# COVID-19 detection over chest X-rays using Convolutional Neural Networks

A Project Report

Submitted in the fulfilment of the requirements for the

course CSCI B-565 Data Mining

By

Sai Varun Datta Vemavarapu
Akhil Varma Vegesna

Masters in Data Science

Luddy School of Informatics, Computing and Engineering

Under the supervision of

Dr. Yuzhen Ye

Associate Professor

Luddy School of Informatics, Computing and Engineering
Indiana University Bloomington

# ABSTRACT

COVID 19 was a rapidly prevailing viral disease with devastating effects on human life. People's lives, their well-being, and the national economy will be affected by this pandemic. It has caused health emergencies around the world. The virus has killed millions of people around the world. This puts a lot of pressure on the healthcare system and social life. Clinical studies of patients infected with COVID 19 have shown that most of these types of patients are infected with lung infections after being exposed to the disease. Therefore, COVID 19 needs to be detected early and quickly, which helps in the fighting virus. One of the main reasons for the rapid spread of the disease is the lack of test kits and the time it takes to provide test results. Therefore, chest computed tomography and imaging tests such as chest x-rays can detect the virus quickly and effectively because the lungs are affected when a person comes into contact with the virus. In our work, chest x-rays are cheap and available in any clinic, so we used chest x-rays instead of computed tomography scans of the chest to detect COVID 19. An infection was detected using a convolutional neural network algorithm. We built four different models: G19, ResNet50, InceptionV3 and Xception. As a result, we determined the confusion matrix and the accuracy of all models. The receiving model achieved the highest accuracy at 91%.

TABLE OF CONTENTS

# 1. INTRODUCTION

Coronavirus Disease 2019 (COVID-19) continues to have a major impact on the well-being and health of people around the world, caused by an individual virus caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARSCoV2). The illness causes breathing problems and can cause serious problems. Various symptoms of this virus include fever, dry cough, pain, sore throat, fatigue, diarrhea, headache and rash, discoloration of fingers and toes, conjunctivitis, dyspnea, loss of taste and smell, chest pain. And the loss of pressure, speech, or movement. As of February 2021, nearly 2.49 million people had died of COVID nd these are just confirmed cases. Many countries are now trying to invent vaccines that can prevent the effects of COVID 19 without side effects. The cost of diagnosing an illness is quite high. Currently, most tests are genetic tests known as reverse transcription-polymerase chain reactions. They are very accurate and can detect even the slightest trace of infection, but the results are time-consuming and costly. Therefore, not all hospitals can afford it. Current studies show that viruses belonging to this family are well recognized by X-rays. Therefore, the use of CT scans or chest x-rays is much faster, more accurate, and cheaper than PCR tests. In addition, chest x-rays are available in all hospitals, making them more economical than CT scans.

Artificial intelligence (AI) technology is becoming more and more important in this epidemic situation. AI is used to detect or predict various diseases and infectious diseases at an early stage. The use of AI in healthcare has the potential to assist healthcare providers in many aspects of the patient's care and management process. The use of ML algorithms to diagnose and detect illness is of great help to physicians as a support tool. Deep learning is used to identify a variety of problems, such as chest x-ray detection of respiratory illness. By providing successful results, DL occupies a unique position in the AI stream on image-based classification and regression issues. Image-based applications have become popular in recent years due to the use of convolutional neural networks (CNNs). ConvNet is a neural network that needs to minimize image pre-processing before it can be passed to the network and has a very high ability to extract features from the image.

Therefore, chest x-ray is the best way to detect COVID19. This was achieved with the help of convolutional neural networks. After cleaning up the image and applying data extensions, we used a deep learning-based CNN model to verify performance accuracy. Describes how to evaluate the results of training and testing using the model confusion matrix.

# 2. THEROTICAL ANALYSIS

In our work, we used chest x-rays which are cheap and available in any clinic, so we used chest x-rays instead of computed tomography scans of the chest to detect COVID 19. An infection was detected using a convolutional neural network algorithm. We built four different models: VGG19, ResNet50, InceptionV3 and Xception. The results, confusion matrix, and accuracy of all models have been determined.

### **2.1.1** Convolutional Neural Networks

Convolutional neural networks (ConvNets or CNN) are one of the main categories of image recognition and classification. A convolutional neural network (ConvNet / CNN) is a deep learning algorithm that can record input images, assign meaning to different aspects / objects (learnable weighting and distortion) in the image and distinguish them from each other. ConvNet requires much less pre-processing compared to other classification algorithms.

Deep Learning CNN model for training and testing. Each input image passes through a series of convolution layers with filters (kernel), pooling, and fully connected layer (FC), with probability values from 0 to the following figure processing the input image and objects based on the values.

