Detecting COVID-19 in X-ray Images using VGG16 CNN Model

In [1]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os

%matplotlib inline
```

In [2]:

```
#initialize the initial learning rate, number of epochs to train for and batch size
INIT_LR = 1e-3
EPOCHS = 25
BS = 8
```

In [5]:

```
# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("Loading images...")
imagePaths = list(paths.list_images(r"E:\AAIC\Case Studies\keras-covid-19\dataset"))
data = []
labels = []

# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]

    # load the image, swap color channels, and resize it to be a fixed
    # 224x224 pixels while ignoring aspect ratio
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))

    # update the data and labels lists, respectively
    data.append(image)
    labels.append(label)
```

```
Loading images...
```

In [10]:

```
# convert the data and labels to NumPy arrays while scaling the pixel
# intensities to the range [0, 255]
data = np.array(data) / 255.0
labels = np.array(labels)
```

In [11]:

```
data
```

Out[11]:

```
array([[[[0.17254902, 0.16470588, 0.16862745],
         [0.20784314, 0.2       , 0.20392157],
         [0.22352941, 0.21568627, 0.21960784],
         ...,
         [0.36078431, 0.36078431, 0.36078431],
         [0.28235294, 0.28235294, 0.28235294],
         [0.21176471, 0.20392157, 0.21568627]],

        [[0.21176471, 0.20392157, 0.20784314],
         [0.21960784, 0.21176471, 0.21568627],
         [0.23137255, 0.22352941, 0.23137255],
         ...,
         [0.25882353, 0.25098039, 0.2627451 ],
         [0.27843137, 0.2745098 , 0.28235294],
         [0.27843137, 0.28235294, 0.29019608]],

        [[0.21176471, 0.20392157, 0.21568627],
         [0.22352941, 0.22352941, 0.23137255],
         [0.23529412, 0.23529412, 0.23529412],
         ...,
         [0.35686275, 0.35686275, 0.36470588],
         [0.24313725, 0.24313725, 0.25098039],
         [0.18431373, 0.17647059, 0.18823529]],

        ...,

        [[0.2745098 , 0.2745098 , 0.28235294],
         [0.29411765, 0.29411765, 0.30196078],
         [0.27058824, 0.27058824, 0.27843137],
         ...,
         [0.26666667, 0.26666667, 0.2745098 ],
         [0.35294118, 0.35294118, 0.36078431],
         [0.35686275, 0.35686275, 0.36470588]],

        [[0.31372549, 0.31372549, 0.32156863],
         [0.29019608, 0.29019608, 0.29803922],
         [0.2745098 , 0.2745098 , 0.28235294],
         ...,
         [0.28627451, 0.28627451, 0.29411765],
         [0.3372549 , 0.3372549 , 0.34509804],
         [0.36862745, 0.36862745, 0.37647059]],

        [[0.30196078, 0.30196078, 0.30196078],
         [0.23921569, 0.23921569, 0.25098039],
         [0.26666667, 0.26666667, 0.2745098 ],
         ...,
         [0.30980392, 0.30980392, 0.31764706],
         [0.40392157, 0.40392157, 0.41176471],
         [0.38823529, 0.38823529, 0.39607843]]],


       [[[0.04705882, 0.04705882, 0.04705882],
         [0.05098039, 0.05098039, 0.05098039],
         [0.04705882, 0.04705882, 0.04705882],
         ...,
         [0.04705882, 0.04705882, 0.04705882],
         [0.04705882, 0.04705882, 0.04705882],
         [0.05098039, 0.05098039, 0.05098039]],

        [[0.05098039, 0.05098039, 0.05098039],
         [0.05098039, 0.05098039, 0.05098039],
         [0.05490196, 0.05490196, 0.05490196],
         ...,
         [0.05098039, 0.05098039, 0.05098039],
         [0.04705882, 0.04705882, 0.04705882],
         [0.04705882, 0.04705882, 0.04705882]],

