Screenshot of the Application

+ Code + Text

```
[1] #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 7.4 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
```

```
[3] 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
```

```
[5] import pandas as pd
    import numpy as np
    from keras.models import load_model
    import matplotlib.pyplot as plt
```

```
[6] #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'
```

```
[9]  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'
```

```
[14] #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, 4351.02 files/s]
```

```
[15] #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)
```

```
[16] 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')

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

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

```
[18] #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
```

```
#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.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.Dropout(0.5))
resnet_model.add(layers.Dense(512, activation='relu'))
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)

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

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

 flatten (Flatten)           (None, 128)               0

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 512)               66048

 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'])
```

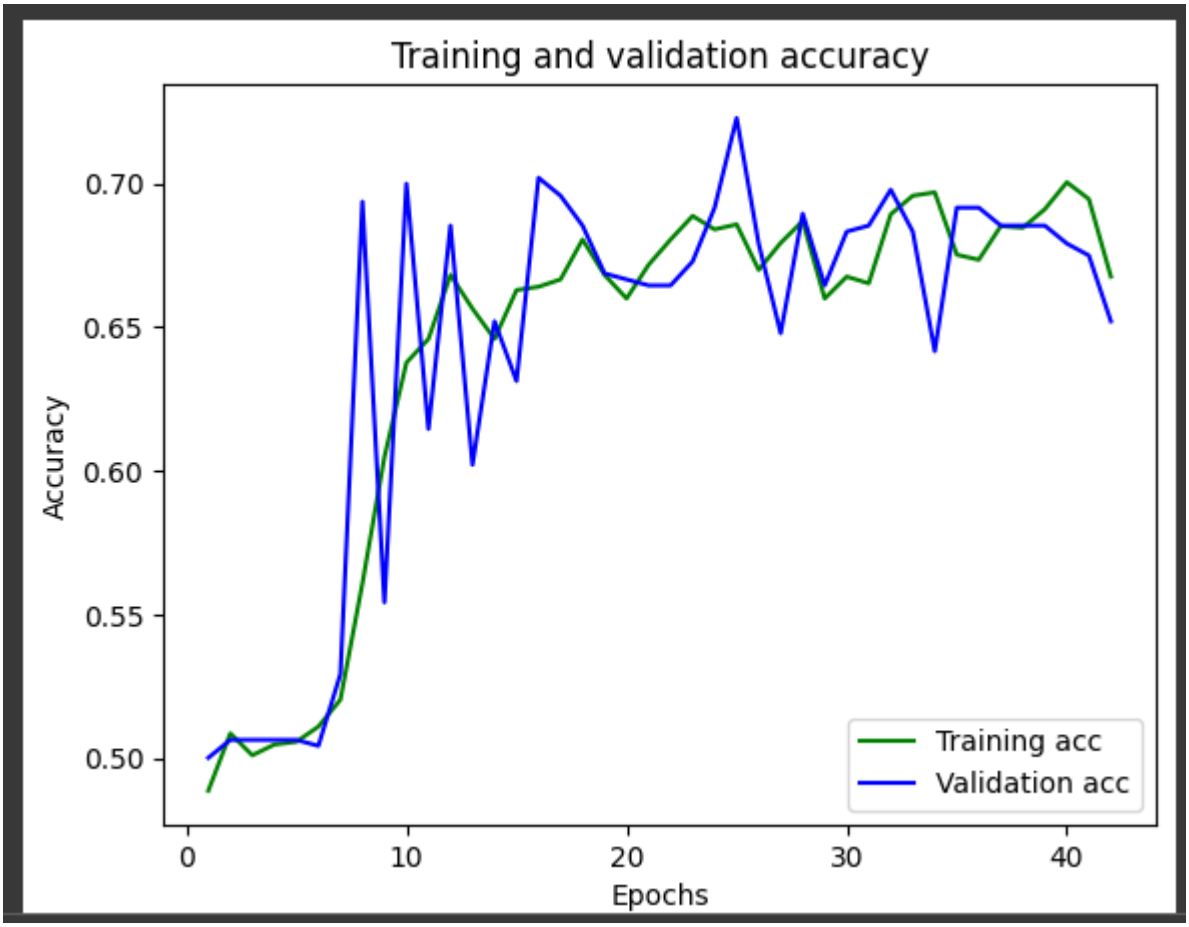
```
[23] #model fitting
     batch_size=32
     history=resnet_model.fit(train_generator,
                        steps_per_epoch=1735//batch_size,
                        epochs=100,
                        validation_data=validation_generator,
                        validation_steps=495//batch_size,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

```
54/54 [==============================] - ETA: 0s - loss: 0.5858 - acc: 0.6911 - precision: 0.6464 - recall: 0.8304 - auc:
Epoch 39: val_acc did not improve from 0.72292
54/54 [==============================] - 13s 229ms/step - loss: 0.5858 - acc: 0.6911 - precision: 0.6464 - recall: 0.8304
Epoch 40/100
53/54 [============================>.] - ETA: 0s - loss: 0.5734 - acc: 0.7005 - precision: 0.6541 - recall: 0.8311 - auc:
Epoch 40: val_acc did not improve from 0.72292
54/54 [==============================] - 14s 263ms/step - loss: 0.5735 - acc: 0.7005 - precision: 0.6542 - recall: 0.8319
Epoch 41/100
54/54 [==============================] - ETA: 0s - loss: 0.5751 - acc: 0.6947 - precision: 0.6516 - recall: 0.8219 - auc:
Epoch 41: val_acc did not improve from 0.72292
54/54 [==============================] - 14s 251ms/step - loss: 0.5751 - acc: 0.6947 - precision: 0.6516 - recall: 0.8219
Epoch 42/100
54/54 [==============================] - ETA: 0s - loss: 0.5873 - acc: 0.6676 - precision: 0.6247 - recall: 0.8312 - auc:
Epoch 42: val_acc did not improve from 0.72292
54/54 [==============================] - 13s 240ms/step - loss: 0.5873 - acc: 0.6676 - precision: 0.6247 - recall: 0.8312
Epoch 42: early stopping
```