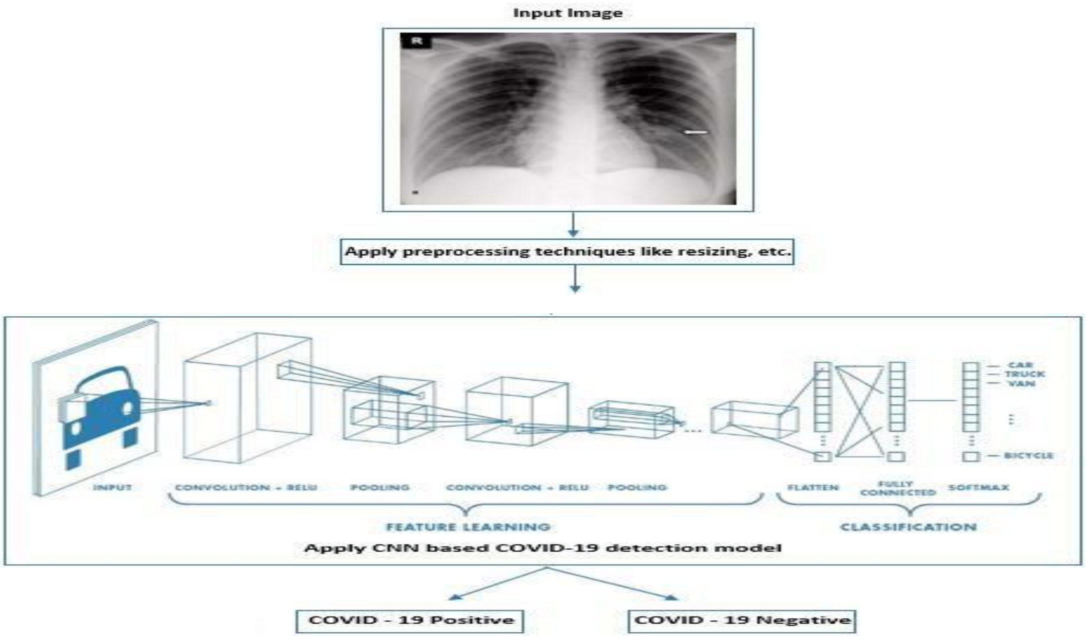



Fig.3.1. Representation of CNN algorithm's prediction

### 2.1.2 Convolution Layer

Convolution is the first layer to extract features from the input image. Convolution maintains relationships between pixels by learning image features using small squares of input data. This is a math operation that requires two inputs, such as an image matrix and a filter or kernel.

- An image matrix (volume) of dimension **(h x w x d)**
- A filter **($f_h$ x $f_w$ x d)**
- Outputs a volume dimension **(h − $f_h$ + 1) x (w − $f_w$ + 1) x 1**

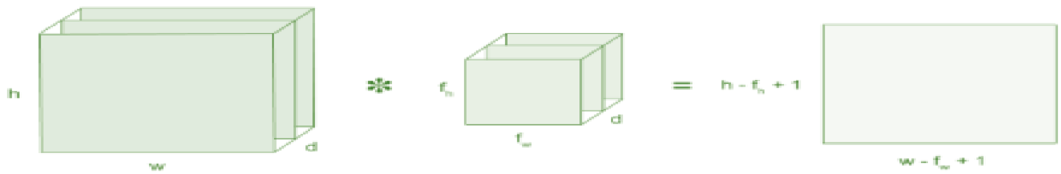
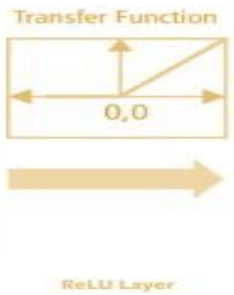



Fig.3.2. Image matrix multiplies kernel or filter matrix.

### 2.1.3 Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$. ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

| | | | |
|---|---|---|---|
| 15 | 20 | -10 | 35 |
| 18 | -110 | 25 | 100 |
| 20 | -15 | 25 | -10 |
| 101 | 75 | 18 | 23 |




| | | | |
|---|---|---|---|
| 15 | 20 | 0 | 35 |
| 18 | 0 | 25 | 100 |
| 20 | 0 | 25 | 0 |
| 101 | 75 | 18 | 23 |

Fig.3.3. ReLU operation

### 2.1.4 Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

### 2.1.5 Fully Connected Layer

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like a neural network.

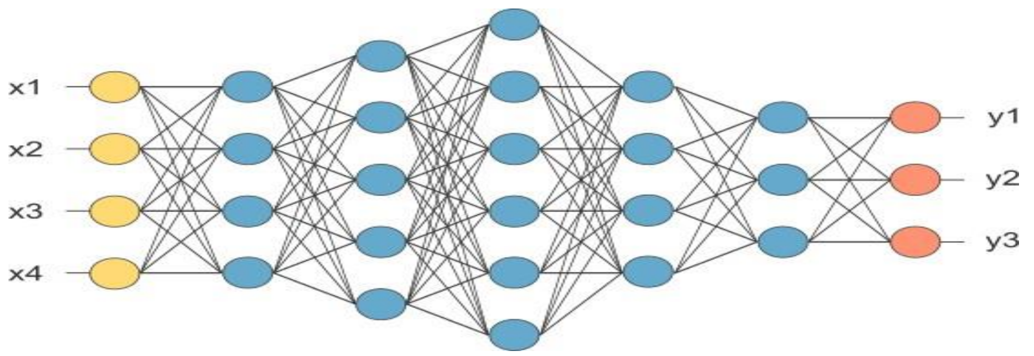



Fig.3.4. After pooling layer, flattened as FC layer.

## 2.2. Xception

Xception is a 71-layer depth foldable neural network. The Xception architecture is a linear stack of deeply separable folding layers with remaining connections. This is a deep and complex neural network architecture with deeply separable convolutions. It was developed by Google researchers. This is an extension of the Inception design. Its parameter size is similar to the parameter size of the Inception network but slightly better than the Inception network.

His basic hypothesis: The mapping between cross-channel and spatial correlations can be completely separated. It is divided into 14 modules, with linear residual connections around all modules except the first and last modules.

