CT Scan Image Classification

In recent years, due to explosive growth of digital content, automatic classification of images has become one of the most critical challenges in visual information indexing. It is big problem. Image classification is a task where the system takes an input image and classifies it with an appropriate label. in general image classification refers to task of extracting information from the image by labelling the pixels of the image to different classes. It can be done in two ways one is Supervised classification, Unsupervised classification.

Now a day, Deep learning algorithms are providing successful results in the areas like computer vision. The Convolutional Neural Network, a machine learning algorithm is being used for the image classification. CNN is a type of feed-forward artificial neural network that has been successfully applied to analyse visual images.

Dataset:

This dataset contains 1252 CT scans that are positive for SARS-CoV-2 infection (COVID-19) and 1230 CT scans for patients non-infected by SARS-CoV-2, 2482 CT scans in total. These data have been collected from real patients in hospitals from Sao Paulo, Brazil.

Aim:

To encourage the research and development of artificial intelligent methods which are able to identify if a person is infected by SARS-CoV-2 through the analysis of his/her CT scans.

Problem Statement:

In this case, the goal is to develop artificial intelligence methods that can identify if a person is infected with SARS-CoV-2 (the virus causing COVID-19) based on the analysis of their CT scan images.

Building Image Classification Project:

Load Dataset:

Here I have used Keras Library to load each layer that works on top of TensorFlow so we have to import all libraries which are important for further process.

Now load all the images and resize as per our convenience so we have to take a fixed size on which we can work. Here we take image size [80,80].

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
```

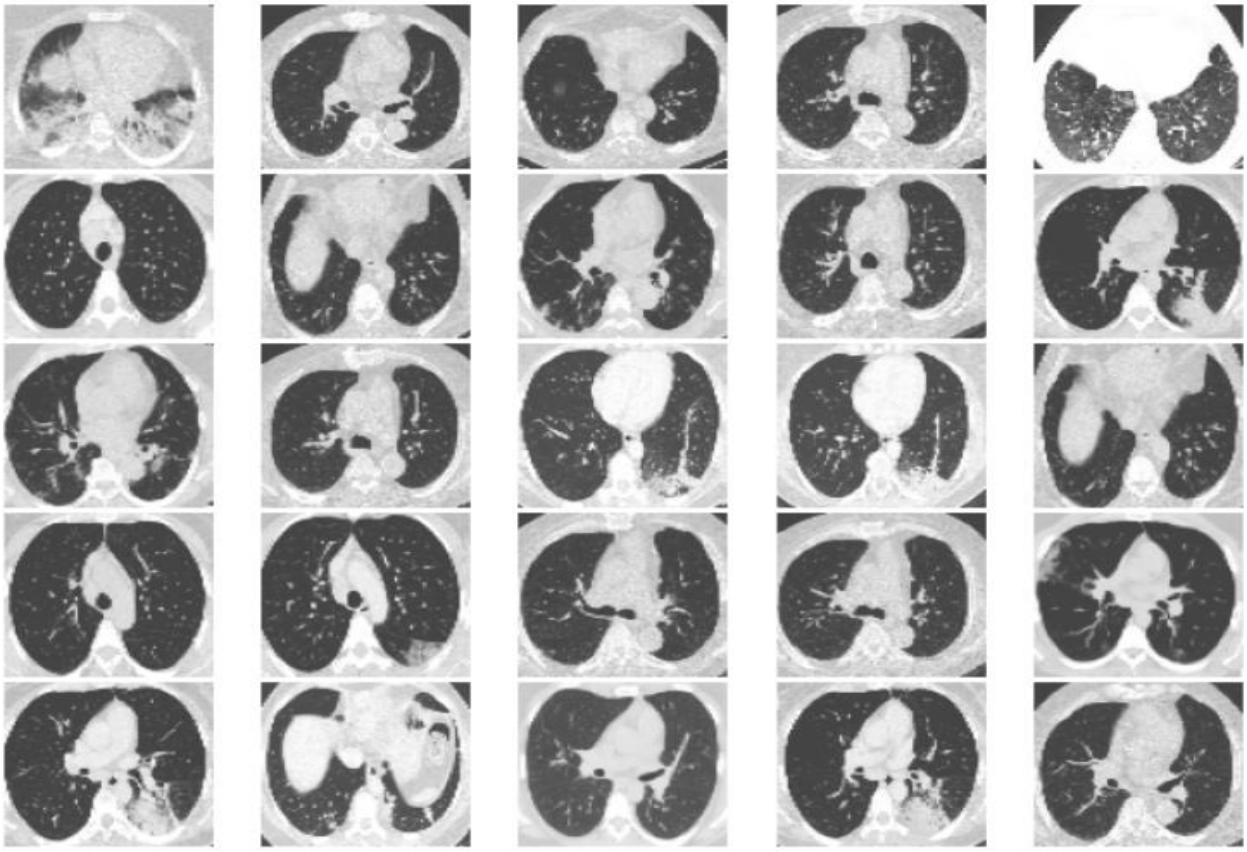
Data Processing:

```
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')
```

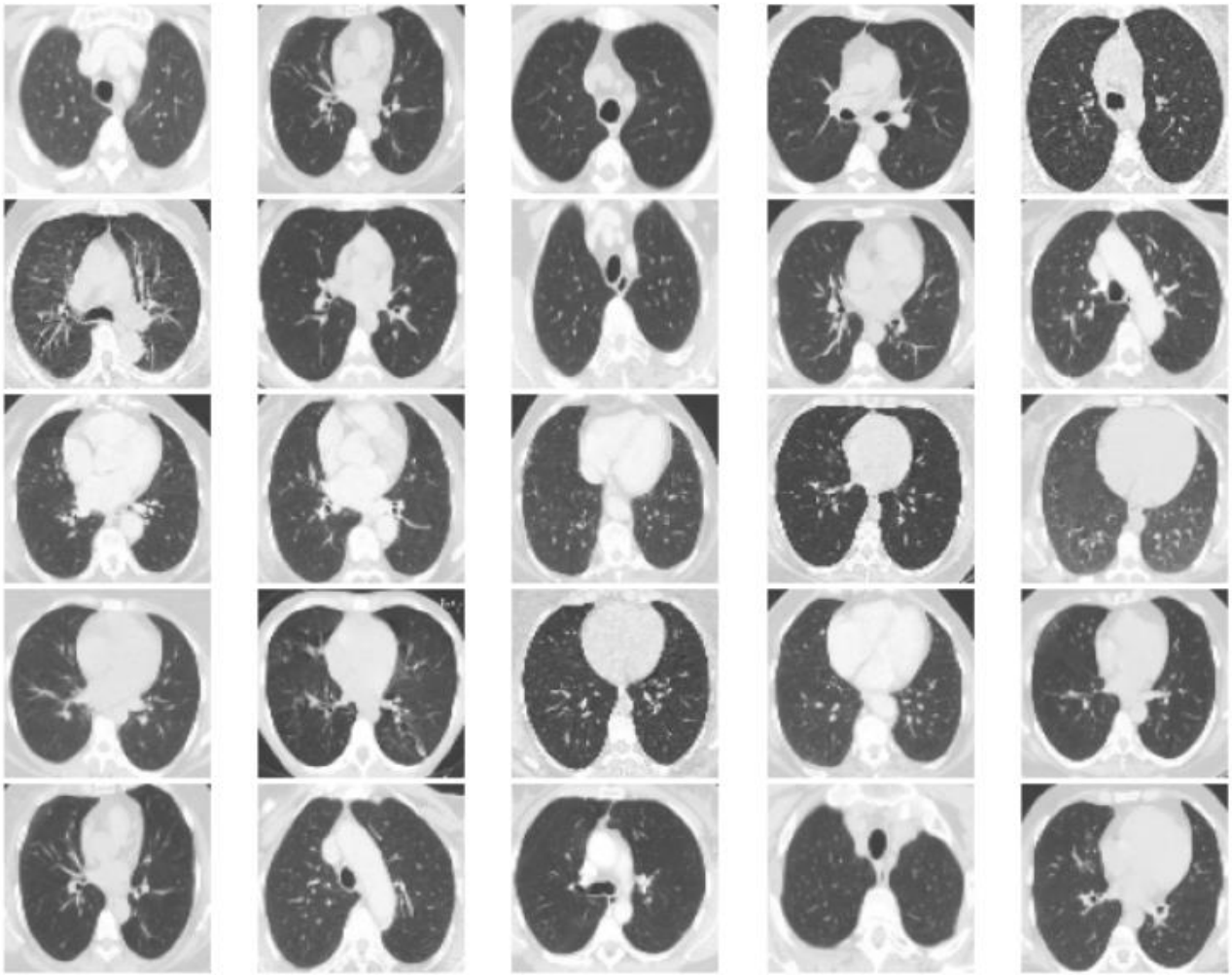
Explore sample Image

Here we can see some images of covid positive CT scan images and covid negative CT scan images.

COVID-19 Positive CT Scan:



COVID-19 Negative CT Scan:



Now we have to convert data into array and normalize it using NumPy. Pixels always range between 0-225.

Data Augmentation- ImageDataGenerator is created with the specified augmentation parameters. Original image is then pass through the data generator to obtain augmented image.

Split Datasets into train and test

y train and y test contain class labels 0 and 1 representing COVID and Non-COVID for X train and X test.

Model Selection:

Here I am using ResNet50 model (50 layers Residual Network) it has ability to train very deep networks effectively and their excellent performance on changing recognition task.

The architecture of our model:

- Flatten- to convert parrel layer to squeeze the layer into 1 dimension.
- Dropout- it is regularization technique to reduce overfitting. Dropout with rate 0.3
- Dense- feed forward neural network (2 nodes, activation =" sigmoid")

Here we mention early stopping and model checkpoint.

