CT Scan Image Classification

CT Scan Image Classification

Connect to Google Drive to access Dataset

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
    %cd '/content/drive/My Drive/Colab Notebooks/Data/'
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/My Drive/Colab Notebooks/Data
```

Import Libraries

```
from builtins import range, input

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten, AveragePooling2D, Dropout
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import confusion_matrix, roc_curve
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

import cv2
from glob import glob

from keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.utils import to_categorical
```

```
#define size to which images are to be resized
IMAGE_SIZE = [80, 80]

#define paths
covid_path = '/content/drive/MyDrive/Colab Notebooks/Data/COVID'
noncovid_path = '/content/drive/MyDrive/Colab Notebooks/Data/non-COVID'

# Use glob to grab images from path .jpg or jpeg
covid_files = glob(covid_path + '/*')
noncovid_files = glob(noncovid_path + '/*')
```

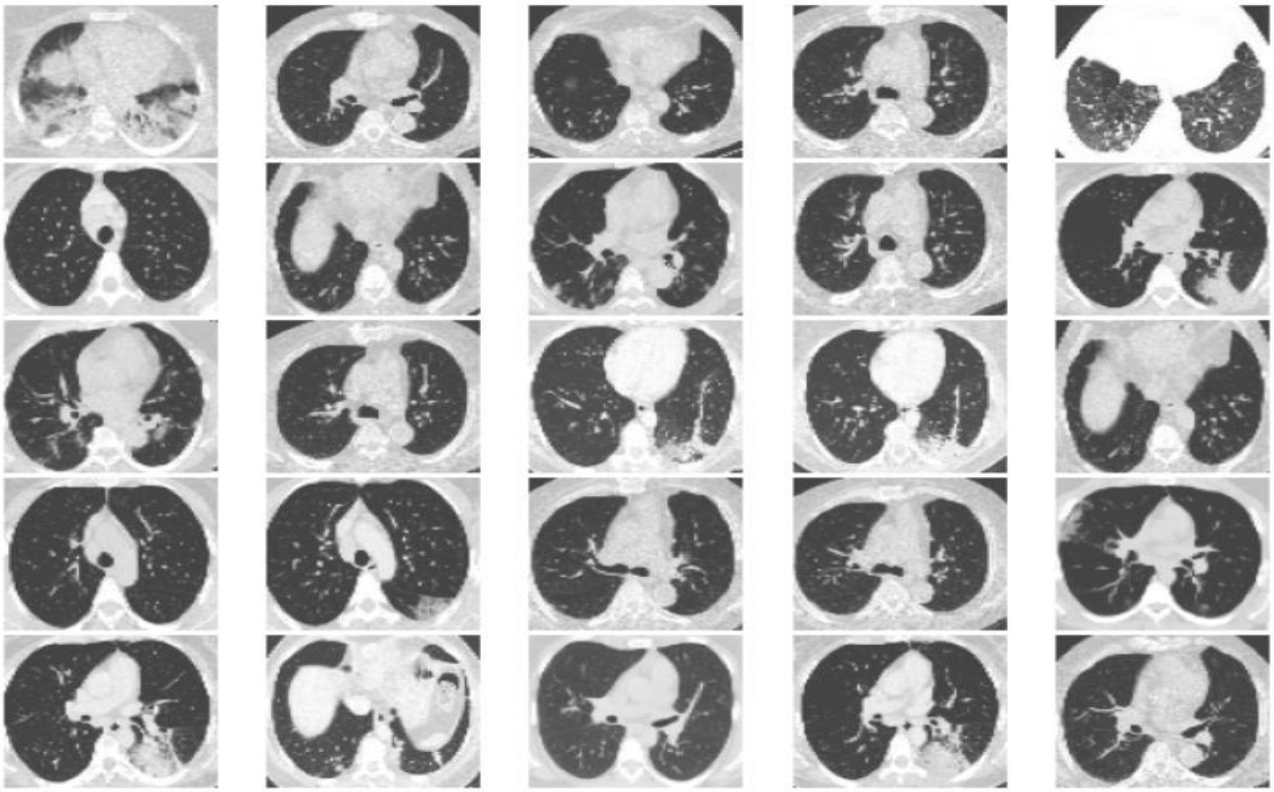
Resize images

```
[ ]  covid_labels = []
     noncovid_labels = []

     covid_images=[]
     noncovid_images=[]

     for i in range(len(covid_files)):
       image = cv2.imread(covid_files[i])              # read file
       image = cv2.resize(image,(80,80))               # resize as per model for covid
       covid_images.append(image)                      # append image
       covid_labels.append('COVID')                    #append class label

     for i in range(len(noncovid_files)):
       image = cv2.imread(noncovid_files[i])
       image = cv2.resize(image,(80,80))
       noncovid_images.append(image)
       noncovid_labels.append('non-COVID')
```

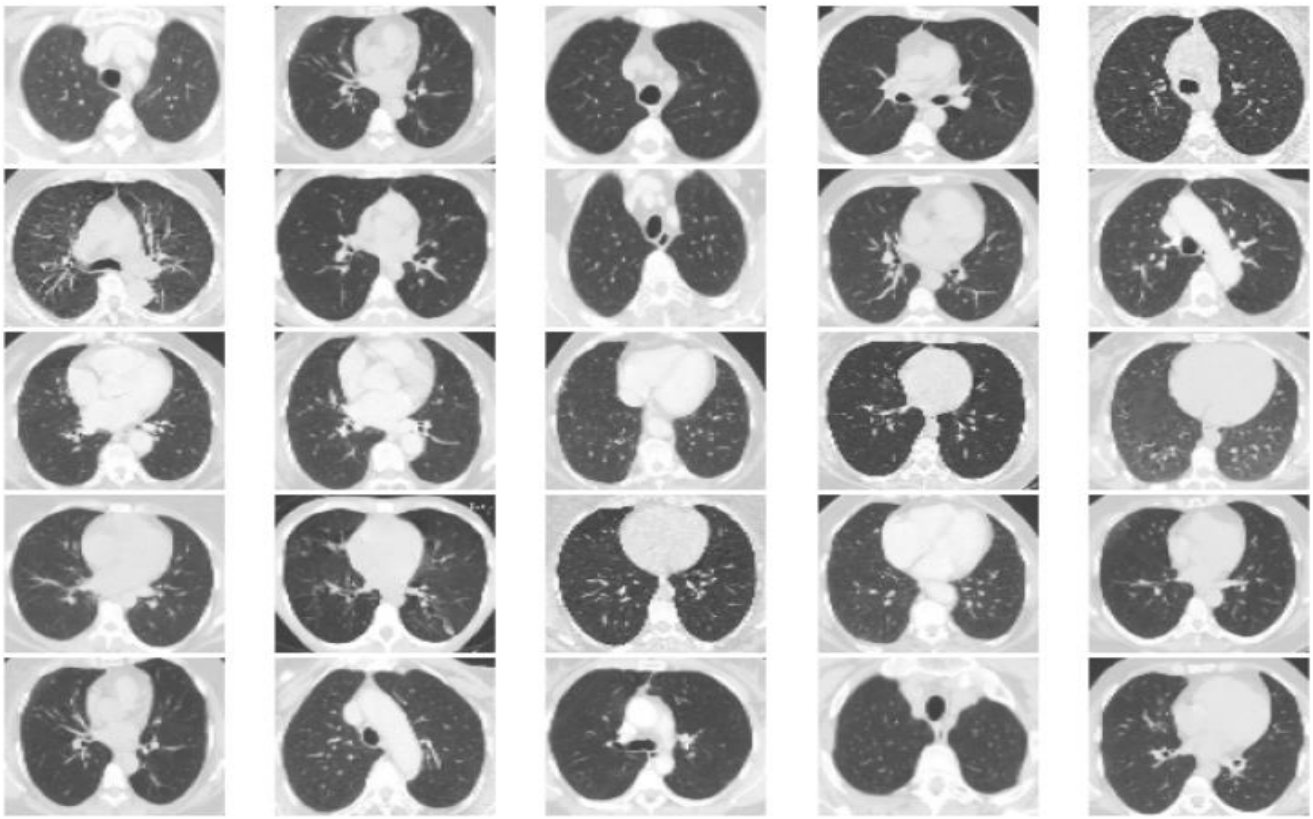
Visualize First 25 images from dataset

