International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

DETECTION OF COVID-19 FROM CHEST X-RAY IMAGES USING MACHINE LEARNING

SUBASRI G[1], VIJETHA K[2], VISHAL V[3,] ANITHA M[4]

[1,2,3] UG Scholar, Department of CSE, Kingston College, Vellore-59
[4]Asst.Professor, Department of CSE, Kingston College, Vellore-59

ABSTRACT:
Novel corona virus is fast spreading pathogen worldwide and is threatening billions of lives. SARS n-CoV2 is known to affect the lungs of the COVID-19 positive patients. Chest x-rays are the most widely used imaging technique for clinical diagnosis due to fast imaging time and low cost. The purpose of this study is touse deep learning technique for automatic detection of COVID-19 using chest x-rays.

Chest X-ray is the first imaging technique that plays an important role in the diagnosis of COVID-19 disease. Due to the high availability of large-scale annotated image datasets, great success has been achieved using convolutional neural networks for image recognition and classification. However, due to the limited availability of annotated medical images, the classification of medical images remains the biggest challenge in medical diagnosis. Thus, transfer learning, an effective mechanism that canprovide a promising solution by transferring knowledge from generic object recognitiontasks to domain-specific tasks.

The main objective is to predictive analytics model to diagnose malignant or benign with selecting the highest accuracy result of supervised machine learning algorithm to improvise the prediction. Additionally, discuss the performance from the given hospital dataset with evaluation classification report and identify the confusion matrix. The data validation, data cleaning/preparing and data visualization will be doneon the entire given dataset and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with precision, Recall and F1 Score.

Keywords-chest X-ray radiographs; COVID-19; biomedical imaging; deep learning; deep convolutional neural network,RestNet50, Inception, Xception.

INTRODUCTION

Coronaviruses (CoV) are perilous viruses that may cause Severe Acute Respiratory Syndrome (SARS-CoV), Middle East Respiratory Syndrome (MERS-CoV). The novel 2019 Coronavirus disease (COVID-19) was discovered as a novel disease pneumonia in the city of Wuhan, China at the end of 2019. Now, it becomes a Coronavirus outbreak around the world, the number of infected people and deaths are increasing rapidly every day according to the updated reports of the World Health Organization (WHO). The aim of the project is to assist radiologists to automatically diagnose COVID-19 in X-ray images using deep neural frameworks.

The data set containing confirmed COVID-19 positive. The model used a pre-trained VGG-16 model for

the classification task. Transfer learning with fine- tuning was used in this study to effectively train the network on a relatively small chestx-ray data set. Initial experiments show that the model achieves promising results and can be greatly used to expedite COVID-19 detection. The exponential increase in COVID-19 patients is overwhelming healthcare systems across the world.

PREFACE

With limited testing kits, it is impossible for everypatient with respiratory illness to be tested using conventional techniques (RT-PCR). The tests also have long turn-around time, and limited sensitivity. Detecting possible COVID-19 infections on Chest X-Ray may help quarantine high risk patients while testresults are awaited. X-Ray machines are already available in most healthcare systems, and with most modern X-Ray systems already digitized, there is no transportation timeinvolved for the samples either. In this work we propose the use of chest X-Ray to prioritize the selection of patients for further RT-PCR testing. This may be useful in aninpatient setting where the present systems are struggling to decide whether to keep thepatient in the ward along with other patients or isolate them in COVID-19 areas. It would also help in identifying patients with high likelihood of COVID with a false negative RT-PCR who would need repeat testing. Further, we propose the use of modern AI techniques to detect the COVID-19 patients using X-Ray images in an automated manner, particularly in settings where radiologists are not available, and helpmake the proposed testing technology scalable.

SCOPE OF THE PROJECT

Chest X-rays can give COVID-19 test results in just 10 minutes: Expert Accordingto Dr Vishal Rao, member of the Covid consultative group and onco surgeon who initiated the project, X-rays can be used to screen Covid-19 patients which is cost effective.

RELATED WORKS

[1] The work done by the authors Seung Hoon Yoo, Hui Geng, Tin Lok Chiu, Siu Ki Yu, Dae Chul Cho, Jin Heo,Min Sung Choi, Hyun Choi, Cong Cung Van, Nguen Viet Nhung, Byung Jun Min and Ho Lee, “Deep Learning-Based Decision-Tree Classifier for COVID-19 Diagnosis From Chest X-ray Imaging”.,in the year 2020.

The global pandemic of corona virus disease 2019 (COVID-19) hasresulted in an increased demand for testing, diagnosis, and treatment. Reverse transcription polymerase chain reaction (RT-PCR) is the definitive test for the diagnosisof COVID-19. Chest X-ray radiography is a fast, effective, and affordable test that identifies the possible COVID-19 related pneumonia. This study investigates the feasibility of using a deep learning based decision-tree classifier for detecting COVID-19 from Chest X-ray radiography images. The proposed classifier comprises three binary decision trees, each trained by a deep learning model with convolution neural network

based on the PyTorch frame.

[2] The work done by the authors Ezz El-Din Hemdan, Marwa A. Shouman, and Mohamed Esmail, "COVIDX-Net: A Framework of Deep Learning Classifiers to Diagnose COVID-19 in X-Ray Images" in the year 2020.

Coronaviruses are perilous viruses that may cause Severe Acute Respiratory Syndrome, Middle East Respiratory Syndrome. The aim of this article is to introduce anew deep learning framework, namely COVIDX-Net to assist radiologists to automatically diagnose COVID-19 in X-ray images. Due to the lack of public COVID-19 datasets, the study is validated on 50 Chest X-ray images with 25 confirmed positiveCOVID-19 cases. The COVIDX-Net includes seven different architectures of deep convolutional neural network models, such as modified Visual Geometry Group Network (VGG19) and the second version of Google MobileNet. Each deep neural network model is able to analyze the normalized intensities of the X-ray image to classify the patient status either negative or positive COVID-19 case.