        [[0.04705882, 0.04705882, 0.04705882],
         [0.04705882, 0.04705882, 0.04705882],
         [0.46666667, 0.46666667, 0.46666667],
         ...,
         [0.05098039, 0.05098039, 0.05098039],
         [0.04705882, 0.04705882, 0.04705882],
  [0.04705882, 0.04705882, 0.04705882],
  [0.04705882, 0.04705882, 0.04705882]],

 ...,

 [[0.03921569, 0.03921569, 0.03921569],
  [0.04313725, 0.04313725, 0.04313725],
  [0.03921569, 0.03921569, 0.03921569],
  ...,
  [0.04313725, 0.04313725, 0.04313725],
  [0.04313725, 0.04313725, 0.04313725],
  [0.04313725, 0.04313725, 0.04313725]],

 [[0.03921569, 0.03921569, 0.03921569],
  [0.03921569, 0.03921569, 0.03921569],
  [0.03921569, 0.03921569, 0.03921569],
  ...,
  [0.03921569, 0.03921569, 0.03921569],
  [0.04313725, 0.04313725, 0.04313725],
  [0.04313725, 0.04313725, 0.04313725]],

 [[0.04313725, 0.04313725, 0.04313725],
  [0.03921569, 0.03921569, 0.03921569],
  [0.03921569, 0.03921569, 0.03921569],
  ...,
  [0.04313725, 0.04313725, 0.04313725],
  [0.04313725, 0.04313725, 0.04313725],
  [0.04313725, 0.04313725, 0.04313725]]],


[[[0.        , 0.        , 0.        ],
  [0.00392157, 0.00392157, 0.00392157],
  [0.        , 0.        , 0.        ],
  ...,
  [0.        , 0.        , 0.        ],
  [0.00392157, 0.00392157, 0.00392157],
  [0.89019608, 0.89019608, 0.89019608]],

 [[0.00392157, 0.00392157, 0.00392157],
  [0.00392157, 0.00392157, 0.00392157],
  [1.        , 1.        , 1.        ],
  ...,
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.88627451, 0.88627451, 0.88627451]],

 [[0.00392157, 0.00392157, 0.00392157],
  [0.00392157, 0.00392157, 0.00392157],
  [1.        , 1.        , 1.        ],
  ...,
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.88627451, 0.88627451, 0.88627451]],

 ...,

 [[0.        , 0.        , 0.        ],
  [0.00392157, 0.00392157, 0.00392157],
  [0.        , 0.        , 0.        ],
  ...,
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.88627451, 0.88627451, 0.88627451]],

 [[0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  ...,
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.89019608, 0.89019608, 0.89019608]],

 [[0.        , 0.        , 0.        ],
  [0.00392157, 0.00392157, 0.00392157],
  [0.        , 0.        , 0.        ],
  ...,
  [0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ],
         [0.89019608, 0.89019608, 0.89019608]]],


       ...,


       [[[0.3254902 , 0.3254902 , 0.3254902 ],
         [0.3254902 , 0.3254902 , 0.3254902 ],
         [0.3372549 , 0.3372549 , 0.3372549 ],
         ...,
         [0.45490196, 0.45490196, 0.45490196],
         [0.45098039, 0.45098039, 0.45098039],
         [0.44313725, 0.44313725, 0.44313725]],

        [[0.31764706, 0.31764706, 0.31764706],
         [0.31764706, 0.31764706, 0.31764706],
         [0.33333333, 0.33333333, 0.33333333],
         ...,
         [0.48235294, 0.48235294, 0.48235294],
         [0.47058824, 0.47058824, 0.47058824],
         [0.44705882, 0.44705882, 0.44705882]],

        [[0.30196078, 0.30196078, 0.30196078],
         [0.33333333, 0.33333333, 0.33333333],
         [0.3254902 , 0.3254902 , 0.3254902 ],
         ...,
         [0.41176471, 0.41176471, 0.41176471],
         [0.44705882, 0.44705882, 0.44705882],
         [0.49411765, 0.49411765, 0.49411765]],