Early stopping is a technique used to prevent a model from training for too long and potentially overfitting the data. It involves monitoring a specified metric, such as validation loss or accuracy, during the training process.

Model Checkpoint is a callback function that allows you to save the model's weights during training, typically after each epoch or whenever a specified metric improves. It helps in creating checkpoints of the model's weights at different stages of training.

Training the model:

Train the selected model on the training set using binary cross-entropy loss functions, optimizers Adamax, and evaluation metrics accuracy. Monitor the training process to ensure the model is learning and not overfitting.

The prediction process involves passing the input data through the model's layers, applying activation functions, aggregating information, and producing an output based on the model's architecture and learned parameters.

Accuracy of our model is 75%. our model correctly predicted the outcome of the data in 75% of the cases. If we have 100 data points and our model correctly predicts 75 of them, while it makes incorrect predictions for the remaining 25.

Model Building

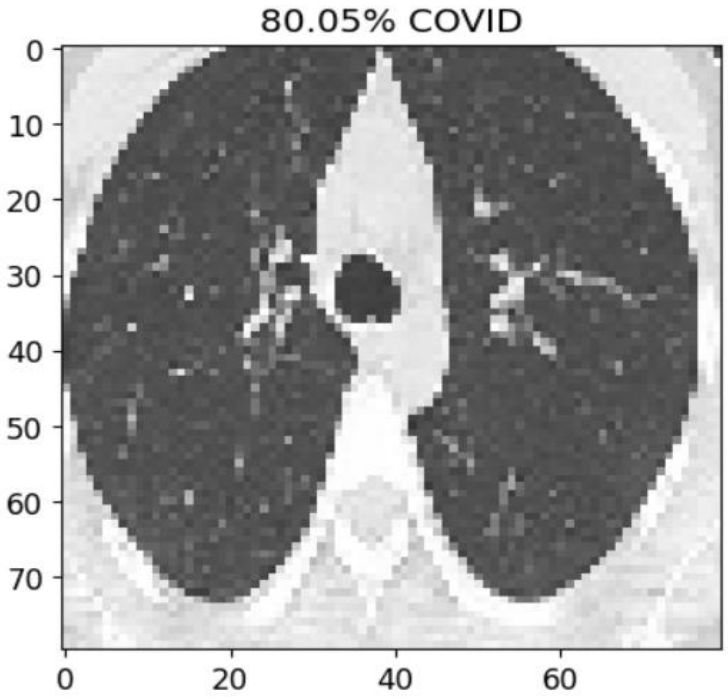
```
[ ]  # Building Model
     from tensorflow import keras
     resnet = ResNet50(weights="imagenet", include_top=False,
         input_tensor=Input(shape=(80, 80, 3)))