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

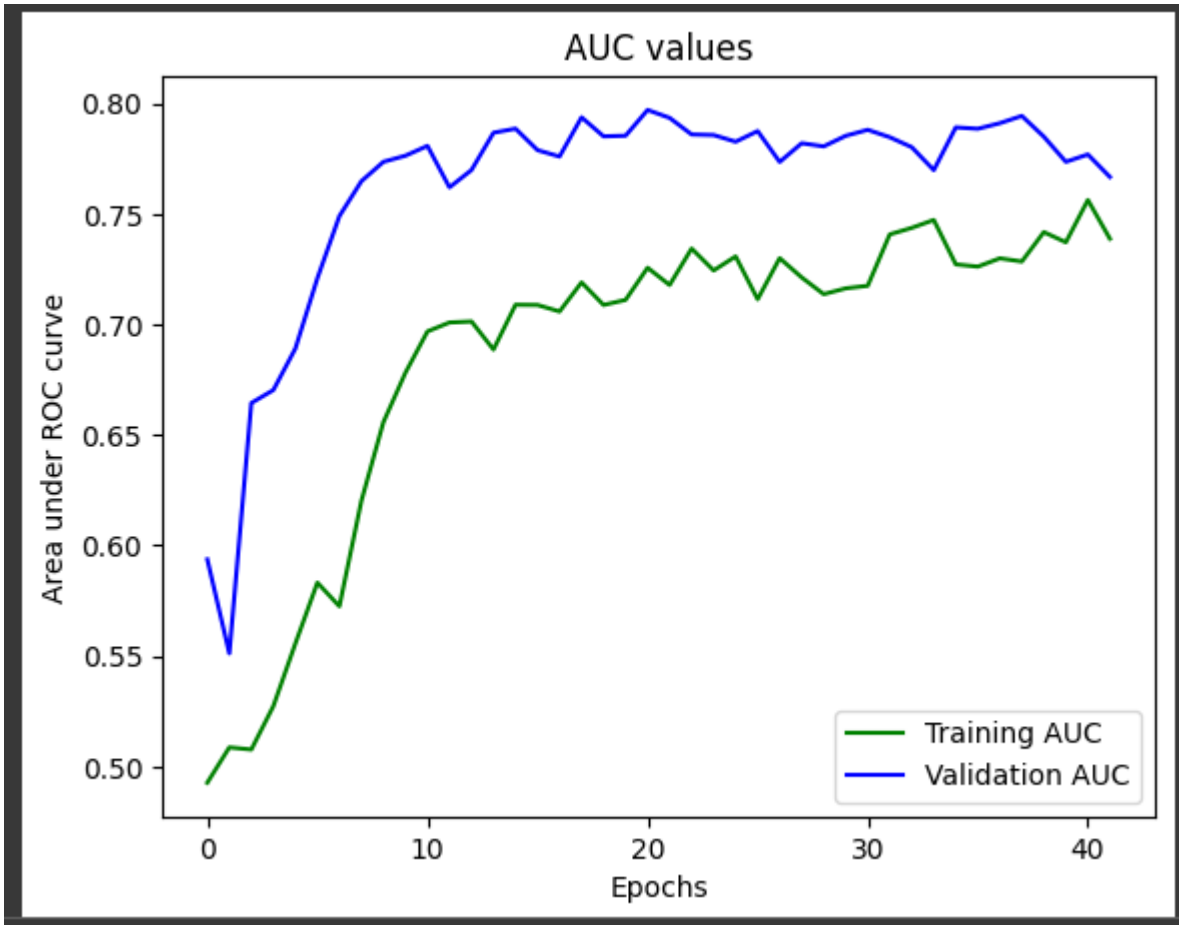
	loss	acc	precision	recall	auc	val_loss	val_acc	val_precision	val_recall	val_auc
37	0.588929	0.684674	0.638468	0.845519	0.728447	0.572346	0.685417	0.620112	0.936709	0.794273
38	0.585814	0.691133	0.646353	0.830368	0.741747	0.569873	0.685417	0.625000	0.920502	0.784892
39	0.573469	0.700529	0.654171	0.831943	0.737138	0.577034	0.679167	0.613889	0.936441	0.773522
40	0.575123	0.694656	0.651601	0.821853	0.756249	0.582048	0.675000	0.611702	0.958333	0.776849
41	0.587297	0.667645	0.624667	0.831169	0.738742	0.601359	0.652083	0.588832	0.978903	0.766674

```
[43] 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()
```