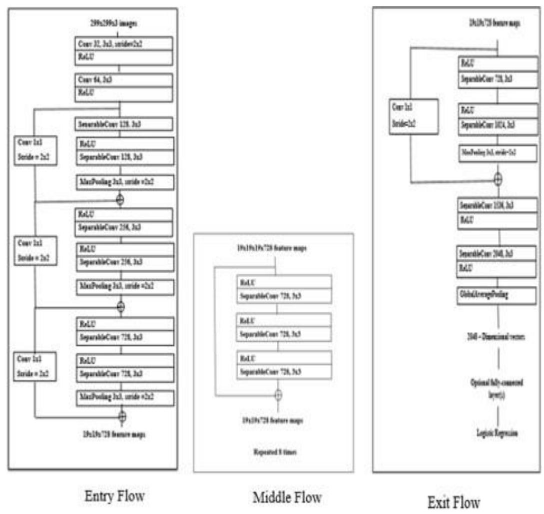



Fig.3.5. Xception Model Architecture.

## 2.3. Inception V3

Inception V3 is the third version of Google's DL convolution architecture. Inception V3 is a CNN that supports image analysis and object recognition. It focuses primarily on consuming less computing power by improving and modifying the previous Inception architecture. Originally trained on the original ImageNet 1000 class dataset trained with over 1 million training images. Inception V3 has RMSProp Optimizer, factorized 7x7 convolution, auxiliary classifier BatchNorm, and label smoothing (this is a kind of regularization component added to the loss equation, which is a network clutter. Prevents you from becoming too safe with respect to the network, i.e., prevents over-fitting).

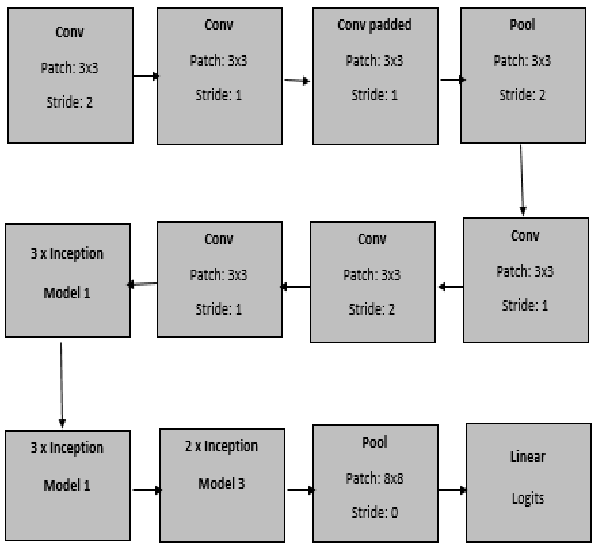


Fig.3.6. InceptionV3 Model Architecture.

## 2.4. ResNet50

Compared to other CNNs, ResNets are easy to understand. In Residual neural network, there are fewer filters and lower complexity during training. ResNet 50 is a CNN which is 50 layers deep. The ResNet-50 model has 5 stages with an identity and convolution block. Each Conv block has over three convolution layers and each identity block has three convolution layers. ResNet-50 has over 23 million trainable parameters.

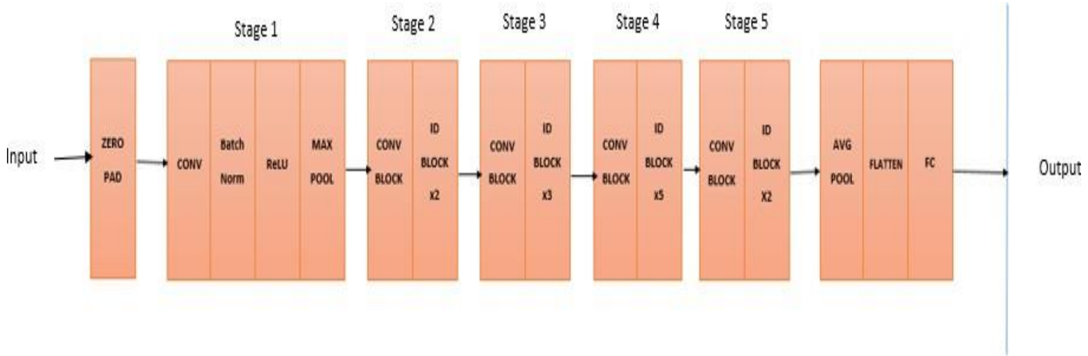



Fig.3.7. ResNet50 Model Architecture.

## 2.5. VGG19

VGG19 model is a kind of VGG model which comprises of 19 layers i.e., 16 conv layers, 5 MaxPool layers, 3 fully connected layers and one SoftMax layer. VGG is a successor of AlexNet. It was created by a different group named Visual Geometry

Group from Oxford. It uses the ideas of its predecessors and uses deep conv neural layers to increase accuracy.


Fig.3.8. VGG19 Model Architecture.

# 3. EXPERIMENTAL INVESTIGATIONS

After cleaning up the images and applying data augmentation, we used a deep learning-based CNN models and checked the accuracy of the performance. The evaluation of results in terms of training and testing with confusion matrices for the models are discussed.

### 3.1.1 Modules

There are four modules in our project “COVID-19 detection over chest X-rays using CNN”. Each module includes a type of CNN model.

1. Xception Model

2. InceptionV3 Model

3. ResNet50 Model

4. VGG19 Model

## 3.2. Dataset

To test the results of the model, we used chest radiographs of healthy patients suffering from COVID. I collected a dataset from a GitHub open repository. Our dataset contains a total of 940 images of chest x-rays. Of these images, 505 chest x-rays were taken from healthy patients and 435 chest x-rays were taken from patients with COVID 19. The dataset is divided into two parts. 80% for model training and 20% for model testing. The following image shows an example of a chest radiograph of a healthy patient affected by COVID19.