```
# look at a random image for fun
def plot_images(images, title):
    nrows, ncols = 5, 5
    figsize = [6, 6]

    fig, ax = plt.subplots(nrows=nrows, ncols=ncols, figsize=figsize, facecolor=(1, 1, 1))

    for i, axi in enumerate(ax.flat):
        axi.imshow(images[i])
        axi.set_axis_off()

    plt.suptitle(title, fontsize=24)
    plt.tight_layout(pad=0.2, rect=[0, 0, 1, 0.9])
    plt.show()
plot_images(covid_images, 'COVID-19 Positive CT Scan')
plot_images(noncovid_images, 'COVID-19 Negative CT Scan')
```

COVID-19 Positive CT Scan



COVID-19 Negative CT Scan



Normalization

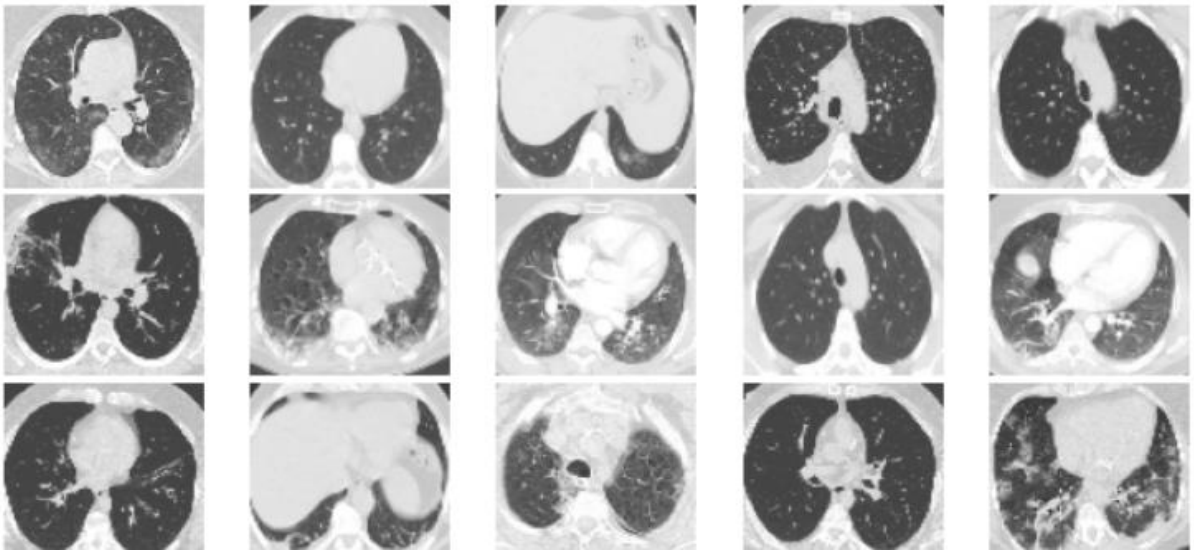
```
[ ]  # Convert to array and Normalize to interval of [0,1]
     covid_images = np.array(covid_images) / 255
     noncovid_images = np.array(noncovid_images) / 255
```

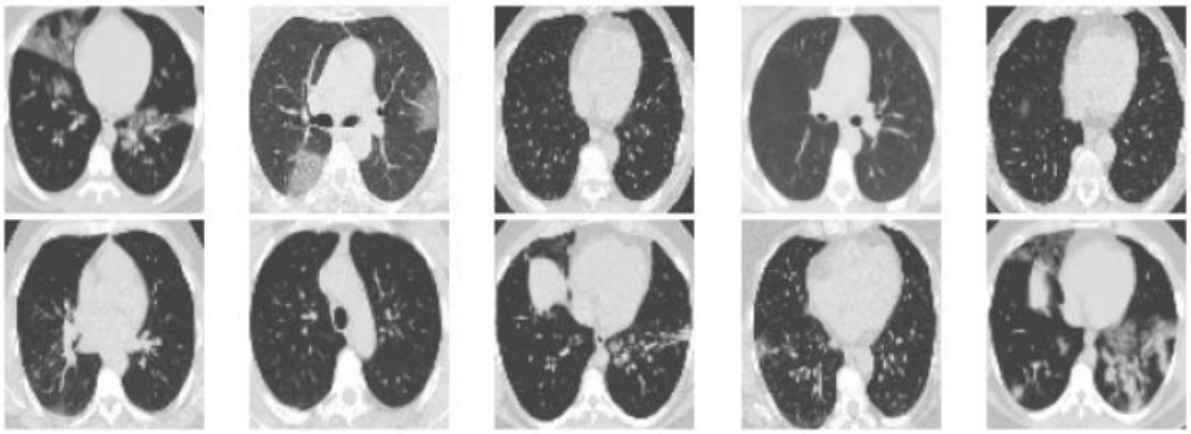
Train Test Splitting