        ...,

        [[0.01568627, 0.01568627, 0.01568627],
         [0.01568627, 0.01568627, 0.01568627],
         [0.01176471, 0.01176471, 0.01176471],
         ...,
         [0.25882353, 0.25882353, 0.25882353],
         [0.18823529, 0.18823529, 0.18823529],
         [0.14117647, 0.14117647, 0.14117647]],

        [[0.01568627, 0.01568627, 0.01568627],
         [0.01568627, 0.01568627, 0.01568627],
         [0.01176471, 0.01176471, 0.01176471],
         ...,
         [0.26666667, 0.26666667, 0.26666667],
         [0.20392157, 0.20392157, 0.20392157],
         [0.14509804, 0.14509804, 0.14509804]],

        [[0.01176471, 0.01176471, 0.01176471],
         [0.01176471, 0.01176471, 0.01176471],
         [0.01176471, 0.01176471, 0.01176471],
         ...,
         [0.25882353, 0.25882353, 0.25882353],
         [0.22352941, 0.22352941, 0.22352941],
         [0.15686275, 0.15686275, 0.15686275]]],


       [[[0.17647059, 0.17647059, 0.17647059],
         [0.17254902, 0.17254902, 0.17254902],
         [0.19607843, 0.19607843, 0.19607843],
         ...,
         [0.15686275, 0.15686275, 0.15686275],
         [0.16078431, 0.16078431, 0.16078431],
         [0.16078431, 0.16078431, 0.16078431]],

        [[0.17647059, 0.17647059, 0.17647059],
         [0.18039216, 0.18039216, 0.18039216],
         [0.20392157, 0.20392157, 0.20392157],
         ...,
         [0.16470588, 0.16470588, 0.16470588],
         [0.16470588, 0.16470588, 0.16470588],
         [0.16470588, 0.16470588, 0.16470588]],

        [[0.18039216, 0.18039216, 0.18039216],
         [0.18431373, 0.18431373, 0.18431373],
         [0.22352941, 0.22352941, 0.22352941],
  ...,
  [0.17254902, 0.17254902, 0.17254902],
  [0.16862745, 0.16862745, 0.16862745],
  [0.16862745, 0.16862745, 0.16862745]],

 ...,

 [[0.09803922, 0.09803922, 0.09803922],
  [0.14117647, 0.14117647, 0.14117647],
  [0.16078431, 0.16078431, 0.16078431],
  ...,
  [0.11372549, 0.11372549, 0.11372549],
  [0.11372549, 0.11372549, 0.11372549],
  [0.11372549, 0.11372549, 0.11372549]],

 [[0.09803922, 0.09803922, 0.09803922],
  [0.14509804, 0.14509804, 0.14509804],
  [0.16470588, 0.16470588, 0.16470588],
  ...,
  [0.11372549, 0.11372549, 0.11372549],
  [0.11372549, 0.11372549, 0.11372549],
  [0.11372549, 0.11372549, 0.11372549]],

 [[0.09803922, 0.09803922, 0.09803922],
  [0.14509804, 0.14509804, 0.14509804],
  [0.16862745, 0.16862745, 0.16862745],
  ...,
  [0.11372549, 0.11372549, 0.11372549],
  [0.11372549, 0.11372549, 0.11372549],
  [0.11372549, 0.11372549, 0.11372549]]],


[[[0.02352941, 0.02352941, 0.02352941],
  [0.03529412, 0.03529412, 0.03529412],
  [0.04313725, 0.04313725, 0.04313725],
  ...,
  [0.0627451 , 0.0627451 , 0.0627451 ],
  [0.05098039, 0.05098039, 0.05098039],
  [0.03529412, 0.03529412, 0.03529412]],

 [[0.01960784, 0.01960784, 0.01960784],
  [0.03529412, 0.03529412, 0.03529412],
  [0.04313725, 0.04313725, 0.04313725],
  ...,
  [0.07058824, 0.07058824, 0.07058824],
  [0.05098039, 0.05098039, 0.05098039],
  [0.03529412, 0.03529412, 0.03529412]],