(a)

(b)

Fig.4.1. Sample x-ray of (a) COVID infected patient (b) non-COVID person.

## 3.3. Packages

In this project we have used several packages namely TensorFlow, NumPy, Sklearn, Seaborn, cv2 and Matplotlib.

- TensorFlow

  TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Itis a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. It was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

- NumPy

  NumPy is a python library used for operating with arrays. It additionally has functions for operating in domain of algebra, Fourier rework, and matrices. NumPy was created in 2005 by Travis Oliphant. Its associate degree open supply project and you'll be able to use it freely. NumPy stands for Numerical Python. NumPy is a Python library and is written partially in Python.

- Scikit-learn

  Scikit-learn is a machine learning package in python. Within the Scikit package, all the functions are put down in optimized code, it's a really manageable and economical tool for knowledge analysis and data processing.

- Seaborn

  Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- OpenCV

  OpenCV provides a real-time optimized Computer Vision library, tools, and hardware. It also supports model execution for Machine Learning (ML) and Artificial Intelligence (AI).

- Matplotlib

  Matplotlib is an incredible image library in Python for second plots of arrays. Matplotlib may be a multi-platform knowledge image library engineered on NumPy arrays and designed to figure with the broader SciPy stack. It absolutely was introduced by John Hunter within the year 2002. One of the best advantages of image is that it permits U.S.A. visual access to large amounts of knowledge in simply assailable visuals. Matplotlib consists of many plots like line, bar, scatter, bar chart etc**.**

# 5. EXPERIMENTAL RESULTS

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
block1_conv1 (Conv2D)           (None, 111, 111, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
block1_conv1_bn (BatchNormaliza (None, 111, 111, 32) 128         block1_conv1[0][0]
__________________________________________________________________________________________________
block1_conv1_act (Activation)   (None, 111, 111, 32) 0           block1_conv1_bn[0][0]
__________________________________________________________________________________________________
block1_conv2 (Conv2D)           (None, 109, 109, 64) 18432       block1_conv1_act[0][0]
__________________________________________________________________________________________________
block1_conv2_bn (BatchNormaliza (None, 109, 109, 64) 256         block1_conv2[0][0]
__________________________________________________________________________________________________
block1_conv2_act (Activation)   (None, 109, 109, 64) 0           block1_conv2_bn[0][0]
__________________________________________________________________________________________________
block2_sepconv1 (SeparableConv2 (None, 109, 109, 128 8768        block1_conv2_act[0][0]
__________________________________________________________________________________________________
block2_sepconv1_bn (BatchNormal (None, 109, 109, 128 512         block2_sepconv1[0][0]
__________________________________________________________________________________________________
block2_sepconv2_act (Activation (None, 109, 109, 128 0           block2_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block2_sepconv2 (SeparableConv2 (None, 109, 109, 128 17536       block2_sepconv2_act[0][0]
__________________________________________________________________________________________________
block2_sepconv2_bn (BatchNormal (None, 109, 109, 128 512         block2_sepconv2[0][0]
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 55, 55, 128)  8192        block1_conv2_act[0][0]
__________________________________________________________________________________________________
block2_pool (MaxPooling2D)      (None, 55, 55, 128)  0           block2_sepconv2_bn[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 55, 55, 128)  512         conv2d[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 55, 55, 128)  0           block2_pool[0][0]
                                                                 batch_normalization[0][0]
__________________________________________________________________________________________________
block3_sepconv1_act (Activation (None, 55, 55, 128)  0           add[0][0]
__________________________________________________________________________________________________
block3_sepconv1 (SeparableConv2 (None, 55, 55, 256)  33920       block3_sepconv1_act[0][0]
__________________________________________________________________________________________________
block3_sepconv1_bn (BatchNormal (None, 55, 55, 256)  1024        block3_sepconv1[0][0]
__________________________________________________________________________________________________
block3_sepconv2_act (Activation (None, 55, 55, 256)  0           block3_sepconv1_bn[0][0]
__________________________________________________________________________________________________
block3_sepconv2 (SeparableConv2 (None, 55, 55, 256)  67840       block3_sepconv2_act[0][0]
__________________________________________________________________________________________________
block3_sepconv2_bn (BatchNormal (None, 55, 55, 256)  1024        block3_sepconv2[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 28, 28, 256)  32768       add[0][0]
__________________________________________________________________________________________________
block3_pool (MaxPooling2D)      (None, 28, 28, 256)  0           block3_sepconv2_bn[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 28, 28, 256)  1024        conv2d_1[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 28, 28, 256)  0           block3_pool[0][0]
                                                                 batch_normalization_1[0][0]
__________________________________________________________________________________________________
block4_sepconv1_act (Activation (None, 28, 28, 256)  0           add_1[0][0]
__________________________________________________________________________________________________
block4_sepconv1 (SeparableConv2 (None, 28, 28, 728)  188672      block4_sepconv1_act[0][0]
__________________________________________________________________________________________________
block4_sepconv1_bn (BatchNormal (None, 28, 28, 728)  2912        block4_sepconv1[0][0]
```

Fig.6.1. Xception Model Summary

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 111, 111, 32) 864         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 111, 111, 32) 96          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 111, 111, 32) 0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 109, 109, 32) 9216        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 109, 109, 32) 96          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 109, 109, 32) 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 109, 109, 64) 18432       activation_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 109, 109, 64) 192         conv2d_2[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 109, 109, 64) 0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 54, 54, 64)   0           activation_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 54, 54, 80)   5120        max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 54, 54, 80)   240         conv2d_3[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 54, 54, 80)   0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 52, 52, 192)  138240      activation_3[0][0]
__________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 52, 52, 192)  576         conv2d_4[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 52, 52, 192)  0           batch_normalization_4[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 25, 25, 192)  0           activation_4[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 25, 25, 64)   12288       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 25, 25, 64)   192         conv2d_8[0][0]
__________________________________________________________________________________________________
activation_8 (Activation)       (None, 25, 25, 64)   0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 25, 25, 48)   9216        max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 25, 25, 96)   55296       activation_8[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 25, 25, 48)   144         conv2d_6[0][0]
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 25, 25, 96)   288         conv2d_9[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, 25, 25, 48)   0           batch_normalization_6[0][0]
__________________________________________________________________________________________________
activation_9 (Activation)       (None, 25, 25, 96)   0           batch_normalization_9[0][0]
__________________________________________________________________________________________________
average_pooling2d (AveragePooli (None, 25, 25, 192)  0           max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 25, 25, 64)   12288       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 25, 25, 64)   76800       activation_6[0][0]
```