[3] The work done by the authors, Sohaib Asif, Yi Wenhui, Hou Jin, Yi Tao, SizJinhai," Classification of COVID-19 from Chest X-ray images using DeepConvolutional Neural Networks" in the year 2020.

This study aimed to automatically detect COVID-19 pneumonia patients usingdigital chest x-ray images while maximizing the accuracy in detection using deep convolutional neural networks (DCNN). The dataset consists of 864 COVID-19, 1345 viral pneumonia and 1341 normal chest x-ray images. In this study, DCNN based modelInception V3 with transfer learning have been proposed for the detection of coronaviruspneumonia infected patients using chest X-ray radiographs and gives a classification accuracy of more than 98% (training accuracy of 97% and validation accuracy of 93%).The results demonstrate that transfer learning proved to be effective, showed robust performance and easily deployable approach for COVID-19 detection. DCNN typically perform better with a larger dataset than a smaller one. Transfer learning can be beneficial in those applications of CNN where the dataset is not large. The idea of transfer learning uses the trained model from large datasets such asImageNet is used for application with comparatively smaller dataset. This eliminates therequirement of having large dataset and also reduces the long training period as is required by the deep learning algorithm. The biggest advantage of using transfer learning method is that it allows the training of data with fewer datasets and requires less calculation costs. With the transferlearning method, which is widely used in the field of deep learning, the information gained by the pre-trained model on a large dataset is transferred to the model to be trained. The Inception V3 model which performs convolution, pooling, softmax and fully connected procedures.
METHODOLOGY:

1 Convolutional Neural Network:

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the

capabilities of humans and machines. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet ismuch lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used. CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w xd(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrixof RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

Technically, deep learning CNN models to train and test, each input image willpass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1.

4.4.2 VGG 16

This model achieves 92.7% top-5 test accuracy on ImageNet dataset which contains 14 million images belonging to 1000 classes. The ImageNet dataset contains images of fixed size of 224*224 and have RGB channels. So, we have a tensor of (224,224, 3) as our input. This model process the input image and outputs a vector of 1000 values.

The input to the network is image of dimensions (224, 224, 3). The first two layers have 64 channels of 3*3 filter size and same padding. Then after a max pool layer of stride (2, 2), two layers which have convolution layers of 256 filter size and filter size (3,3). This followed by a max pooling layer of stride (2, 2) which is same as previous layer. Then there are 2 convolution layers of filter size (3, 3) and 256 filter. After that there are 2 sets of 3 convolution layer and a max pool layer. Each have 512 filters of (3, 3)size with same padding.This image is then passed to the stack of two convolution layers. In these convolution and max pooling layers, the filters we use is of the size 3*3 instead of 11*11 in AlexNet and 7*7 in ZF-Net. In some of the layers, it also uses 1*1 pixel which is used to manipulate the number of input channels. There is a padding of 1-pixel

(same padding) done after each convolution layer to prevent the spatial feature of the image.

Step 1: Creating a Dataset contains two folders, in which one has sampled X-Ray imagesof Normal Patients. Another folder in which the X-Ray images of corona virus patients.

Step 2: After creating two folders we will merge the images and set the labels
Step 3: Developing a quality assessment model. Then we will split that into training andtesting set and creating a VGG model that will predict the data.

Step 4: Construct a fully connected layer and append it on the top of the VGG model.

Step 5: Training and experimentation on datasets

IMPLEMENTATION DETAILS:

1. Image preprocessing:

In the image processing module we are converting the color image to GrayScale imagebecause the color image increases the complexity of the model as it contains the three different layer image where as the GrayScale image contains only one layer of image.

2.Preparing training and test sets:

Preparing a training set, Applying the same preparation to a testing set, Controling thattrain and test sets have the same shape. Those sets will be fast, easy, Spliting Train andtest. To avoid introducing a bias in test using train-data, the train-test split should be performed before data preparation steps.

To simulate a train and test set we are going to split randomly this data set into 80% train and 20% test.

3.Image Transformations:

In this stage details which are not known, or we can say that interesting features of an image is highlighted. Such as brightness and contrast. Image restoration is the stage in which the appearance of an image is improved. Color image processing is a famous areabecause it has increased the use of digital images on the internet. This includes color modeling, processing in a digital domain. The label is assigned to the object, which is based on descriptors.

This module applies the classification algorithm for the identification of plant diseases.The deep learning algorithm namely Convolution Neural Network is used to train the model and to find the accuracy of the model.

4.Data Visualization:

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide anaccessible way to see and understand trends, outliers, and patterns in data. The right visualization of data must be paired with the right set of information.

5.Creating the Model:

- Creating a ResNet-18 model which is a convolutional neural network that is 18 layers deep. In which the network trained on more than a million images from the ImageNet database.
- The pre-trained model VGG-16 model for the classification task. Transfer learning with fine-tuning was used in this study to effectively train the network on a relativelysmall chest x-ray data set.
- Creating a CNN model for the dataset.