     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']
     )
```

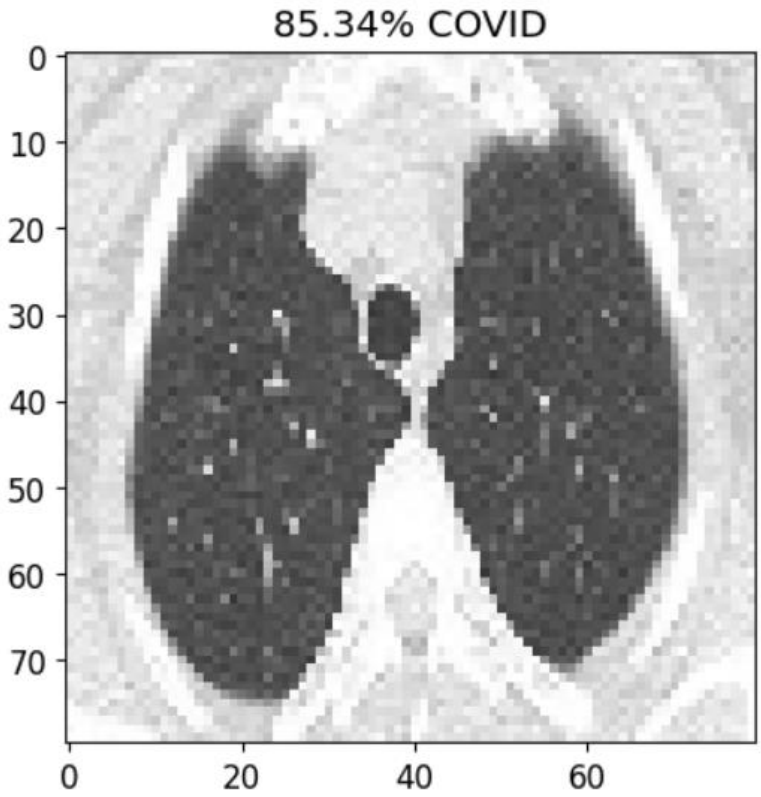
```
# Define callbacks
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    verbose=1,
    restore_best_weights=True
)

