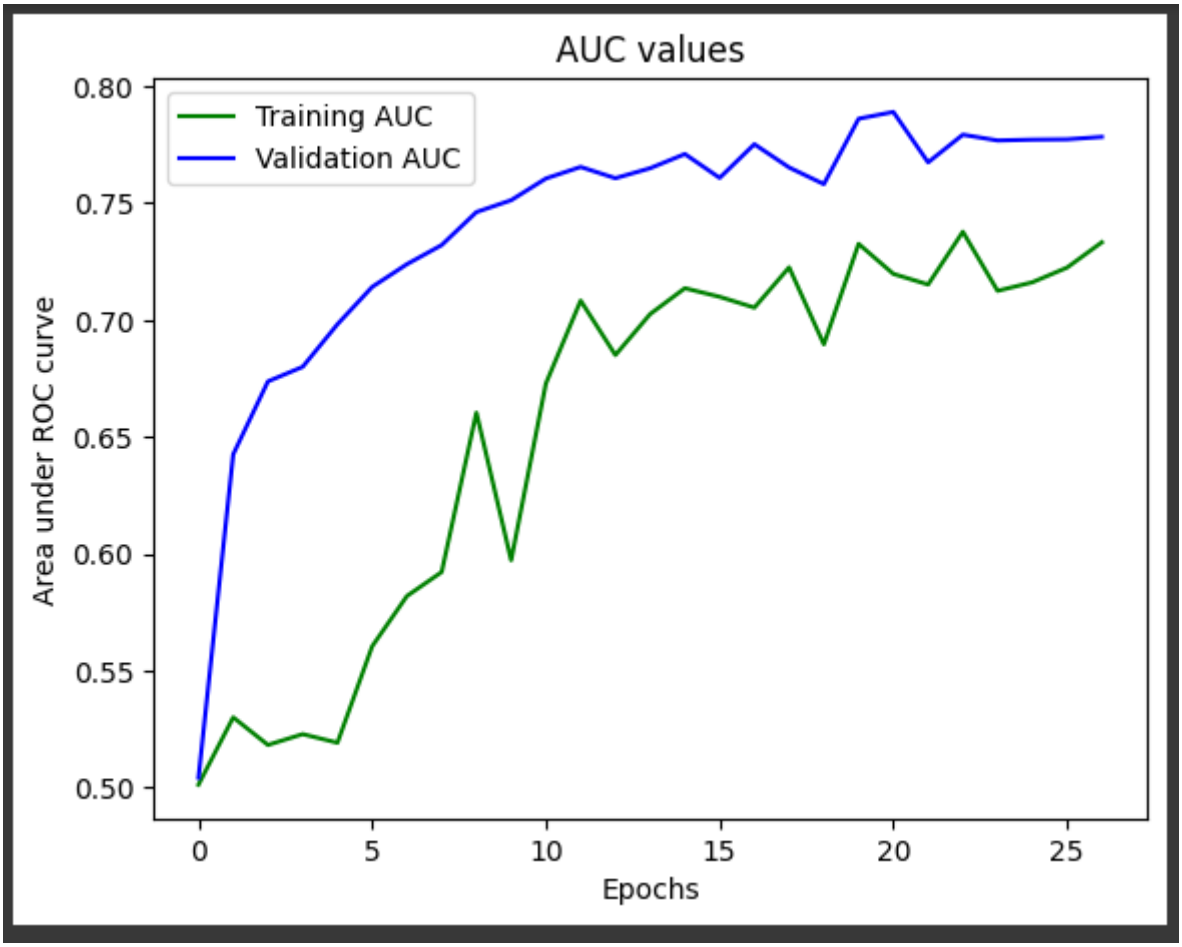
```
#@title Visualizing accuracy
acc = hist['acc']
val_acc = hist['val_acc']
loss = hist['loss']
val_loss = hist['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'g', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()

plt.figure()
```

```
plt.plot(epochs, loss, 'g', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()

plt.show()
```


Training and validation accuracy
Training acc
Validation acc
Accuracy
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
Epochs