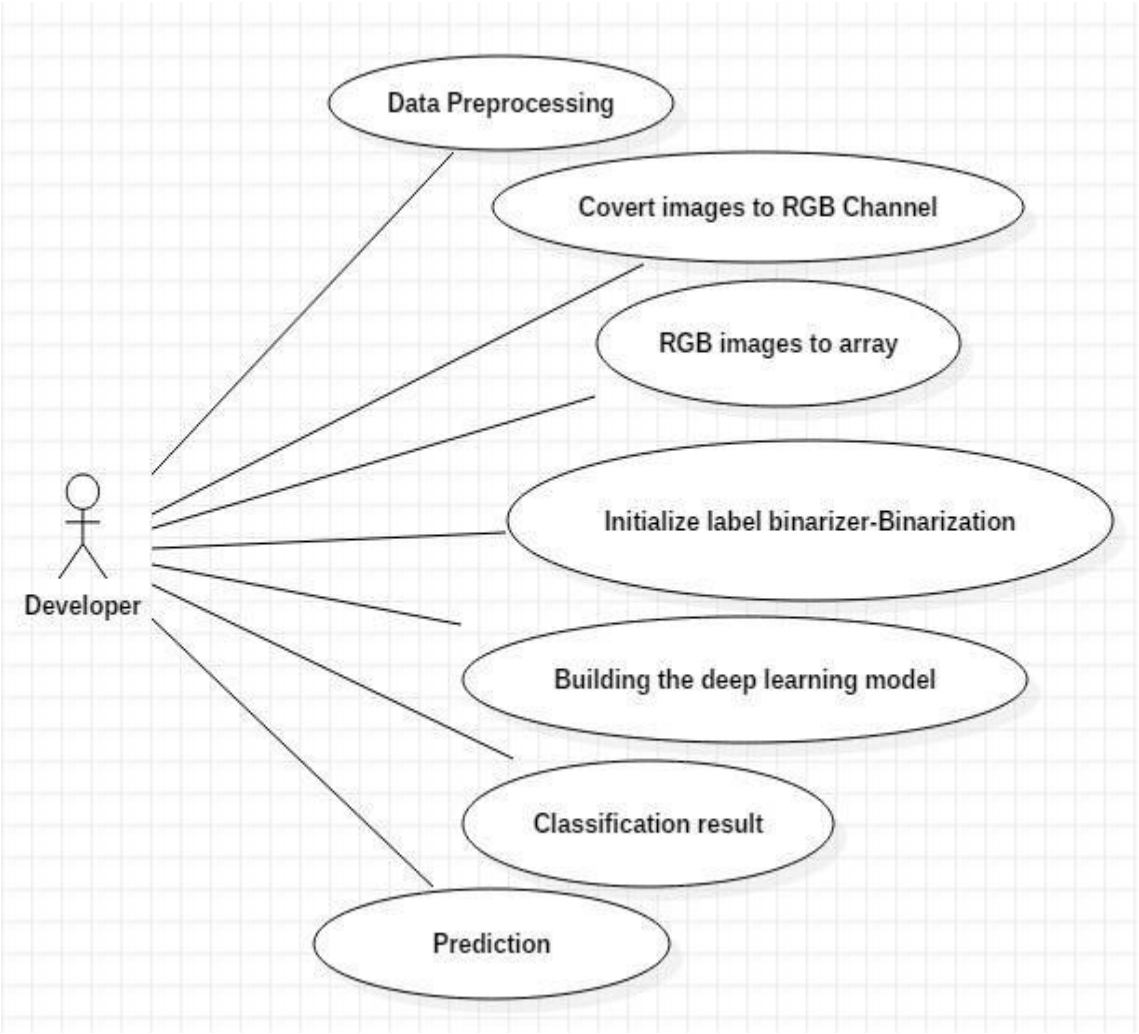
6. Training the Model:

The goal of training a model is to find a set of weights and biases that have lowloss, on average, training a model simply means learning (determining) good values forall the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to finda model that minimizes loss, this process is called empirical risk minimization.

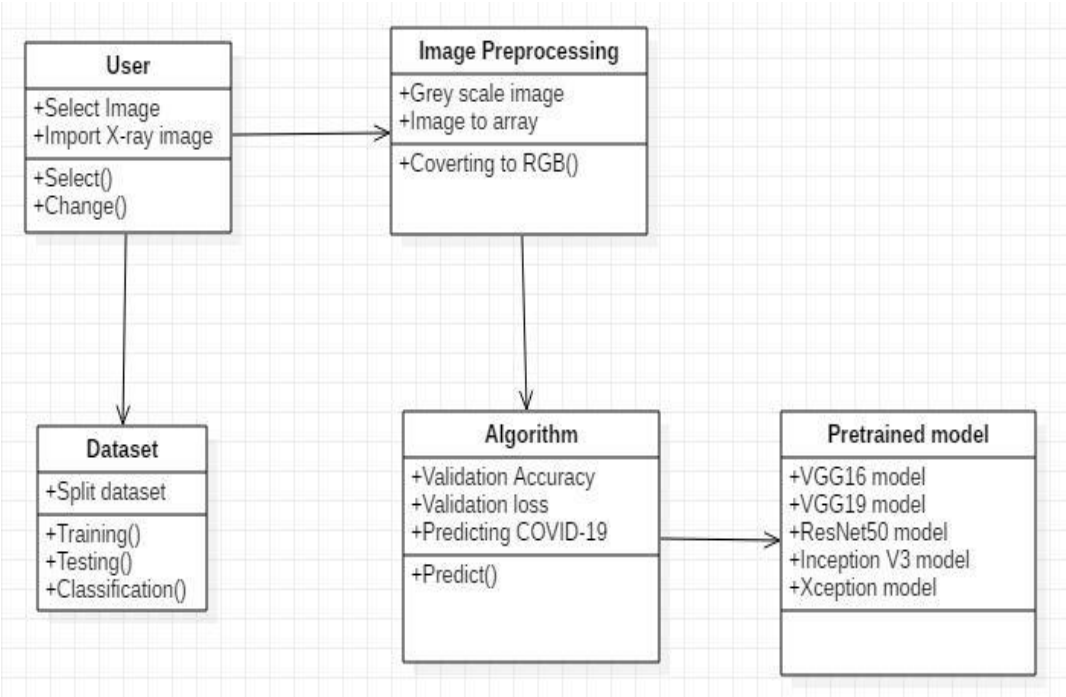
Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad themodel's prediction was on a single example. If the model's prediction is perfect, the lossis zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have low loss, on average.