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

```
# 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
```

Predicted images: some predicted images.





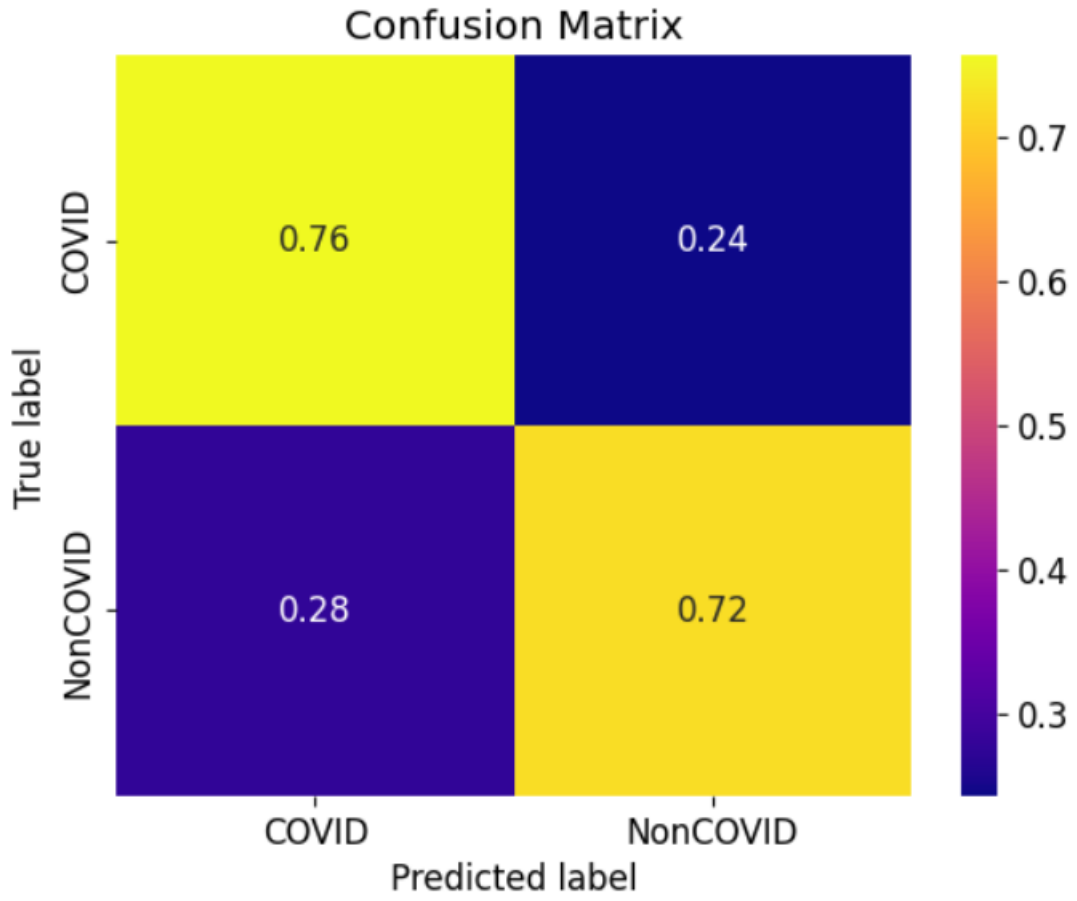
Classification Report:

```
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
```

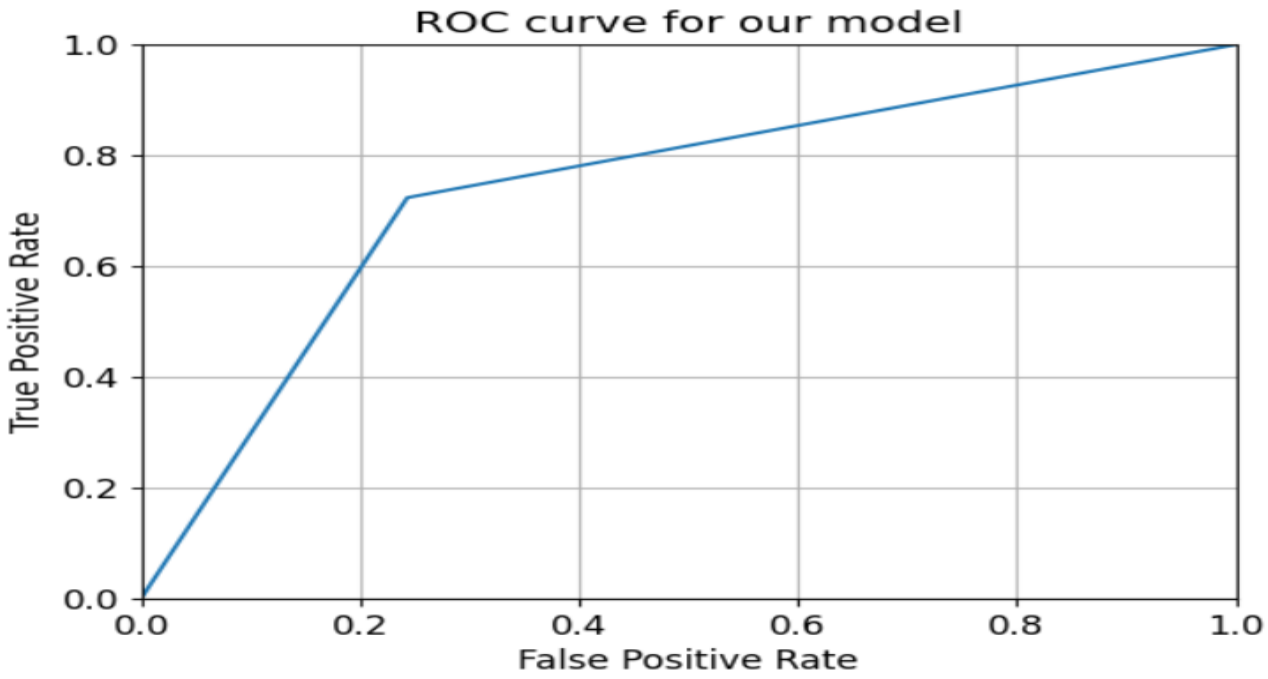
Confusion Matrix:

```
Confusion Matrix with Normalized Values
```