```
# Split into training and testing sets for both types of images

covid_x_train, covid_x_test, covid_y_train, covid_y_test = train_test_split(
    covid_images, covid_labels, test_size=0.2, random_state=1)

noncovid_x_train, noncovid_x_test, noncovid_y_train, noncovid_y_test = train_test_split(
    noncovid_images, noncovid_labels, test_size=0.2, random_state=1)

# Merge sets for both types of images
X_train = np.concatenate((noncovid_x_train, covid_x_train), axis=0)
X_test = np.concatenate((noncovid_x_test, covid_x_test), axis=0)
y_train = np.concatenate((noncovid_y_train, covid_y_train), axis=0)
y_test = np.concatenate((noncovid_y_test, covid_y_test), axis=0)

# Make labels into categories - either 0 or 1, for our model
y_train = LabelBinarizer().fit_transform(y_train)
y_train = to_categorical(y_train)
y_test = LabelBinarizer().fit_transform(y_test)
y_test = to_categorical(y_test)

# y_train and y_test contain class lables 0 and 1 representing COVID and NonCOVID for X_train and X_test
```

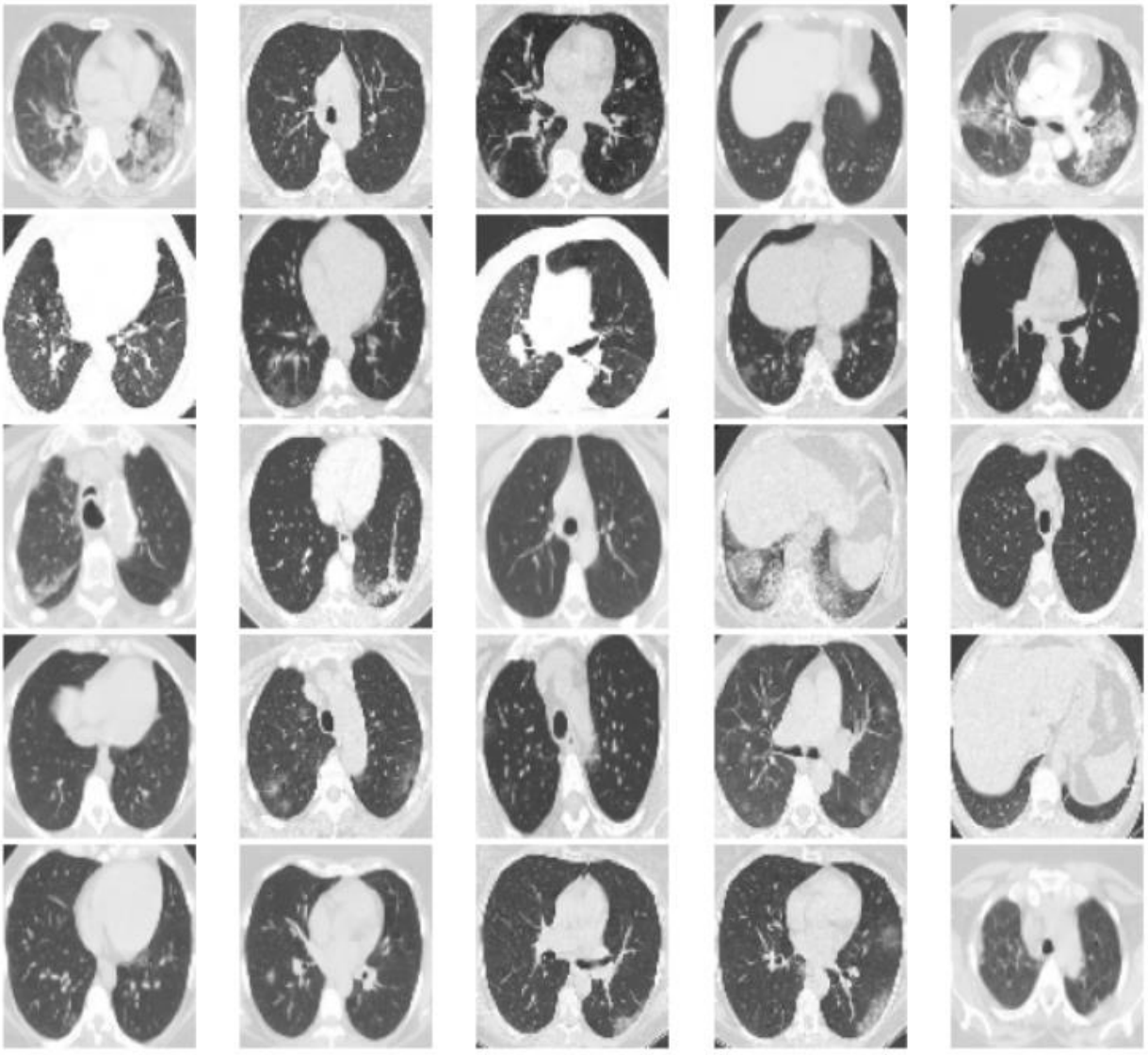
```
plot_images(covid_x_train, 'X_train')
plot_images(covid_x_test, 'X_test')
```

X_train





X_test



Model Building

```
[ ]  # Building Model
     from tensorflow import keras
     resnet = ResNet50(weights="imagenet", include_top=False,
         input_tensor=Input(shape=(80, 80, 3)))

     outputs = resnet.output
     outputs = Flatten(name="flatten")(outputs)
     outputs = Dropout(0.3)(outputs)
     outputs = Dense(2, activation="sigmoid")(outputs)

     model = Model(inputs=resnet.input, outputs=outputs)

     for layer in resnet.layers:
         layer.trainable = False

     model.compile(
             loss='binary_crossentropy',
             optimizer=keras.optimizers.Adamax(learning_rate=0.001),
             metrics=['accuracy']
     )
```

```
# Define callbacks
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    verbose=1,
    restore_best_weights=True
)

model_checkpoint = ModelCheckpoint(
    'model_checkpoint.h5',
    monitor='val_loss',
    save_best_only=True,
    verbose=1
)
```

```
model.summary()
```

Data Augmentation

```
[ ]  datagen = ImageDataGenerator(rotation_range=20,
       width_shift_range=0.2,
       height_shift_range=0.2,
       zoom_range=0.2,
       horizontal_flip=True
     )
```

Training the Model

```
[ ]  batch_size=2
     epochs=30

     hist = model.fit(datagen.flow(X_train, y_train, batch_size=batch_size),
                      validation_data=(X_test, y_test),
                      validation_steps=len(X_test) / batch_size,
                      steps_per_epoch=len(X_train) / batch_size,
                      epochs=epochs,callbacks=[early_stopping, model_checkpoint]
                     )
```