5. SYSTEM DESIGN

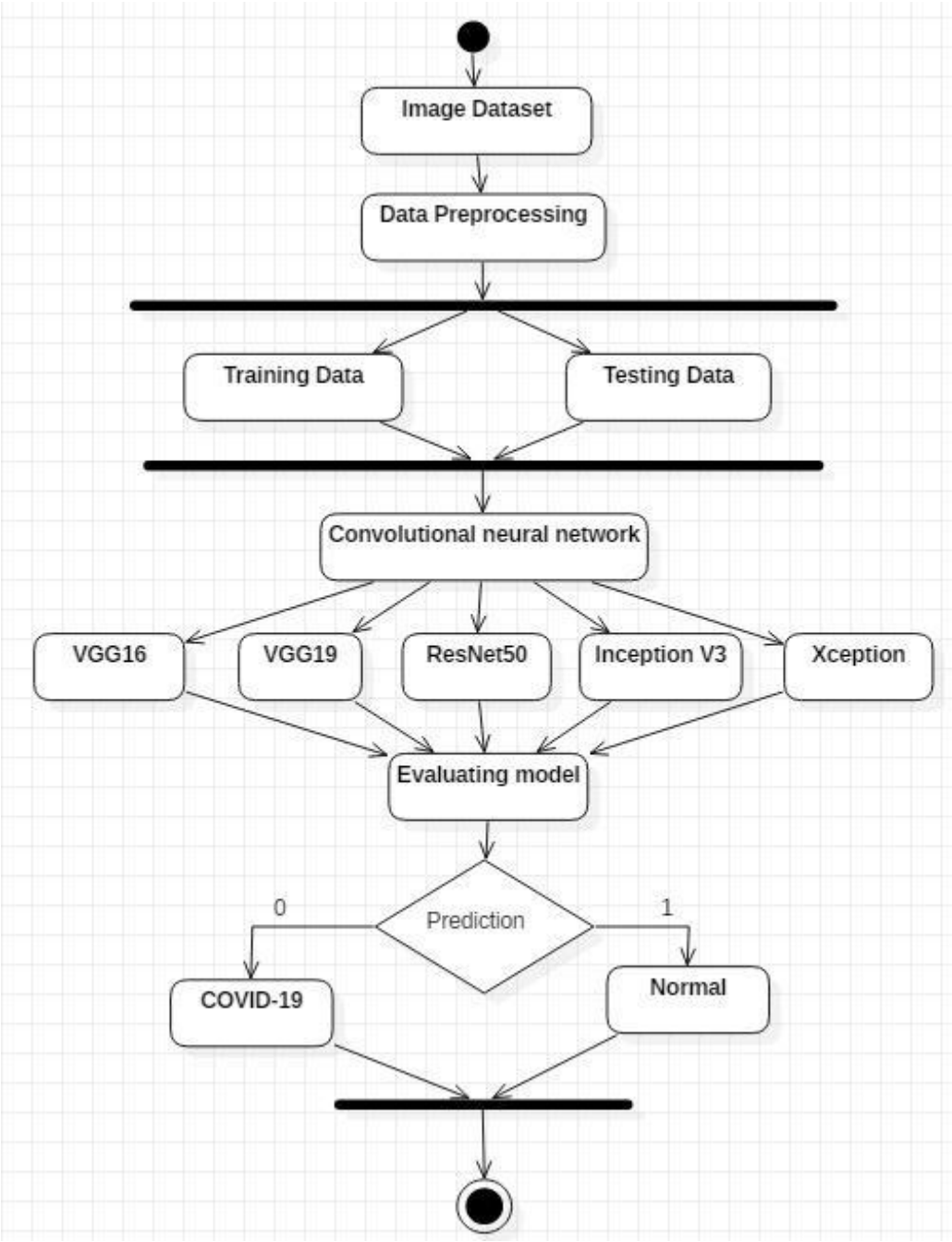
USECASE DIAGRAM



CLASS DIAGRAM



ACTIVITY DIAGRAM:



RESULTS:

VGG 16

```
Compiling Starts
WARNING:tensorflow:From <ipython-input-11-91be3d96ebe8>:4: Model.fit_generator (from tensorflow.python.keras.engine.training) i
s deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
20/20 [==============================] - 255s 13s/step - loss: 0.6633 - accuracy: 0.5875 - val_loss: 0.5372 - val_accuracy: 0.8
750
Epoch 2/10
20/20 [==============================] - 241s 12s/step - loss: 0.5162 - accuracy: 0.7563 - val_loss: 0.4309 - val_accuracy: 0.8
500
Epoch 3/10
20/20 [==============================] - 241s 12s/step - loss: 0.3860 - accuracy: 0.8875 - val_loss: 0.3422 - val_accuracy: 0.8
750
Epoch 4/10
20/20 [==============================] - 240s 12s/step - loss: 0.3323 - accuracy: 0.9062 - val_loss: 0.2652 - val_accuracy: 1.0
000
Epoch 5/10
20/20 [==============================] - 237s 12s/step - loss: 0.2433 - accuracy: 0.9250 - val_loss: 0.2115 - val_accuracy: 1.0
000
Epoch 6/10
20/20 [==============================] - 242s 12s/step - loss: 0.1940 - accuracy: 0.9625 - val_loss: 0.1828 - val_accuracy: 0.9
750
Epoch 7/10
20/20 [==============================] - 232s 12s/step - loss: 0.1613 - accuracy: 0.9563 - val_loss: 0.1489 - val_accuracy: 1.0
000
Epoch 8/10
20/20 [==============================] - 236s 12s/step - loss: 0.1447 - accuracy: 0.9625 - val_loss: 0.1418 - val_accuracy: 0.9
750
Epoch 9/10
20/20 [==============================] - 239s 12s/step - loss: 0.1432 - accuracy: 0.9688 - val_loss: 0.1145 - val_accuracy: 1.0
000
Epoch 10/10
20/20 [==============================] - 240s 12s/step - loss: 0.1423 - accuracy: 0.9688 - val_loss: 0.0976 - val_accuracy: 1.0
000
```