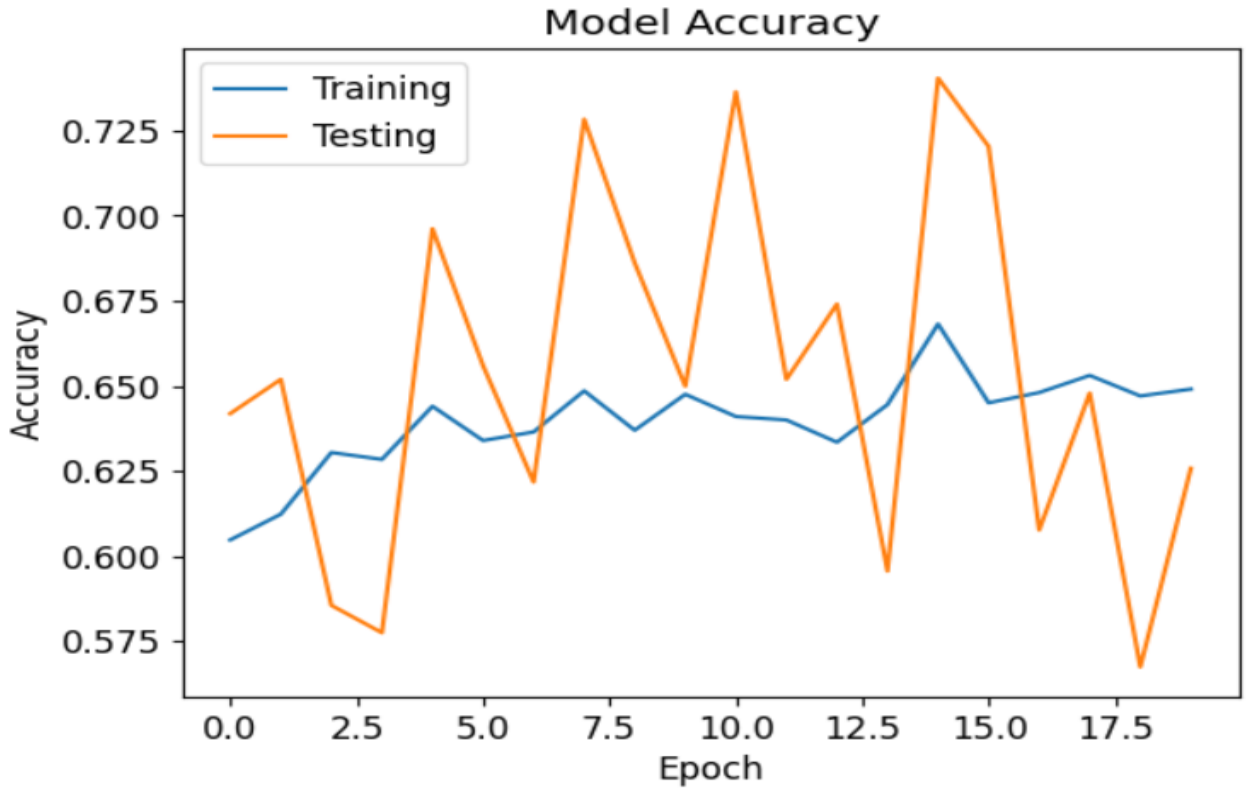
Model correctly classified 76% of the COVID cases as COVID. while the remaining 24% were incorrectly classified as non-COVID and model correctly classified 72% of the non-COVID cases as non-COVID. while the remaining 28% were incorrectly classified as COVID.