```
Epoch 1/30
991/993 [============================>.] - ETA: 0s - loss: 0.6901 - accuracy: 0.6044
Epoch 1: val_loss improved from inf to 0.59253, saving model to model_checkpoint.h5
993/993 [==============================] - 23s 20ms/step - loss: 0.6897 - accuracy: 0.6047 - val_loss: 0.5925 - val_accuracy: 0.6419
Epoch 2/30
990/993 [============================>.] - ETA: 0s - loss: 0.6860 - accuracy: 0.6116
Epoch 2: val_loss did not improve from 0.59253
993/993 [==============================] - 17s 17ms/step - loss: 0.6853 - accuracy: 0.6123 - val_loss: 0.6269 - val_accuracy: 0.6519
Epoch 3/30
991/993 [============================>.] - ETA: 0s - loss: 0.6722 - accuracy: 0.6302
Epoch 3: val_loss did not improve from 0.59253
993/993 [==============================] - 16s 16ms/step - loss: 0.6715 - accuracy: 0.6304 - val_loss: 0.8262 - val_accuracy: 0.5855
Epoch 4/30
992/993 [============================>.] - ETA: 0s - loss: 0.6733 - accuracy: 0.6280
Epoch 4: val_loss did not improve from 0.59253
993/993 [==============================] - 16s 16ms/step - loss: 0.6733 - accuracy: 0.6284 - val_loss: 0.8341 - val_accuracy: 0.5775
Epoch 5/30
991/993 [============================>.] - ETA: 0s - loss: 0.6508 - accuracy: 0.6433
Epoch 5: val_loss improved from 0.59253 to 0.57693, saving model to model_checkpoint.h5
993/993 [==============================] - 18s 18ms/step - loss: 0.6503 - accuracy: 0.6440 - val_loss: 0.5769 - val_accuracy: 0.6962
Epoch 6/30
993/993 [==============================] - ETA: 0s - loss: 0.6611 - accuracy: 0.6339
Epoch 6: val_loss did not improve from 0.57693
993/993 [==============================] - 17s 17ms/step - loss: 0.6611 - accuracy: 0.6339 - val_loss: 0.6241 - val_accuracy: 0.6559
Epoch 7/30
992/993 [============================>.] - ETA: 0s - loss: 0.6523 - accuracy: 0.6361
Epoch 7: val_loss did not improve from 0.57693
993/993 [==============================] - 16s 16ms/step - loss: 0.6520 - accuracy: 0.6365 - val_loss: 0.6993 - val_accuracy: 0.6217
Epoch 8/30
989/993 [============================>.] - ETA: 0s - loss: 0.6457 - accuracy: 0.6481
Epoch 8: val_loss improved from 0.57693 to 0.55151, saving model to model_checkpoint.h5
993/993 [==============================] - 17s 18ms/step - loss: 0.6457 - accuracy: 0.6485 - val_loss: 0.5515 - val_accuracy: 0.7284

993/993 [==============================] - 17s 18ms/step - loss: 0.6457 - accuracy: 0.6485 - val_loss: 0.5515 - val_accuracy: 0.7284
Epoch 9/30
990/993 [============================>.] - ETA: 0s - loss: 0.6566 - accuracy: 0.6364
Epoch 9: val_loss did not improve from 0.55151
993/993 [==============================] - 16s 16ms/step - loss: 0.6563 - accuracy: 0.6370 - val_loss: 0.5785 - val_accuracy: 0.6861
Epoch 10/30
993/993 [==============================] - ETA: 0s - loss: 0.6366 - accuracy: 0.6475
Epoch 10: val_loss did not improve from 0.55151
993/993 [==============================] - 17s 17ms/step - loss: 0.6366 - accuracy: 0.6475 - val_loss: 0.6372 - val_accuracy: 0.6499
Epoch 11/30
993/993 [==============================] - ETA: 0s - loss: 0.6583 - accuracy: 0.6410
Epoch 11: val_loss improved from 0.55151 to 0.54613, saving model to model_checkpoint.h5
993/993 [==============================] - 18s 18ms/step - loss: 0.6583 - accuracy: 0.6410 - val_loss: 0.5461 - val_accuracy: 0.7364
Epoch 12/30
991/993 [============================>.] - ETA: 0s - loss: 0.6523 - accuracy: 0.6398
Epoch 12: val_loss did not improve from 0.54613
993/993 [==============================] - 16s 16ms/step - loss: 0.6526 - accuracy: 0.6400 - val_loss: 0.6415 - val_accuracy: 0.6519
Epoch 13/30
990/993 [============================>.] - ETA: 0s - loss: 0.6521 - accuracy: 0.6323
Epoch 13: val_loss did not improve from 0.54613
993/993 [==============================] - 16s 16ms/step - loss: 0.6515 - accuracy: 0.6334 - val_loss: 0.5796 - val_accuracy: 0.6740
Epoch 14/30
991/993 [============================>.] - ETA: 0s - loss: 0.6479 - accuracy: 0.6438
Epoch 14: val_loss did not improve from 0.54613
993/993 [==============================] - 17s 17ms/step - loss: 0.6472 - accuracy: 0.6445 - val_loss: 0.7726 - val_accuracy: 0.5956
Epoch 15/30
991/993 [============================>.] - ETA: 0s - loss: 0.6230 - accuracy: 0.6680
Epoch 15: val_loss improved from 0.54613 to 0.53028, saving model to model_checkpoint.h5
993/993 [==============================] - 18s 18ms/step - loss: 0.6231 - accuracy: 0.6682 - val_loss: 0.5303 - val_accuracy: 0.7404
```

```
Epoch 16/30
991/993 [============================>.] - ETA: 0s - loss: 0.6438 - accuracy: 0.6448
Epoch 16: val_loss did not improve from 0.53028
993/993 [==============================] - 16s 16ms/step - loss: 0.6436 - accuracy: 0.6450 - val_loss: 0.5368 - val_accuracy: 0.7203
Epoch 17/30
992/993 [============================>.] - ETA: 0s - loss: 0.6497 - accuracy: 0.6477
Epoch 17: val_loss did not improve from 0.53028
993/993 [==============================] - 16s 16ms/step - loss: 0.6493 - accuracy: 0.6480 - val_loss: 0.7487 - val_accuracy: 0.6076
Epoch 18/30
990/993 [============================>.] - ETA: 0s - loss: 0.6360 - accuracy: 0.6535
Epoch 18: val_loss did not improve from 0.53028
993/993 [==============================] - 16s 16ms/step - loss: 0.6361 - accuracy: 0.6531 - val_loss: 0.6345 - val_accuracy: 0.6479
Epoch 19/30
990/993 [============================>.] - ETA: 0s - loss: 0.6343 - accuracy: 0.6475
Epoch 19: val_loss did not improve from 0.53028
993/993 [==============================] - 16s 16ms/step - loss: 0.6354 - accuracy: 0.6470 - val_loss: 0.8400 - val_accuracy: 0.5674
Epoch 20/30
993/993 [==============================] - ETA: 0s - loss: 0.6493 - accuracy: 0.6490Restoring model weights from the end of the best epoch: 15.