 [[0.01568627, 0.01568627, 0.01568627],
  [0.02745098, 0.02745098, 0.02745098],
  [0.03921569, 0.03921569, 0.03921569],
  ...,
  [0.0627451 , 0.0627451 , 0.0627451 ],
  [0.03921569, 0.03921569, 0.03921569],
  [0.03137255, 0.03137255, 0.03137255]],

 ...,

 [[0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  ...,
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ]],

 [[0.01960784, 0.01960784, 0.01960784],
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  ...,
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
  [0.01960784, 0.01960784, 0.01960784]],

 [[0.61568627, 0.61568627, 0.61568627],
  [0.        , 0.        , 0.        ],
  [0.        , 0.        , 0.        ],
         ...,
         [0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ],
         [0.61568627, 0.61568627, 0.61568627]]]])
```

In [12]:

```
np.unique(labels)
```

Out[12]:

```
array(['covid', 'normal'], dtype='<U6')
```

In [13]:

```
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
```

In [14]:

```
np.unique(labels)
```

Out[14]:

```
array([0., 1.], dtype=float32)
```

In [15]:

```
# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels,test_size=0.20, stratify=labels, random_state=42)
```

In [16]:

```
# initialize the training data augmentation object
trainAug = ImageDataGenerator(rotation_range=15,fill_mode="nearest")
```

In [17]:

```
# load the VGG16 network, ensuring the head FC layer sets are left off
baseModel = VGG16(weights="imagenet", include_top=False,input_tensor=Input(shape=(224, 224, 3)))
```

```
WARNING:tensorflow:From C:\Users\sanjeev\Anaconda3\lib\site-
packages\tensorflow\python\ops\init_ops.py:1251: calling VarianceScaling.__init__ (from
tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Downloading data from https://github.com/fchollet/deep-learning-
models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 51s 1us/step
```

In [18]:

```
# construct the head of the model that will be placed on top of the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
```

In [19]:

```
# place the head FC model on top of the base model (this will become the actual model we will train)
```

```
model = Model(inputs=baseModel.input, outputs=headModel)
```

In [20]:

```
# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False
```

In [21]:

```
# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
```

```
[INFO] compiling model...
```

In [22]:

```
# train the head of the network
print("[INFO] training head...")
H = model.fit_generator(trainAug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)
```