Fig.6.2. InceptionV3 Model Summary

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_add (Add)          (None, 56, 56, 256)  0           conv2_block1_0_bn[0][0]
                                                                 conv2_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_out (Activation)   (None, 56, 56, 256)  0           conv2_block1_add[0][0]
__________________________________________________________________________________________________
conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block1_out[0][0]
__________________________________________________________________________________________________
conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0           conv2_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0           conv2_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_add (Add)          (None, 56, 56, 256)  0           conv2_block1_out[0][0]
                                                                 conv2_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_out (Activation)   (None, 56, 56, 256)  0           conv2_block2_add[0][0]
```

**Fig.6.3. ResNet50 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_conv4 (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_conv4 (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_conv4 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dropout (Dropout)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 50178
=================================================================
Total params: 20,074,562
Trainable params: 50,178
Non-trainable params: 20,024,384
_________________________________________________________________
```

**Fig.6.4. VGG19 Model Summary**

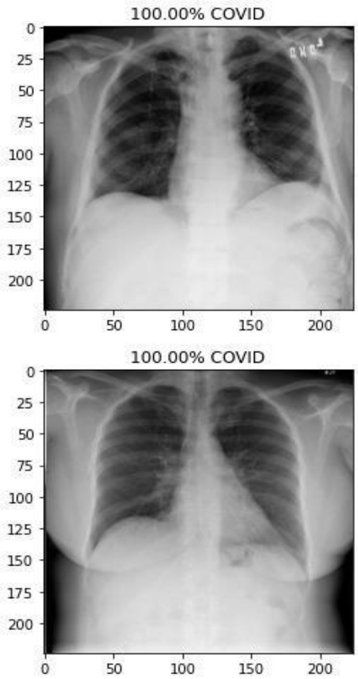



Fig. 6.5. Xception Model:
**Visualization of predictions of chest X-rays**

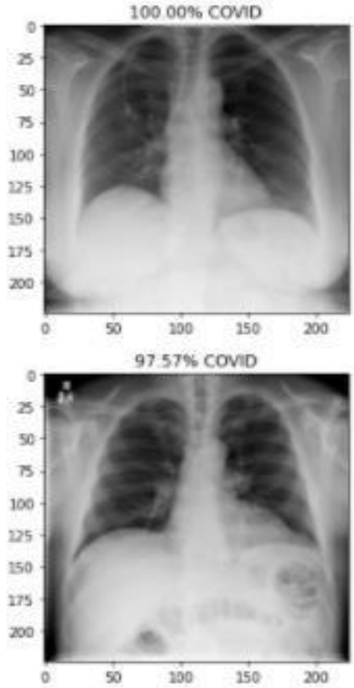