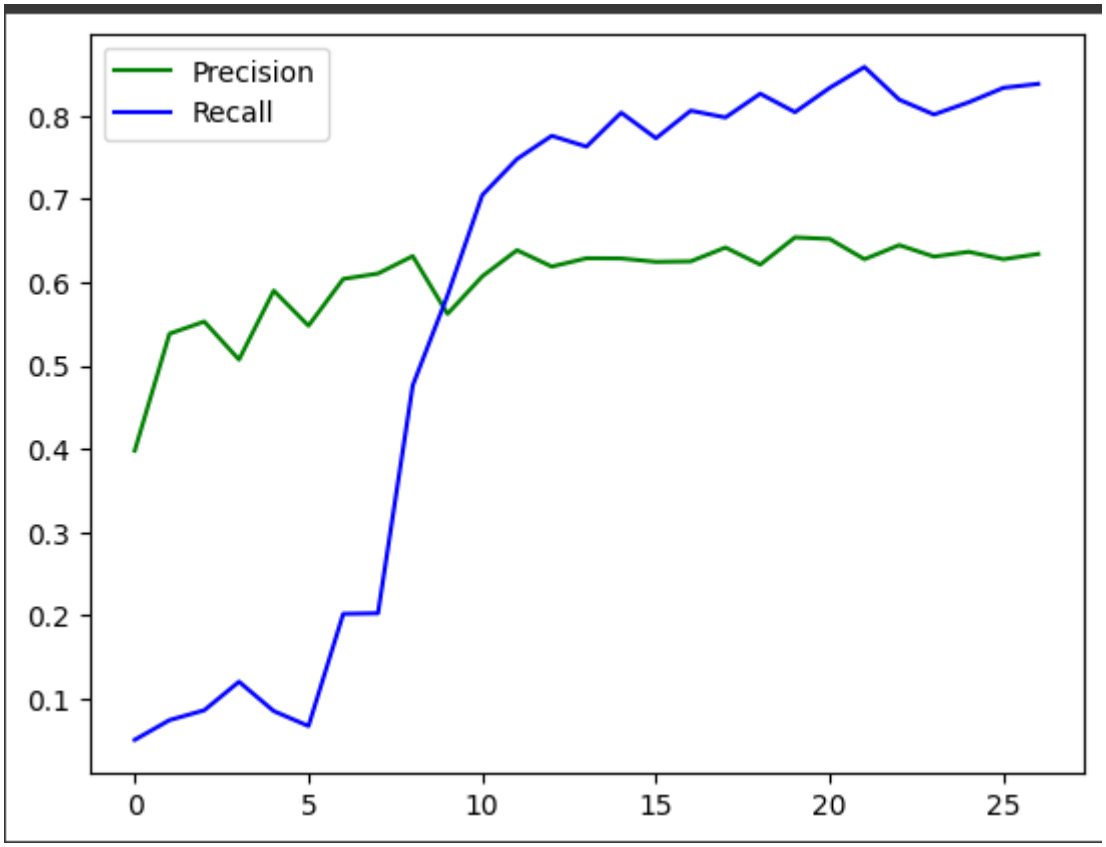

Training and validation loss
Training loss
Validation loss
Loss
0.68
0.66
0.64
0.62
0.60
0.58
0.56
0
5
10
15
20
25
Epochs

```
plt.plot(hist['auc'], 'g', label='Training AUC')
plt.plot(hist['val_auc'], 'b', label='Validation AUC')
plt.legend()
plt.xlabel("Epochs")
plt.ylabel("Area under ROC curve")
plt.title("AUC values")
plt.show()
```




```
[28] # Here, AUC value is closer to 75%
```

```
[29] plt.plot(hist['precision'], 'g', label='Precision')
     plt.plot(hist['recall'], 'b', label='Recall')
     plt.legend()
     plt.show()
```




```
[36] # Here, precision is around 65% & recall is about 85%.
     # It means the models have some chance of giving False positive (slightly low precision),
     # but it is less likely to return false negative (high recall), and maximum positive cases will be caught.
```

Saving model

```
[31] #@title Saving model
     #saving the model for future reference
     resnet_model.save("CovidPredict.h5")
```

Prediction

```
[32] #@title Prediction
     ##       PREDICTION  ##

     # Loading the model
     my_model=load_model('CovidPredict.h5')
```

```
[33] #Generating our test data set
     test_dir=curr+'/final/test'

     test_datagen = ImageDataGenerator(rescale=1./255)
     test_generator = test_datagen.flow_from_directory(test_dir,
                                                       target_size=(224, 224),
                                                       batch_size=32,
                                                       class_mode='binary',
                                                      color_mode='grayscale',
                                                     shuffle=False)
```

```
Found 250 images belonging to 2 classes.
```

```
[34] #evaluating model for test dataset for the metrics
     loss,acc,pre,recall,auc=my_model.evaluate(test_generator, verbose=2)
```

```
8/8 - 3s - loss: 0.5588 - acc: 0.7240 - precision: 0.6608 - recall: 0.9113 - auc: 0.7776 - 3s/epoch - 369ms/step
```

```
[35] # We can also use model.predict to get individual prediction
     # this will return array of value between 0 to 1, & after defining a thresold(generally 0.5), we classify it.
     predt=my_model.predict(test_generator)
```

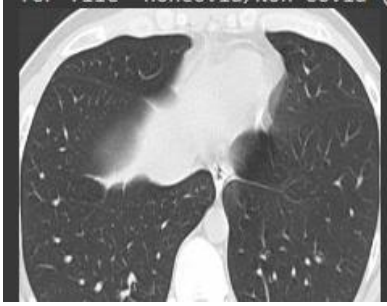
```
8/8 [==============================] - 3s 135ms/step
```

```
[39] files=test_generator.filenames
     class_dict=test_generator.class_indices # a dictionary of the form class name: class index
     rev_dict={}
     for key, value in class_dict.items():
         rev_dict[value]=key   # dictionary of the form class index: class name
```

```
[68] prediction=predt
     for i, p in enumerate(prediction):
         index=np.argmax(p)
         klass=rev_dict[index]
         prob=p[index]
         print('for file ', files[i], ' predicted class is ', klass,' with probability ',(1-prob)*100)
         im = imgshow.open("/content/resize/"+files[i])
         im.show()
         print('**************')
```

```
**************
for file  noncovid/Non-Covid (993).png  predicted class is  covid  with probability  35.43 %
```

```
**************
for file  noncovid/Non-Covid (998).png  predicted class is  covid  with probability  34.7 %
```



```
**************
for file  covid/Covid (790).png  predicted class is  covid  with probability  85.03 %
```

```
**************
for file  covid/Covid (793).png  predicted class is  covid  with probability  73.06 %
```

```
✓ 45s  completed at 5:06 PM
```