In [13]:
```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test, batch_size=BS)
y_pred = np.argmax(y_pred, axis=1)
print(classification_report(Y_test.argmax(axis=1), y_pred,target_names=LB.classes_))
```

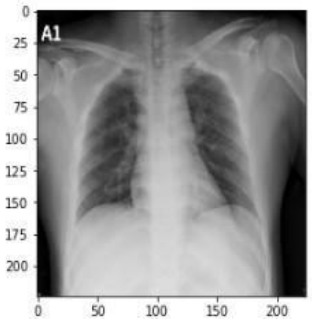
```
              precision    recall  f1-score   support

       Covid       1.00      1.00      1.00        20
      Normal       1.00      1.00      1.00        20

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

In [14]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test.argmax(axis=1), y_pred)
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])
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

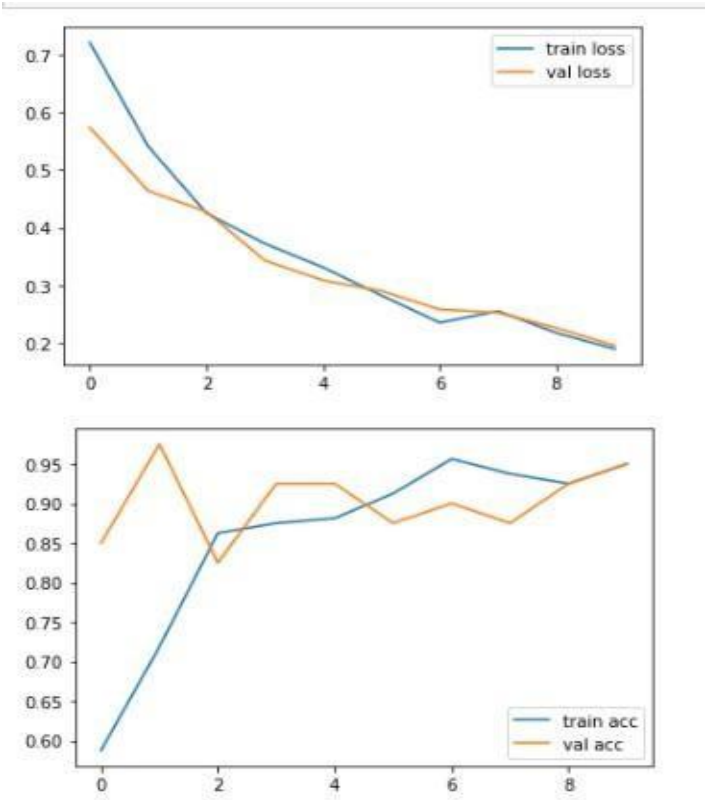
```
[[20  0]
 [ 0 20]]
acc: 1.0000
sensitivity: 1.0000
specificity: 1.0000
```

In [18]:
```
from keras.preprocessing import image
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
img = image.load_img(r'C:\Users\ELCOT\Desktop\Data\Covid\1-s2.0-S1684118220300682-main.pdf-002-a1.png', target_size=(224, 224))
imgplot = plt.imshow(img)
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)
classes = model.predict(img_data)
New_pred = np.argmax(classes, axis=1)
if New_pred==[1]:
  print('Prediction: Normal')
else:
  print('Prediction: Corona')
```

```
Prediction: Corona
```



VGG 19




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

In [12]:
```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test, batch_size=BS)
y_pred = np.argmax(y_pred, axis=1)
print(classification_report(Y_test.argmax(axis=1), y_pred,target_names=LB.classes_))
```

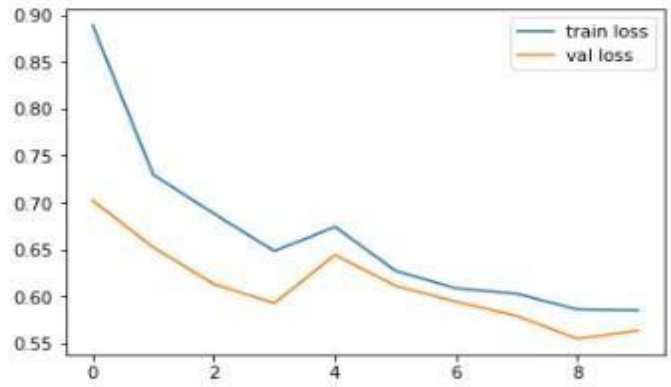
```
              precision    recall  f1-score   support