ROC Curve:



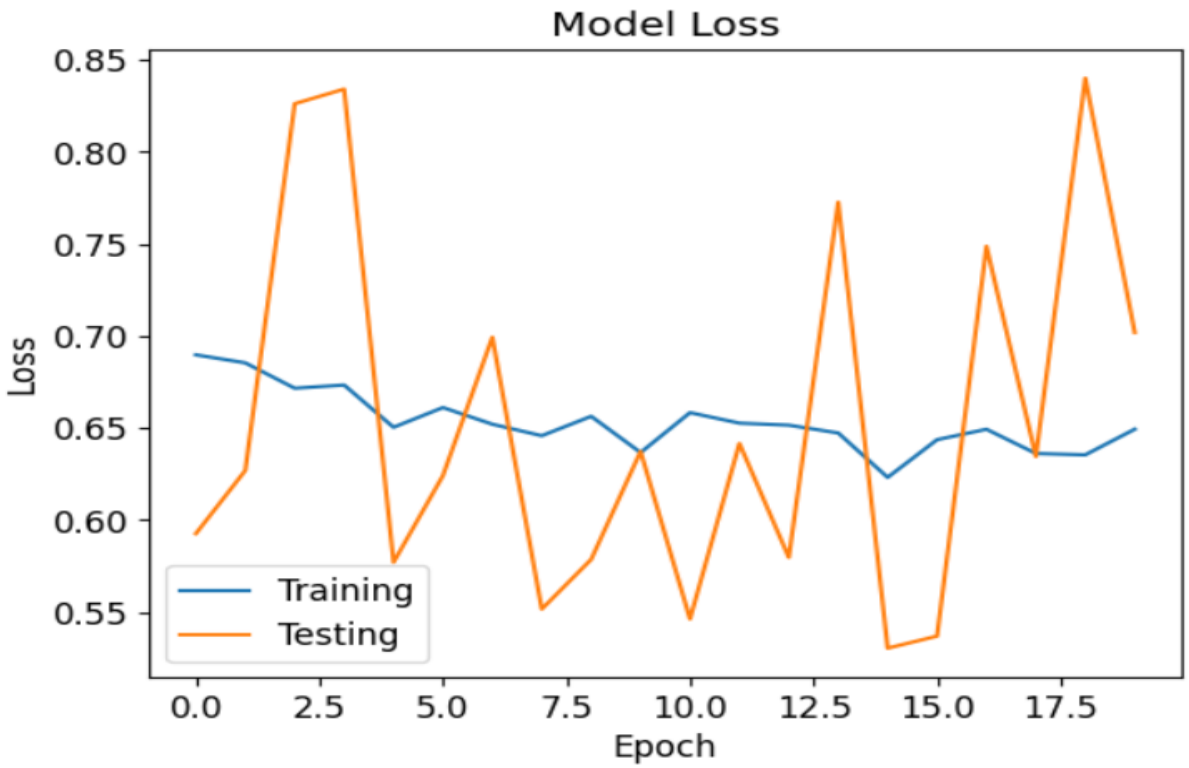
Model performance is good, as it indicates a higher TPR for a given FPR.

Model Accuracy Plot:



Increasing accuracy slightly for training. rising trend indicates that the model's performance is improving.

Model Loss Plot:



Decreasing Training loss. decreasing trend indicates that the model's predictions are getting closer to the actual values.

Conclusion:

Using ResNet50 model on CT Scan image classification, model correctly predicted the outcome of the data in 75% of the cases. Model correctly classified 76% of the COVID cases as COVID.