Fig. 6.6. InceptionV3 Model:
**Visualization of predictions of chest X-rays**

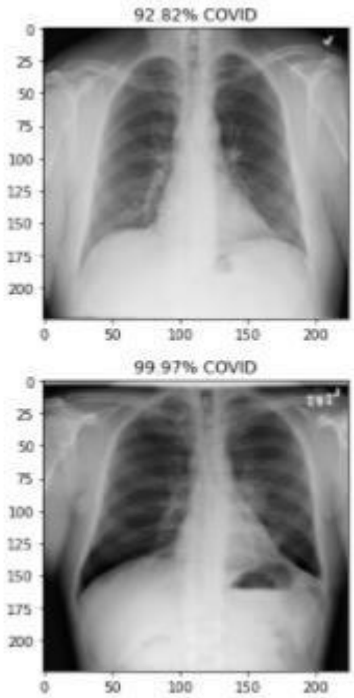



**Fig. 6.7. ResNet50 Model: Visualization of predictions of chest X-rays**

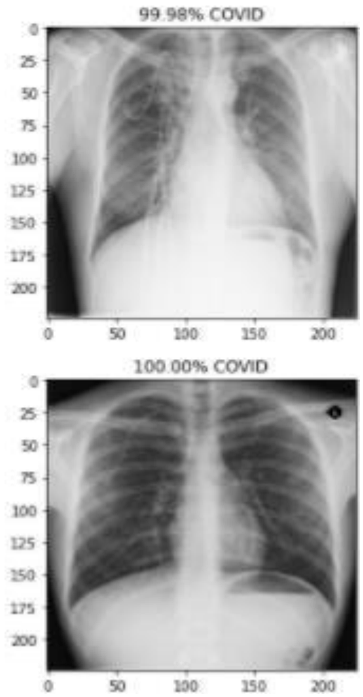



**Fig. 6.8. VGG19 Model: Visualization of predictions of chest X-rays**

**Table 6.1. Classification Report of Xception model on chest X-rays.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.88 | 0.97 | 0.93 | 70 |
| 1 | 0.97 | 0.89 | 0.93 | 80 |
| accuracy | | | 0.93 | 150 |
| macro avg | 0.93 | 0.93 | 0.93 | 150 |
| weighted avg | 0.93 | 0.93 | 0.93 | 150 |

**Table 6.2. Classification Report of InceptionV3 model on chest X-rays.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.93 | 0.89 | 87 |
| 1 | 0.94 | 0.86 | 0.90 | 101 |
| accuracy | | | 0.89 | 188 |
| macro avg | 0.89 | 0.90 | 0.89 | 188 |
| weighted avg | 0.90 | 0.89 | 0.89 | 188 |

**Table 6.3. Classification Report of ResNet50 model on chest X-rays.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.89 | 0.90 | 87 |
| 1 | 0.90 | 0.93 | 0.92 | 101 |
| accuracy | | | 0.91 | 188 |
| macro avg | 0.91 | 0.91 | 0.91 | 188 |
| weighted avg | 0.91 | 0.91 | 0.91 | 188 |

**Table 6.4. Classification Report of VGG19 model on chest X-rays.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.96 | 0.85 | 0.90 | 87 |
| 1 | 0.88 | 0.97 | 0.92 | 101 |
| accuracy | | | 0.91 | 188 |
| macro avg | 0.92 | 0.91 | 0.91 | 188 |
| weighted avg | 0.92 | 0.91 | 0.91 | 188 |

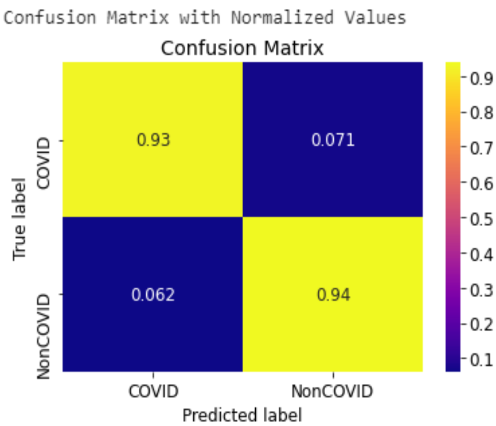



**Fig. 6.7. Xception Model: Confusion Matrix with Normalized Values**

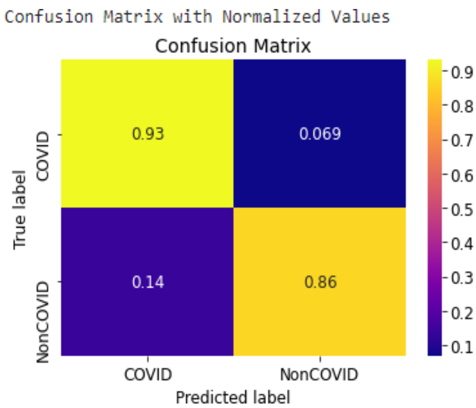
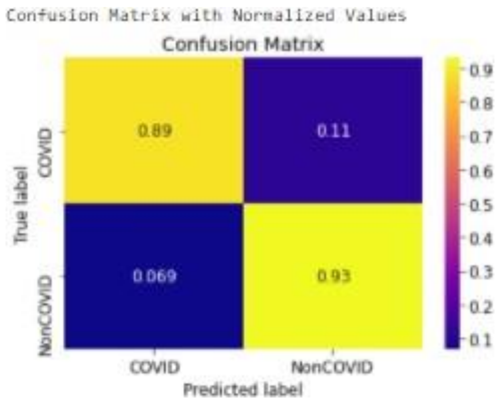



**Fig. 6.8. InceptionV3 Model: Confusion Matrix with Normalized Values**



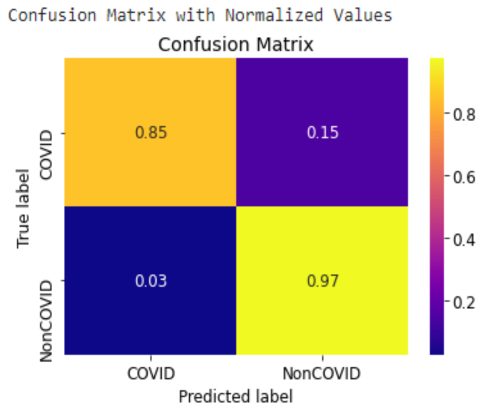
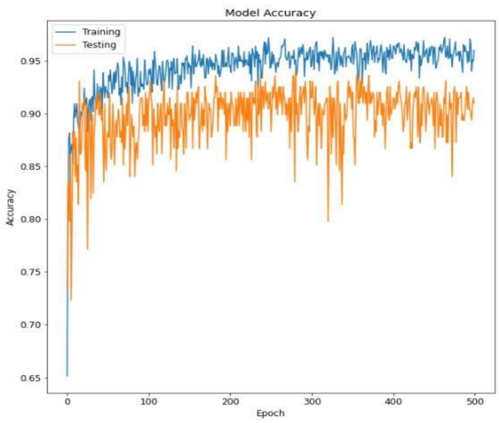