       Covid       1.00      0.90      0.95        20
      Normal       0.91      1.00      0.95        20

    accuracy                           0.95        40
   macro avg       0.95      0.95      0.95        40
weighted avg       0.95      0.95      0.95        40
```

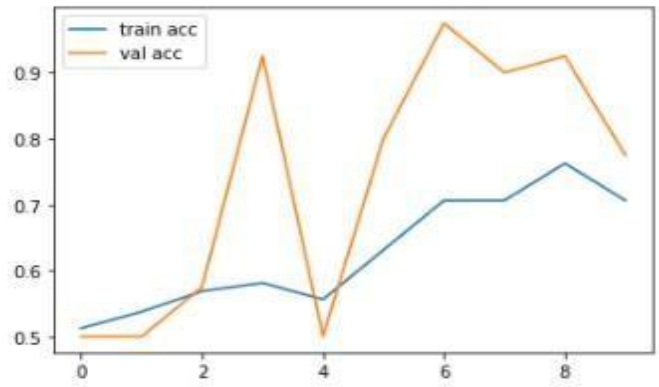
In [13]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test.argmax(axis=1), y_pred)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
[[18  2]
 [ 0 20]]
acc: 0.9500
sensitivity: 0.9000
specificity: 1.0000
```

RESNET 50

```
Compiling Starts
WARNING:tensorflow:From <ipython-input-11-91be3d96ebe8>:4: Model.fit_generator (from tensorflow.python.keras.engine.training) i
s deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
20/20 [==============================] - 85s 4s/step - loss: 0.8889 - accuracy: 0.5125 - val_loss: 0.7022 - val_accuracy: 0.500
0
Epoch 2/10
20/20 [==============================] - 78s 4s/step - loss: 0.7297 - accuracy: 0.5375 - val_loss: 0.6524 - val_accuracy: 0.500
0
Epoch 3/10
20/20 [==============================] - 78s 4s/step - loss: 0.6883 - accuracy: 0.5688 - val_loss: 0.6132 - val_accuracy: 0.575
0
Epoch 4/10
20/20 [==============================] - 83s 4s/step - loss: 0.6482 - accuracy: 0.5813 - val_loss: 0.5928 - val_accuracy: 0.925
0
Epoch 5/10
20/20 [==============================] - 78s 4s/step - loss: 0.6742 - accuracy: 0.5562 - val_loss: 0.6442 - val_accuracy: 0.500
0
Epoch 6/10
20/20 [==============================] - 79s 4s/step - loss: 0.6272 - accuracy: 0.6313 - val_loss: 0.6113 - val_accuracy: 0.800
0
Epoch 7/10
20/20 [==============================] - 80s 4s/step - loss: 0.6086 - accuracy: 0.7063 - val_loss: 0.5944 - val_accuracy: 0.975
0
Epoch 8/10
20/20 [==============================] - 79s 4s/step - loss: 0.6028 - accuracy: 0.7063 - val_loss: 0.5792 - val_accuracy: 0.900
0
Epoch 9/10
20/20 [==============================] - 78s 4s/step - loss: 0.5862 - accuracy: 0.7625 - val_loss: 0.5550 - val_accuracy: 0.925
0
Epoch 10/10
20/20 [==============================] - 79s 4s/step - loss: 0.5851 - accuracy: 0.7063 - val_loss: 0.5637 - val_accuracy: 0.775
0
```






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

In [13]:
```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test, batch_size=BS)
y_pred = np.argmax(y_pred, axis=1)
print(classification_report(Y_test.argmax(axis=1), y_pred,target_names=LB.classes_))
```

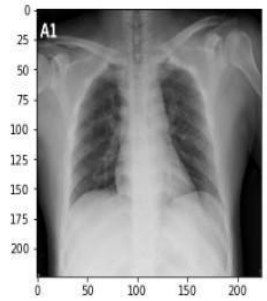
```
              precision    recall  f1-score   support

       Covid       1.00      0.55      0.71        20
      Normal       0.69      1.00      0.82        20

    accuracy                           0.78        40
   macro avg       0.84      0.78      0.76        40
weighted avg       0.84      0.78      0.76        40
```

In [14]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test.argmax(axis=1), y_pred)
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])
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
[[11  9]
 [ 0 20]]
acc: 0.7750
sensitivity: 0.5500
specificity: 1.0000
```

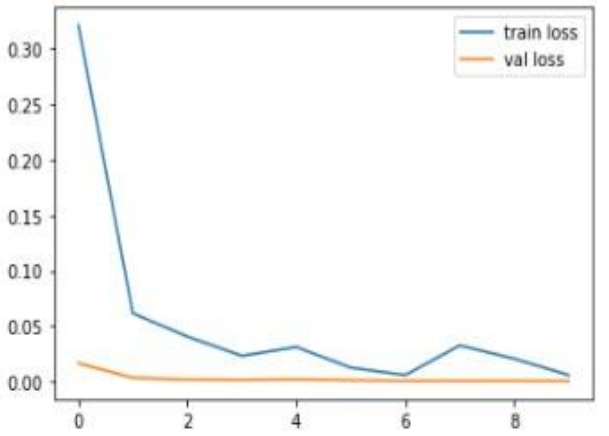
```
In [21]: from keras.preprocessing import image
         from keras.models import load_model
         from keras.applications.resnet50 import preprocess_input
         img = image.load_img(r'C:\Users\ELCOT\Desktop\Data\Covid\1-s2.0-S1684118220300682-main.pdf-002-a1.png', target_size=(224, 224)) #
         imgplot = plt.imshow(img)
         x = image.img_to_array(img)
         x = np.expand_dims(x, axis=0)
         img_data = preprocess_input(x)
         classes = model.predict(img_data)
         New_pred = np.argmax(classes, axis=1)
         if New_pred==[1]:
           print('Prediction: Normal')
         else:
           print('Prediction: Corona')
```

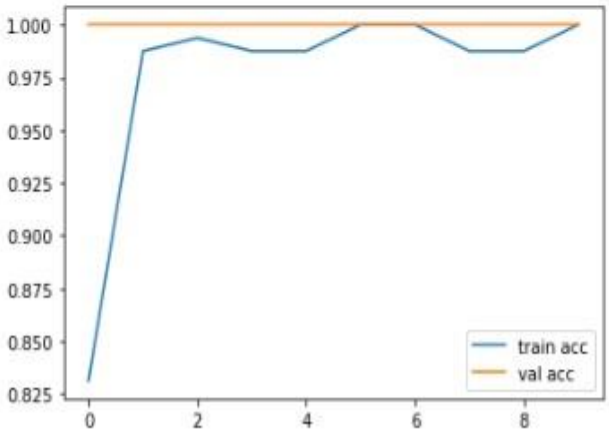
```
Prediction: Normal
```



Inception V3