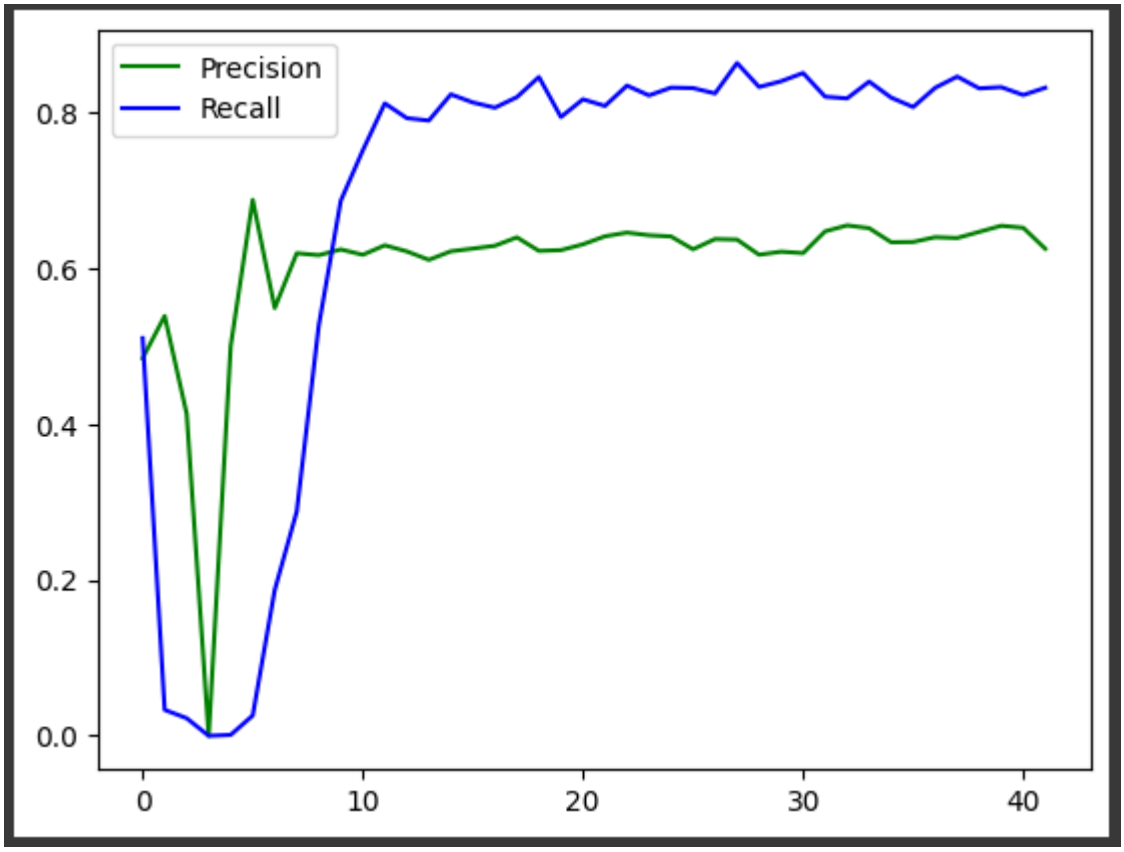

```
[65] # Here we can see, train & validation data's acuuracy & loss aprroches to each other, and its having good accuracy (76 %)
```

```
[69] 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()
```




```
# Here, AUC value is closer to 75%
```

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




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

```
[28] #saving the model for future reference
     resnet_model.save("CovidPredict.h5")
```

```
[29] ##       PREDICTION  ##

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

```
[30] #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')
```

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

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

     8/8 - 4s - loss: 0.5822 - acc: 0.6880 - precision: 0.6198 - recall: 0.9597 - auc: 0.7843 - 4s/epoch - 474ms/step
```

```
[67] # 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 [==============================] - 1s 101ms/step
```