**Fig. 6.9. ResNet50 Model: Confusion Matrix with Normalized Values**

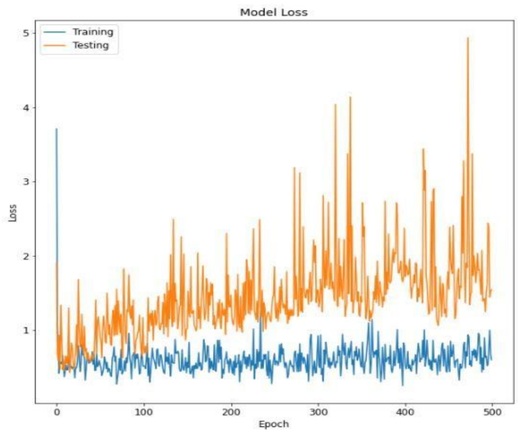
**Fig. 6.10. VGG19 Model: Confusion Matrix with Normalized Values**

**Fig. 6.9. ResNet50 Model: Confusion Matrix with Normalized Values**

**Fig. 6.10. VGG19 Model: Confusion Matrix with Normalized Values**

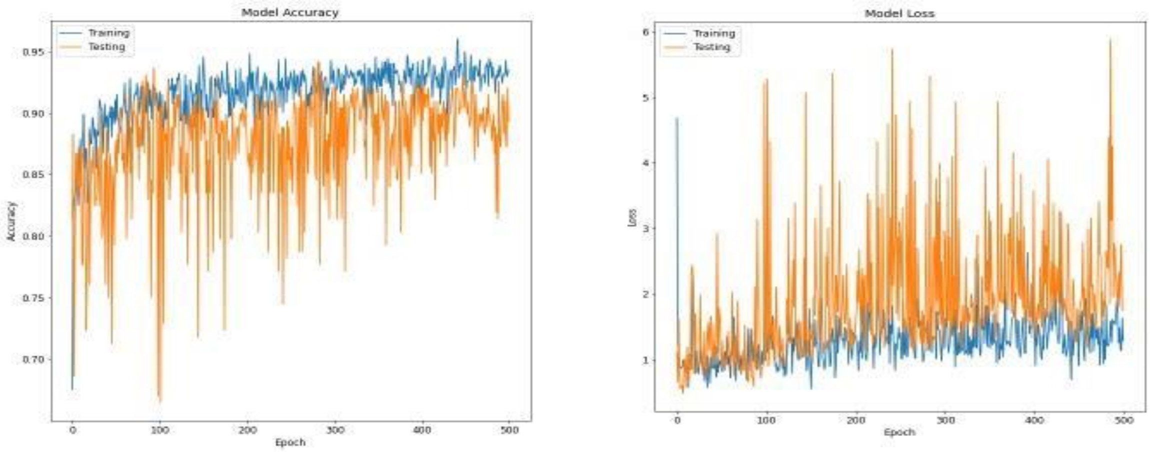



**(a)**



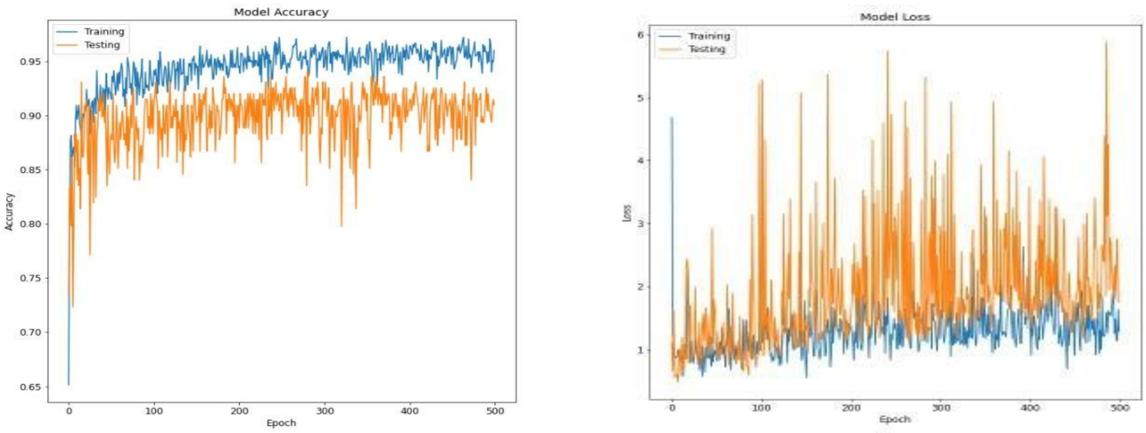

**(b)**

**Fig.6.11. Xception Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.**



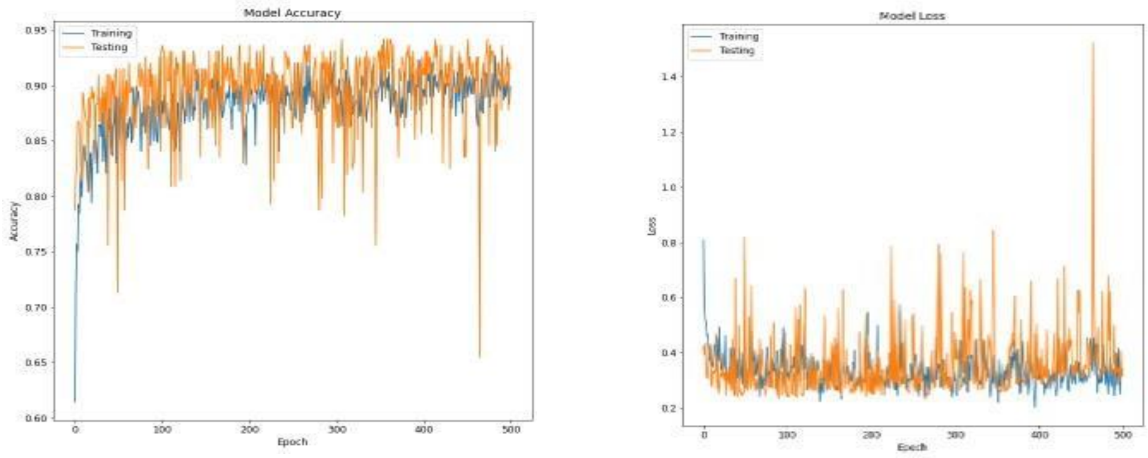

**(a)** **(b)**

**Fig.6.12. IncetionV3 Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.**




**(a)** **(b)**

**Fig.6.13. ResNet50 Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.**




**(a)** **(b)**

**Fig.6.14. VGG19 Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.**

# 6. DISCUSSION OF RESULTS

First, we took a dataset from GitHub, 80% of which was used for training and 20% for testing. In addition, because the images are different sizes, we resized and preprocessed the images to 224 x 224 px (fixed size) before feeding them to the deep learning model. Next, we normalized the image. When creating the model, we added three custom layers so that they can be used repeatedly in the dataset. Next, we added a flattening layer to smooth out all the features and a dropout layer to avoid overfitting. Finally, I used the Adam Optimizer to compile the model and used the category cross-entropy as the loss function. The

image data generator was used to train the model with modified versions of the image, including different reflections, angles, displacements, and rotations. Trained the model with a lot size of 500 epochs and 32 images. The predictions were generated by running a trained model on the images in the test set. Predictions show how much percent of COVID 19 is detected on chest x-rays. Then I plotted the results and graphs to get an idea of the accuracy of the four models.

Next, I applied some results and plots to understand the accuracy of the four models. The accuracy of the Xception model is 93%, which is the highest. ResNet50 and VGG19 are 91%. Inception V3 of 89 Curates.

# 7. CONCLUSION

COVID 19 is growing very rapidly and rapidly. With the increasing number of cases and deaths daily, there is an urgent need for rapid detection of COVID cases. In this task, four CNN models were analyzed to classify people affected by COVID based on chest x-rays. The Xception model achieved the highest accuracy at 93%. The VGG19 and ResNet 50 models achieved 91% accuracy. The Inception V3 model achieved 89% accuracy. All models were compared to plots and confusion matrices. Successful classification of COVID19 scans shows the potential for future automation of diagnostic tasks.