```
[INFO] training head...
Epoch 1/25
WARNING:tensorflow:From C:\Users\sanjeev\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_grad.py:1250: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
5/5 [==============================] - 21s 4s/step - loss: 0.8538 - acc: 0.4500 - val_loss: 0.6208
- val_acc: 0.7000
Epoch 2/25
5/5 [==============================] - 17s 3s/step - loss: 0.7020 - acc: 0.5750 - val_loss: 0.6476
- val_acc: 0.5000
Epoch 3/25
5/5 [==============================] - 17s 3s/step - loss: 0.7348 - acc: 0.5000 - val_loss: 0.5904
- val_acc: 0.8000
Epoch 4/25
5/5 [==============================] - 17s 3s/step - loss: 0.7000 - acc: 0.6000 - val_loss: 0.5649
- val_acc: 1.0000
Epoch 5/25
5/5 [==============================] - 17s 3s/step - loss: 0.5823 - acc: 0.7000 - val_loss: 0.5545
- val_acc: 1.0000
Epoch 6/25
5/5 [==============================] - 17s 3s/step - loss: 0.7666 - acc: 0.5250 - val_loss: 0.5381
- val_acc: 1.0000
Epoch 7/25
5/5 [==============================] - 17s 3s/step - loss: 0.6015 - acc: 0.6000 - val_loss: 0.5273
- val_acc: 1.0000
Epoch 8/25
5/5 [==============================] - 17s 3s/step - loss: 0.5249 - acc: 0.8000 - val_loss: 0.5215
- val_acc: 1.0000
Epoch 9/25
5/5 [==============================] - 17s 3s/step - loss: 0.4750 - acc: 0.8250 - val_loss: 0.5160
- val_acc: 1.0000
Epoch 10/25
5/5 [==============================] - 18s 4s/step - loss: 0.4455 - acc: 0.8500 - val_loss: 0.5063
- val_acc: 1.0000
Epoch 11/25
5/5 [==============================] - 17s 3s/step - loss: 0.5747 - acc: 0.7500 - val_loss: 0.4855
- val_acc: 1.0000
Epoch 12/25
5/5 [==============================] - 17s 3s/step - loss: 0.5493 - acc: 0.7500 - val_loss: 0.4694
- val_acc: 1.0000
Epoch 13/25
5/5 [==============================] - 17s 3s/step - loss: 0.5238 - acc: 0.7750 - val_loss: 0.4592
- val_acc: 1.0000
       - val_acc: 1.0000
       Epoch 14/25
       5/5 [==============================] - 17s 3s/step - loss: 0.4610 - acc: 0.8000 - val_loss: 0.4629
       - val_acc: 0.8000
       Epoch 15/25
       5/5 [==============================] - 17s 3s/step - loss: 0.4368 - acc: 0.8250 - val_loss: 0.4578
       - val_acc: 0.8000
       Epoch 16/25
       5/5 [==============================] - 17s 3s/step - loss: 0.4241 - acc: 0.8250 - val_loss: 0.4496
       - val_acc: 0.8000
       Epoch 17/25
       5/5 [==============================] - 18s 4s/step - loss: 0.4187 - acc: 0.8500 - val_loss: 0.4371
       - val_acc: 0.8000
       Epoch 18/25
       5/5 [==============================] - 18s 4s/step - loss: 0.3898 - acc: 0.9000 - val_loss: 0.4118
       - val_acc: 1.0000
       Epoch 19/25
       5/5 [==============================] - 17s 3s/step - loss: 0.4021 - acc: 0.8250 - val_loss: 0.3981
       - val_acc: 1.0000
       Epoch 20/25
       5/5 [==============================] - 17s 3s/step - loss: 0.4081 - acc: 0.8750 - val_loss: 0.3900
       - val_acc: 1.0000
       Epoch 21/25
       5/5 [==============================] - 17s 3s/step - loss: 0.3874 - acc: 0.8750 - val_loss: 0.4069
       - val_acc: 0.8000
       Epoch 22/25
       5/5 [==============================] - 17s 3s/step - loss: 0.3649 - acc: 0.9500 - val_loss: 0.3881
       - val_acc: 0.9000
       Epoch 23/25
       5/5 [==============================] - 17s 3s/step - loss: 0.3485 - acc: 0.8750 - val_loss: 0.3711
       - val_acc: 1.0000
       Epoch 24/25
       5/5 [==============================] - 18s 4s/step - loss: 0.3459 - acc: 0.9500 - val_loss: 0.3505
       - val_acc: 1.0000
       Epoch 25/25
       5/5 [==============================] - 19s 4s/step - loss: 0.3276 - acc: 0.8750 - val_loss: 0.3392
       - val_acc: 1.0000
```

In [23]:

```
# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
```

```
[INFO] evaluating network...
```

In [24]:

```
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
```

In [31]:

```
predIdxs
```

Out[31]:

```
array([1, 1, 1, 1, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [33]:

```
testY.argmax(axis=1)
```

Out[33]:

```
array([1, 1, 1, 1, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [25]:

```
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))
print(classification_report(testY.argmax(axis=1), predIdxs,target_names=lb.classes_))
```

```
              precision    recall  f1-score   support

       covid       1.00      1.00      1.00         5
      normal       1.00      1.00      1.00         5

    accuracy                           1.00        10
   macro avg       1.00      1.00      1.00        10
weighted avg       1.00      1.00      1.00        10
```

In [26]:

```
# compute the confusion matrix and and use it to derive the raw
# accuracy, sensitivity, and specificity
cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
```

In [27]:

```
# show the confusion matrix, accuracy, sensitivity, and specificity
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
[[5 0]
 [0 5]]
acc: 1.0000
sensitivity: 1.0000
specificity: 1.0000
```

In [59]:

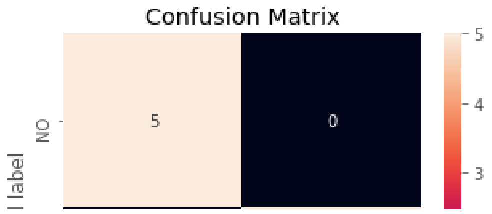
```
def plot_Confusion_Matrix(actual_labels,predict_labels,title):
    """This function plot the confusion matrix"""
    # Reference : https://seaborn.pydata.org/generated/seaborn.heatmap.html
    cm = confusion_matrix(actual_labels, predict_labels)
    classNames = ['NO','YES']
    cm_data = pd.DataFrame(cm,index = classNames,
                  columns = classNames)
    plt.figure(figsize = (5,4))
    sns.heatmap(cm_data, annot=True,fmt="d")
    plt.title(title)
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()
```

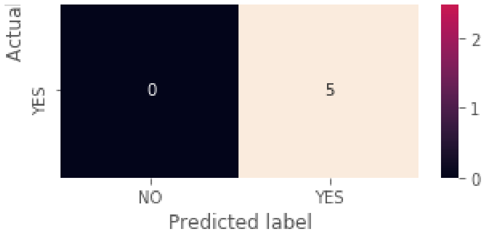
In [62]:

```
import pandas as pd
import seaborn as sns

plot_Confusion_Matrix(testY.argmax(axis=1), predIdxs,"Confusion Matrix")
```

```
C:\Users\sanjeev\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```



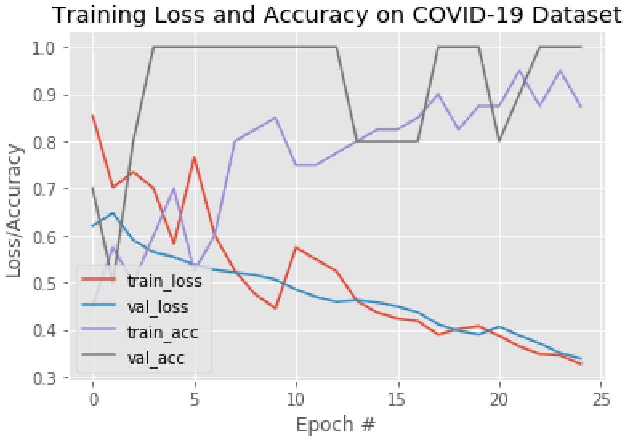


In [56]:

```
# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
```

Out[56]:

```
<matplotlib.legend.Legend at 0x231118dbdd8>
```



In [58]:

```
plt.savefig('plot.jpeg')
```

```
<Figure size 432x288 with 0 Axes>
```

In [34]:

```
# serialize the model to disk
print("[INFO] saving COVID-19 detector model...")
model.save("COVID19_Detection.hdf5", save_format="hdf5")
```

```
[INFO] saving COVID-19 detector model...
```

In [48]:

```
# test the model
image = cv2.imread(r'E:\AAIC\Case Studies\keras-covid-19\dataset\covid\nejmoa2001191_f1-PA.jpeg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image = cv2.resize(image, (224, 224))
```

In [49]:

```
test = []
```

```
test.append(image)
```

In [50]:

```
test = np.array(test)/255.0
```

In [51]:

```
test
```

Out[51]:

```
array([[[[0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         ...,
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941]],

        [[0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         ...,
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941]],

        [[0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         ...,
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941]],

        ...,

        [[0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         ...,
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941]],

        [[0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         ...,
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941]],

        [[0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         ...,
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941],
         [0.02352941, 0.02352941, 0.02352941]]]])
```

In [52]:

```
pred = model.predict(test, batch_size=BS)
```

In [53]:

```
predIds = np.argmax(pred, axis=1)
```

In [54]:

```
predIds
```

Out[54]:

```
array([0], dtype=int64)
```

In [55]:

```
print(lb.classes_[predIds])
```

```
['covid']
```

In [57]:

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
average_pooling2d (AveragePo (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
=================================================================
Total params: 14,747,650
Trainable params: 32,962
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [ ]:

```

```