Epoch 20: val_loss did not improve from 0.53028
993/993 [==============================] - 16s 16ms/step - loss: 0.6493 - accuracy: 0.6490 - val_loss: 0.7019 - val_accuracy: 0.6258
Epoch 20: early stopping
```

```
# Save Model and Weights
model.save('resnet_ct.h5')
model.save_weights('resnet_weights_ct.hdf5')
```

```
[ ] # Load saved model
    model = load_model('resnet_ct.h5')
    final_loss, final_accuracy = model.evaluate(X_test, y_test)
    print('Final Loss: {}, Final Accuracy: {}'.format(final_loss, final_accuracy))
```

```
16/16 [==============================] - 2s 29ms/step - loss: 0.5303 - accuracy: 0.7404
Final Loss: 0.5302808880805969, Final Accuracy: 0.7404426336288452
```

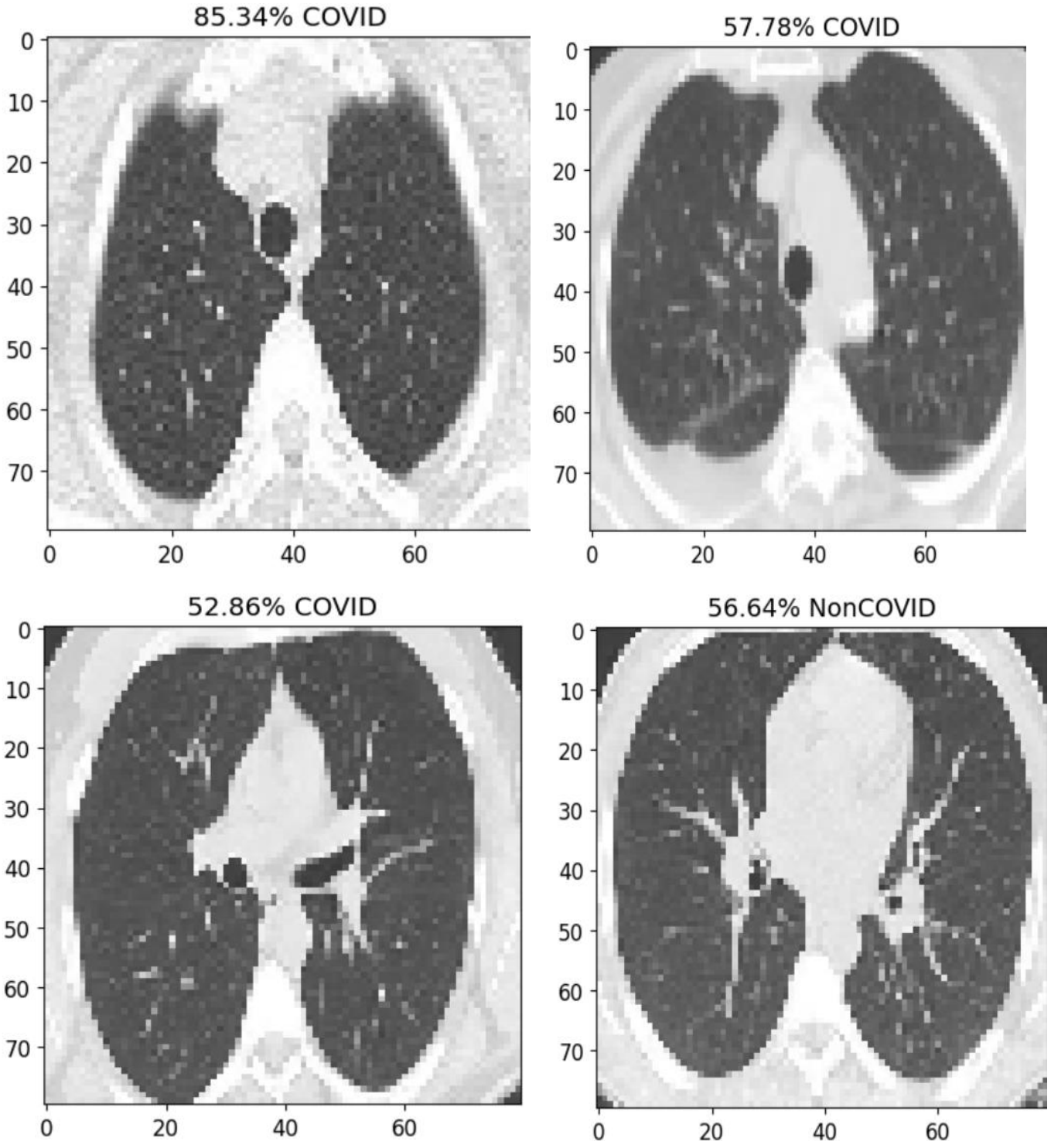