Our future goal is to use it in larger datasets to see if we can achieve this high level of accuracy and make better predictions, as in machine learning. Training your model with large amounts of data makes it more accurate to predict invisible data. This is not clinically approved, so you can consult an expert to get a clear idea of how practical the model is. It turns out that this study will be done in the near future, as this is only an economically legitimate solution.

# 8. REFERENCES

[1] Nishant, P. S., Mehrotra, S., Mohan, B. G. K., & Devaraju, G. (2020). Identifying Classification Technique for Medical Diagnosis. In *ICT Analysis and Applications* (pp. 95-104). Springer, Singapore.

[2] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, *316*(22), 2402-2410.

[3] Elsheikh, A. H., Saba, A. I., Abd Elaziz, M., Lu, S., Shanmugan, S., Muthuramalingam, T., ... & Shehabeldeen, T. A. (2021). Deep learning-based forecasting model for COVID-19 outbreak in Saudi Arabia. *Process Safety and Environmental Protection*, *149*, 223-233.

[4] Ozsahin, I., Sekeroglu, B., & Mok, G. S. (2019). The use of back propagation neural networks and 18F-Florbetapir PET for early detection of Alzheimer's disease using Alzheimer's Disease Neuroimaging Initiative database. *Plos one*, *14*(12), e0226577.

[5] Roy, S. K., Krishna, G., Dubey, S. R., & Chaudhuri, B. B. (2019). HybridSN: Exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, *17*(2), 277-281.

[6] Yoon, H., Lee, J., Oh, J. E., Kim, H. R., Lee, S., Chang, H. J., & Sohn, D. K. (2019). Tumor identification in colorectal histology images using a convolutional neural network. *Journal of digital imaging*, *32*(1), 131-140.

[7] Kalyan, C. G., Neelakanteswara, P., Nagaraj, J., Harika, S., & Basha, C. Z. (2020, November). Automatic Classification of human gender using X-ray images with Fuzzy C means and Convolution Neural Network. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 919-923). IEEE.

[8] Wu, H., Xie, P., Zhang, H., Li, D., & Cheng, M. (2020). Predict pneumonia with chest X-ray images based on convolutional deep neural learning networks. *Journal of Intelligent & Fuzzy Systems*, (Preprint), 1-15.

[9] Wu, H., Xie, P., Zhang, H., Li, D., & Cheng, M. (2020). Predict pneumonia with chest X-ray images based on convolutional deep neural learning networks. *Journal of Intelligent & Fuzzy Systems*, (Preprint), 1-15.

[10] . Xu, B., & Meng, X. A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19). *Preprint*