```
Compiling Starts
WARNING:tensorflow:From <ipython-input-10-91be3d96ebe8>:4: Model.fit_generator (from tensorflow.python.keras.engine.training) i
s deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
20/20 [==============================] - 113s 6s/step - loss: 0.3214 - accuracy: 0.8313 - val_loss: 0.0164 - val_accuracy: 1.00
00
Epoch 2/10
20/20 [==============================] - 117s 6s/step - loss: 0.0614 - accuracy: 0.9875 - val_loss: 0.0033 - val_accuracy: 1.00
00
Epoch 3/10
20/20 [==============================] - 116s 6s/step - loss: 0.0405 - accuracy: 0.9937 - val_loss: 0.0017 - val_accuracy: 1.00
00
Epoch 4/10
20/20 [==============================] - 111s 6s/step - loss: 0.0232 - accuracy: 0.9875 - val_loss: 0.0013 - val_accuracy: 1.00
00
Epoch 5/10
20/20 [==============================] - 130s 7s/step - loss: 0.0311 - accuracy: 0.9875 - val_loss: 0.0020 - val_accuracy: 1.00
00
Epoch 6/10
20/20 [==============================] - 126s 6s/step - loss: 0.0125 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.00
00
Epoch 7/10
20/20 [==============================] - 125s 6s/step - loss: 0.0057 - accuracy: 1.0000 - val_loss: 5.1894e-04 - val_accuracy:
1.0000
Epoch 8/10
20/20 [==============================] - 116s 6s/step - loss: 0.0325 - accuracy: 0.9875 - val_loss: 5.0877e-04 - val_accuracy:
1.0000
Epoch 9/10
20/20 [==============================] - 128s 6s/step - loss: 0.0206 - accuracy: 0.9875 - val_loss: 6.5366e-04 - val_accuracy:
1.0000
Epoch 10/10
20/20 [==============================] - 121s 6s/step - loss: 0.0058 - accuracy: 1.0000 - val_loss: 3.5061e-04 - val_accuracy:
1.0000
```





<Figure size 432x288 with 0 Axes>

In [14]:
```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test, batch_size=BS)
y_pred = np.argmax(y_pred, axis=1)
print(classification_report(Y_test.argmax(axis=1), y_pred,target_names=LB.classes_))
```

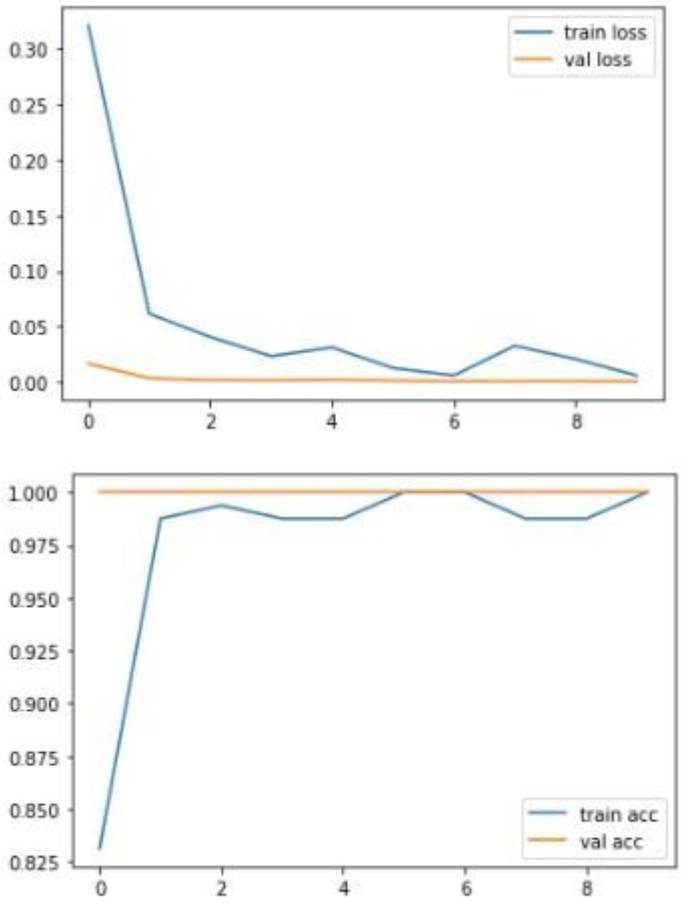
```
              precision    recall  f1-score   support

       Covid       1.00      1.00      1.00        20
      Normal       1.00      1.00      1.00        20

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

In [15]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test.argmax(axis=1), y_pred)
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])
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
[[20  0]
 [ 0 20]]
acc: 1.0000
sensitivity: 1.0000
specificity: 1.0000
```

XCEPTION