Prediction

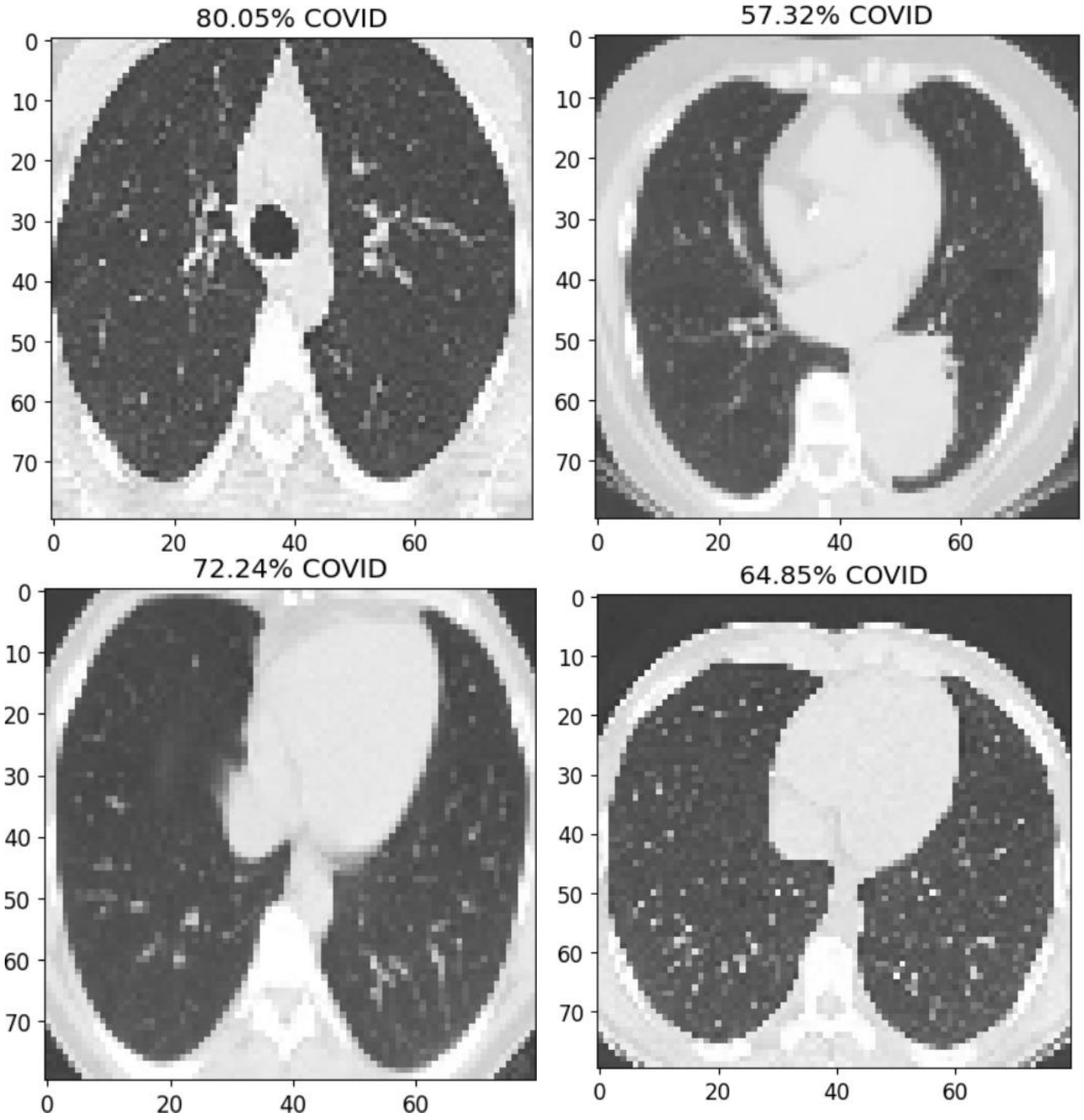
```
[ ] y_pred = model.predict(X_test,batch_size=batch_size)
```

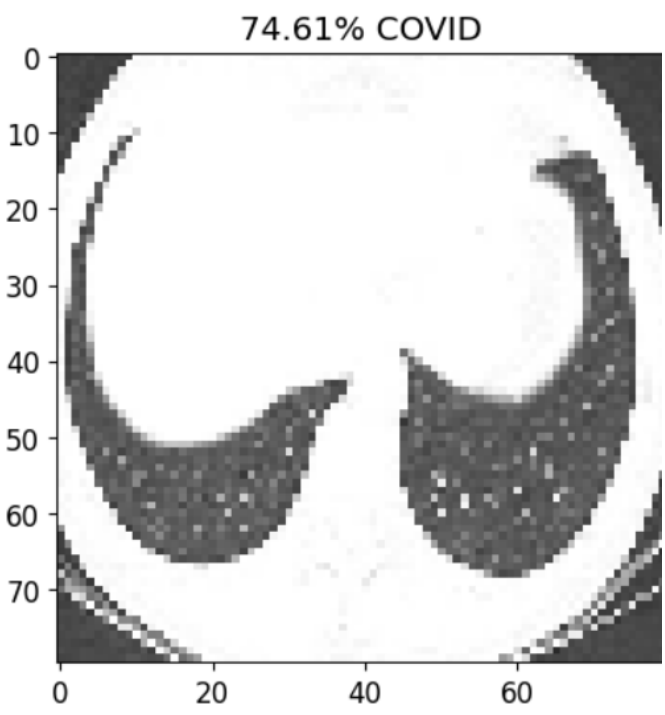
```
249/249 [==============================] - 4s 9ms/step
```

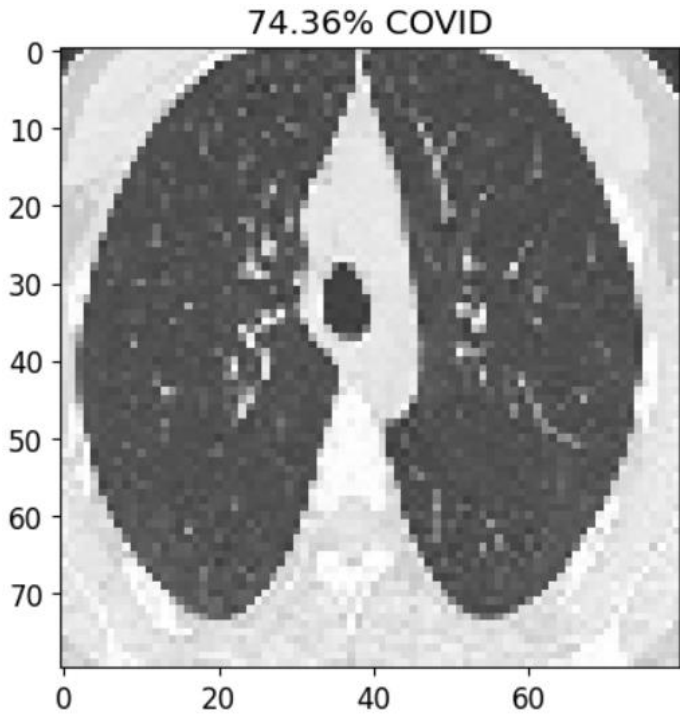
First 10 predictions

```
[ ] prediction=y_pred[0:10]
    for index, probability in enumerate(prediction):
      if probability[1] > 0.5:
            plt.title('%.2f' % (probability[1]*100) + '% COVID')
      else:
            plt.title('%.2f' % ((1-probability[1])*100) + '% NonCOVID')
      plt.imshow(X_test[index])
      plt.show()
```


85.34% COVID
57.78% COVID
52.86% COVID
56.64% NonCOVID


80.05% COVID
57.32% COVID
72.24% COVID
64.85% COVID