```
Compiling Starts
WARNING:tensorflow:From <ipython-input-10-91be3d96ebe8>:4: Model.fit_generator (from tensorflow.python.keras.engine.training) i
s deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
20/20 [==============================] - 113s 6s/step - loss: 0.3214 - accuracy: 0.8313 - val_loss: 0.0164 - val_accuracy: 1.00
00
Epoch 2/10
20/20 [==============================] - 117s 6s/step - loss: 0.0614 - accuracy: 0.9875 - val_loss: 0.0033 - val_accuracy: 1.00
00
Epoch 3/10
20/20 [==============================] - 116s 6s/step - loss: 0.0405 - accuracy: 0.9937 - val_loss: 0.0017 - val_accuracy: 1.00
00
Epoch 4/10
20/20 [==============================] - 111s 6s/step - loss: 0.0232 - accuracy: 0.9875 - val_loss: 0.0013 - val_accuracy: 1.00
00
Epoch 5/10
20/20 [==============================] - 130s 7s/step - loss: 0.0311 - accuracy: 0.9875 - val_loss: 0.0020 - val_accuracy: 1.00
00
Epoch 6/10
20/20 [==============================] - 126s 6s/step - loss: 0.0125 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.00
00
Epoch 7/10
20/20 [==============================] - 125s 6s/step - loss: 0.0057 - accuracy: 1.0000 - val_loss: 5.1894e-04 - val_accuracy:
1.0000
Epoch 8/10
20/20 [==============================] - 116s 6s/step - loss: 0.0325 - accuracy: 0.9875 - val_loss: 5.0877e-04 - val_accuracy:
1.0000
Epoch 9/10
20/20 [==============================] - 128s 6s/step - loss: 0.0206 - accuracy: 0.9875 - val_loss: 6.5366e-04 - val_accuracy:
1.0000
Epoch 10/10
20/20 [==============================] - 121s 6s/step - loss: 0.0058 - accuracy: 1.0000 - val_loss: 3.5061e-04 - val_accuracy:
1.0000
```



<Figure size 432x288 with 0 Axes>

In [12]:
```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test, batch_size=BS)
y_pred = np.argmax(y_pred, axis=1)
print(classification_report(Y_test.argmax(axis=1), y_pred,target_names=LB.classes_))
```

```
              precision    recall  f1-score   support

       Covid       1.00      1.00      1.00        20
      Normal       1.00      1.00      1.00        20

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40
```

In [13]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test.argmax(axis=1), y_pred)
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])
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```

```
[[20  0]
 [ 0 20]]
acc: 1.0000
sensitivity: 1.0000
specificity: 1.0000
```

CONCLUSION

Experiments and evaluation of the neural network models have been successfully done based on 80-20% of X-ray images for the model training and testing phases, respectively. The VGG16,and VGG19 models showed a good and similar performance of automated COVID-19 classification with f1-scores of 0.97 and 0.99 for normal and COVID-19, respectively.

This study demonstrated the useful application of deep learning models to classify COVID-19 in X-ray images based on the proposed COVIDX-Net framework.Clinical studies are the next milestone of this research work.

CNNs is a valuable pattern-recognition method both in theory and in application. In this paper, we proposed an innovative technique to enhance the deep learning ability of CNNs. The transfer learning methods has a better training performance, faster convergence rate, as well as a better recognition ability than the other model.

ACKNOWLEDGEMENT

The authors would like to thank Dr. U.V Arivazhagu for his suggestions and excellent guidance throughout the project period.

REFERENCES

[1] Yoo, S. H., Geng, H., Chiu, T. L., Yu, S. K., Cho, D. C., Heo, J., ... & Min, B. J. (2020). Deep Learning-Based Decision-Tree Classifier for COVID-19 Diagnosis FromChest X-ray Imaging. Frontiers in Medicine, 7, 427.

[2] Hemdan, E. E. D., Shouman, M. A., & Karar, M. E. (2020). Covidx-net: Aframework of deep learning classifiers to diagnose covid-19 in x-ray images. arXiv preprint arXiv:2003.11055.

[3] Elasnaoui, K., & Chawki, Y. (2020). Using X-ray images and deep learning for automated detection of coronavirus disease. Journal of Biomolecular Structure and Dynamics, (just-accepted), 1-22.

[4] 4. Asif, S., Wenhui, Y., Jin, H., Tao, Y., & Jinhai, S. (2020). Classification of COVID-19 from Chest X-ray images using Deep Convolutional Neural Networks. medRxiv.

[5] Majeed, T., Rashid, R., Ali, D., & Asaad, A. (2020). Covid-19 detection using CNNtransfer learning from X-ray Images. medRxiv.

[6] Ahammed, K., Satu, M. S., Abedin, M. Z., Rahaman, M. A., & Islam, S. M. S. (2020). Early Detection of Coronavirus Cases Using Chest X-ray Images Employing Machine Learning and Deep Learning Approaches. medRxiv.

[7] Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., & Acharya, U. R. (2020). Automated detection of COVID-19 cases using deep neural networks with X-ray images. Computers in Biology and Medicine, 103792.

[8] Wang, N., Liu, H., & Xu, C. (2020, July). Deep Learning for The Detection of COVID-19 Using Transfer Learning and Model Integration. In 2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC) (pp. 281-284). IEEE.

[9] Shibly, K. H., Dey, S. K., Islam, M. T. U., & Rahman, M. M. (2020). COVID FasterR-CNN: A Novel Framework to Diagnose Novel Coronavirus Disease (COVID-19) in X-Ray Images. medRxiv.