```
[ ]  # Convert to Binary classes
     y_pred_bin = np.argmax(y_pred, axis=1)
     y_test_bin = np.argmax(y_test, axis=1)
```

Classification Report

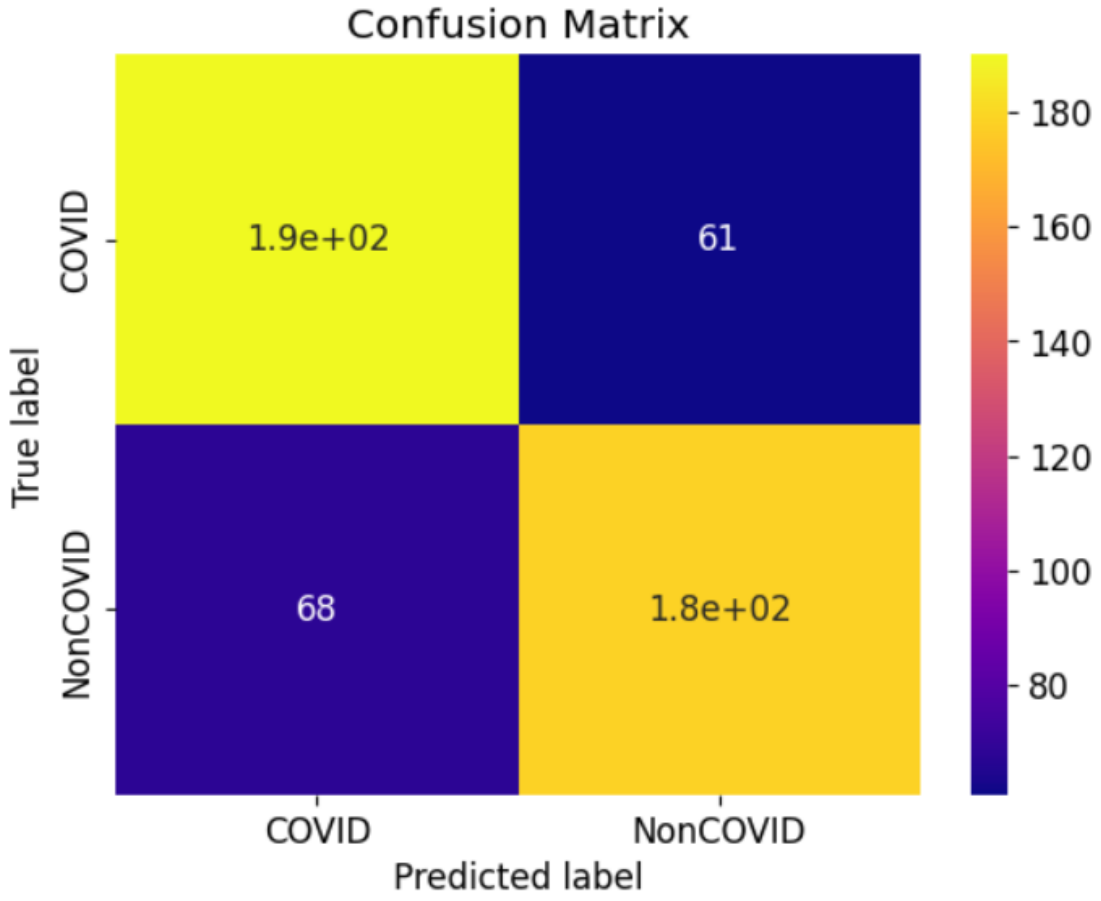
```
[ ]  from sklearn.metrics import classification_report
     print(classification_report(y_test_bin,y_pred_bin))
```

	precision	recall	f1-score	support
0	0.74	0.76	0.75	251
1	0.74	0.72	0.73	246
accuracy			0.74	497
macro avg	0.74	0.74	0.74	497
weighted avg	0.74	0.74	0.74	497

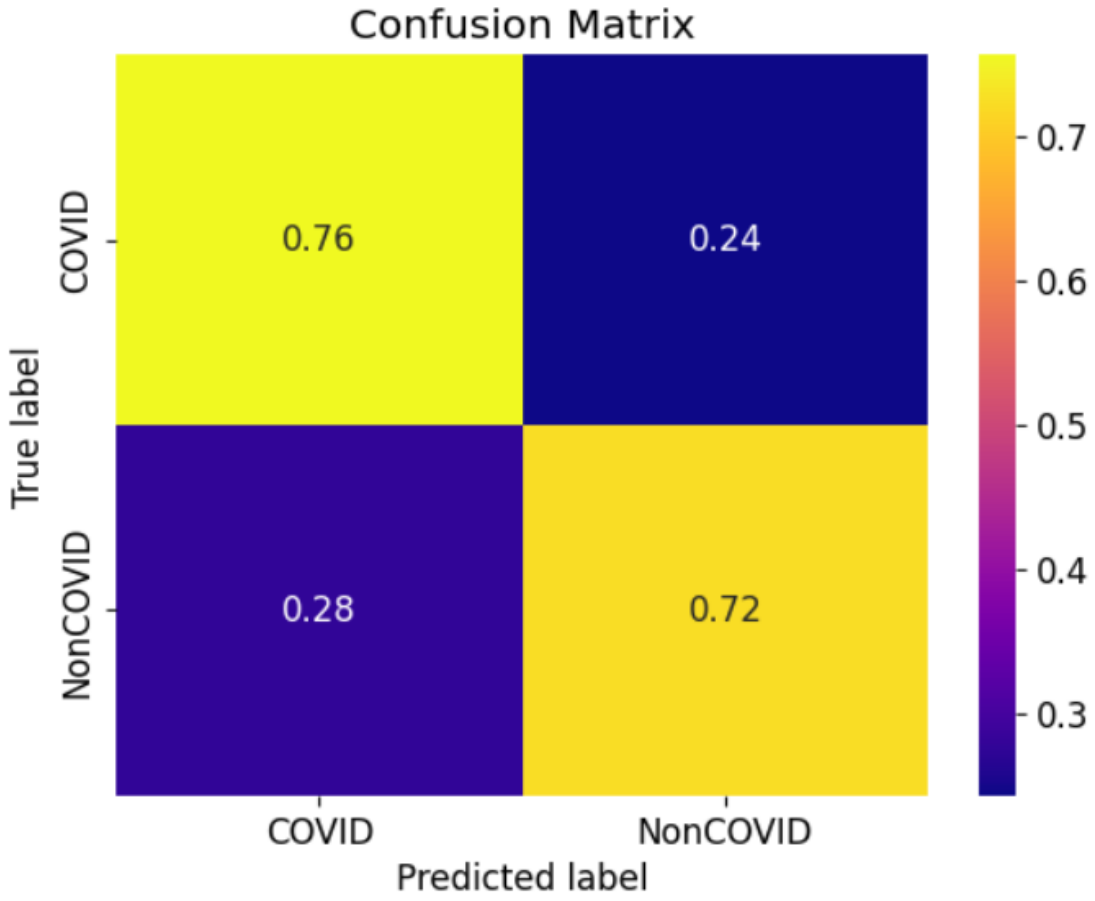
Confusion Matrix

```
[ ]  def plot_confusion_matrix(normalize):
       classes = ['COVID','NonCOVID']
       tick_marks = [0.5,1.5]
       cn = confusion_matrix(y_test_bin, y_pred_bin,normalize=normalize)
       sns.heatmap(cn,cmap='plasma',annot=True)
       plt.xticks(tick_marks, classes)
       plt.yticks(tick_marks, classes)
       plt.title('Confusion Matrix')
       plt.ylabel('True label')
       plt.xlabel('Predicted label')
       plt.show()

     print('Confusion Matrix without Normalization')
     plot_confusion_matrix(normalize=None)

     print('Confusion Matrix with Normalized Values')
     plot_confusion_matrix(normalize='true')
```

```
Confusion Matrix without Normalization
```

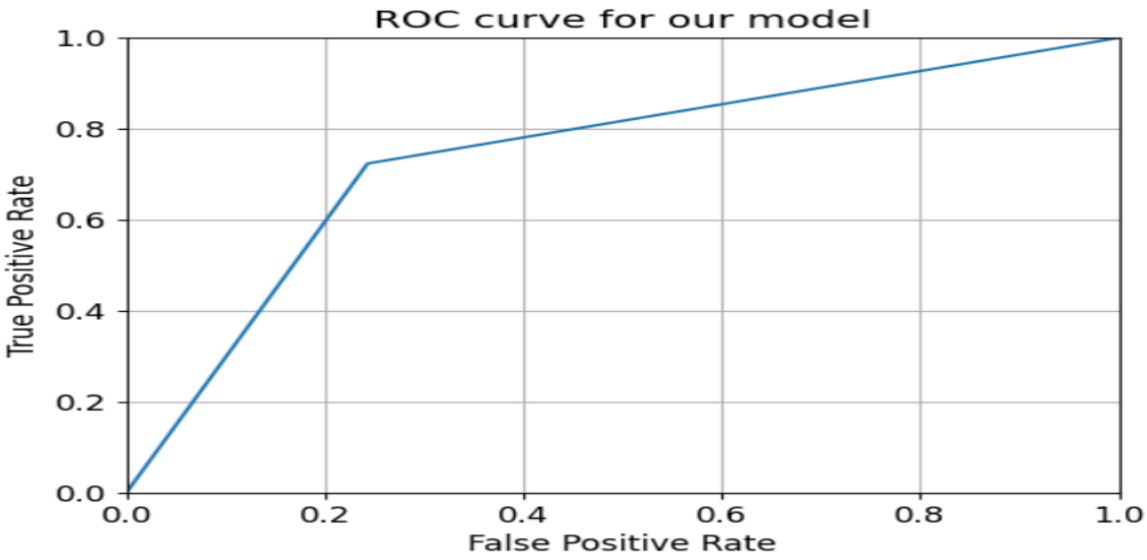


Confusion Matrix with Normalized Values



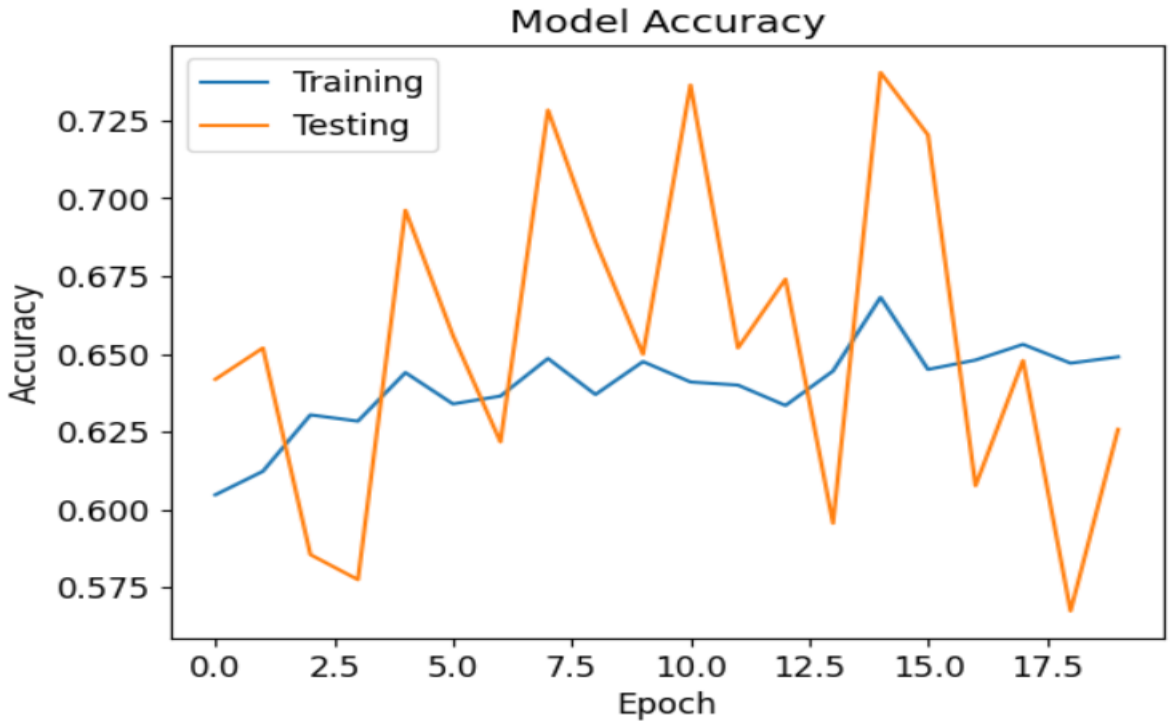
ROC Curve

```
[ ]  fpr, tpr, thresholds = roc_curve(y_test_bin, y_pred_bin)
     plt.plot(fpr, tpr)
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.0])
     plt.rcParams['font.size'] = 12
     plt.title('ROC curve for our model')
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.grid(True)
```



Model Accuracy Plot

```
[ ]  plt.plot(hist.history['accuracy'])
     plt.plot(hist.history['val_accuracy'])

     plt.title('Model Accuracy')
     plt.ylabel('Accuracy')
     plt.xlabel('Epoch')

     plt.legend(['Training', 'Testing'])
     plt.savefig('resnet_ct_accuracy.png')
     plt.show()
```



Model Loss Plot

```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])

plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')

plt.legend(['Training', 'Testing'])
plt.savefig('resnet_ct_loss.png')
plt.show()
```