ANALYSIS & DETECTION OF COVID-19 OVER CHEST X-RAYS USING DIFFERENT CONVOLUTIONAL NEURAL NETWORK MODELS

M. Amrutha Tejaswini[1] and Shashi Mehrotra[2]

[1, 2] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.

Abstract

Novel coronavirus pandemic hit the globe in December 2019. It caused a health emergency globally. The virus has caused millions of deaths all over the globe. This has caused tremendous pressure on the healthcare system and social lives. Thus, the need to detect COVID at an early stage and also at a faster rate will help us fight against the virus. One of the primary reasons for the rapid spreading of the disease is not having the testing kits and also the time it takes to give the result of the test. Hence, using the imaging techniques like chest computed tomography scans or chest X-rays will detect the virus quickly and effectively as lungs are affected when a person comes in contact with the virus. In our work, we used chest X-rays to detect COVID-19 rather than chest computed tomography scans as chest X-rays are more affordable and available in every clinic. We have used a convolutional neural network algorithm to detect the virus. We built four different models i.e., Xception, VGG19, Inception V3 and ResNet50. We determined the results, confusion matrices, and accuracy of all models. Xception model gave out the best accuracy with a 93%.

Keywords: COVID-19, CNN, Deep Learning, Xception, VGG19, Inception V3, ResNet50.

1. Introduction

Coronavirus Disease 2019 (COVID-19) continues to have a devastating effect on the well-being and health of the world population, caused by the virus of individuals by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The illness was first recognized in December 2019 in Wuhan, China, and has since spread all around the world. This disease can cause respiratory problems and cause severe problems. Various symptoms of this virus include fever, dry cough, aches, sore throat, tiredness, diarrhea, headache, and rashes on the skin, discoloration of fingers or toes, conjunctivitis, difficulty in breathing, loss of taste or smell, chest pain, or pressure and loss of speech or movement. As of February 2021, almost 2.49 million lives are lost to COVID-19, and these are only the confirmed cases. At present, many nations are trying to invent a vaccine that can prevent getting affected by COVID-19 without any side effects. The costs for the diagnosis of the disease are pretty high. Presently, most of the tests are genetic tests known as Reverse Transcription Polymerase Chain Reaction. They are very accurate and can detect even the slightest trace of the infection, but at the same time, the results take a long time and are also costly. So, not all hospitals can afford to perform it. Now, studies show that viruses belonging to this family are detected well in radiographic images. So, using CT scans or chest X-rays is much faster, accurate, and also cheaper than PCR tests. Furtherly, chest X-rays are more economical than CT scans as they are available in every hospital.

In this epidemic situation, Artificial Intelligence (AI) techniques are becoming vital. AI is used to detect or predict various diseases and infections in an early stage. The use of AI in healthcare has the potential to help healthcare providers in many aspects of patient care and administrative processes. Using ML algorithms for disease diagnosis and detection is being of huge aid to doctors as a supportive tool. Deep learning is used in detecting several problems like respiratory disorder detection from chest X-rays. DL has gained a unique place in the stream of AI for image-based classification and regression problems by providing successful results. Using Convolutional Neural Networks (CNN) has enabled image-based applications to reach their popularity in the past few years. ConvNet is a neural network that requires very minimal pre-processing of images before giving it to the network, and it is very capable of extracting features from images.

Therefore, chest X-ray images are the best way to detect COVID-19 and we achieved this using CNN. After cleaning up the images and applying data augmentation, we used 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 is discussed.

2. Literature Survey

Using knowledge extraction to detect various infections and diseases by getting a hold of medical data will help in the early detection of various health-related problems. There are numerous machine learning (ML) and deep learning (DL) algorithms that help in disease detection. The authors in [1], provided a comparative view on various ML techniques in medical diagnosis. Quite a several researchers worked on medical imaging on Convolutional Neural Networks. In [2], the author developed a DL algorithm to automatically detect diabetic retinopathy and diabetic macular edema from the images of the retinal fundus. This algorithm was built using large datasets and outperform medical professionals. In [3], the authors forecasted the number of total recovered cases, totally confirmed cases, and total deaths in Saudi Arabia using long short-term memory (LSTM) network. The author in [4], utilized various pre-processing techniques and average pixel per node (APPN) method to detect Alzheimer's from the positron emission tomography (PET) images. Deep CNNs were used in various image processing applications and gained high accuracy which opened its prospects for the future. In [5], the author examined Convolutional Neural Networks for hyperspectral image classification. The authors in [6] performed tumor detection using CNN using the colorectal histology images. In [7], the authors detected prostate cancer using deep learning from CT images. In [8], the authors using X-rays to classify gender using CNN. This was proposed as an application to identify gender after a natural calamity or bombings.

The idea to use radiographic images to detect COVID-19 came from the researches where CT scans and chest X-rays were used in pneumonia detection. In [9], the authors used chest X-rays and CNN to detect pneumonia. Here, if the algorithm is trained on the chest X-rays, it predicts and gives an outcome regarding pneumonia.

In [10], authors attempted to detect COVID-19 on CT scans using a deep learning algorithm. It achieved an accuracy of 82.9%. The authors in [11], used CNN and mobile net on chest X-rays to detect COVID-19. In [12], the authors provided a way to detect COVID-19 at an early stage by using ML algorithms through CT scans. In this research, feature extraction processes were used to improve classification accuracy. Wavelet transformation algorithm was also used to extract features and classification was done using Support Vector Machines. This got an accuracy of 99.68%. The authors in [13], proposed a neural network to measure and predict the severity of pneumonia in COVID-19 chest X-rays and in general which will help the medical professionals know the seriousness and monitor accordingly. In [14], the authors used Inception-ResNetV2 to classify CT scans and chest X-rays into coronavirus, pneumonia, and normal classes. In [15], authors used XCeption, Inception V3, and ResNeXt on chest X-rays to detect COVID-19. In [16], the authors used a darknet model and built 17 convolutional layers with different kinds of filtering in each layer. This research also talked about findings that are usually noticed in chest X-rays of COVID-affected patients. The authors also developed the binary classification of COVID vs no findings and multiclass classification of COVID vs no findings vs pneumonia using the deep learning technique. The binary classification attained an accuracy of 98.08% and multiclass classification attained an accuracy of 87.02%.

Transfer learning is an approach where the model developed for one task is used as a starting point for another task. In [17], the authors built a transfer learning approach on chest X-rays to predict pediatric pneumonia. This model achieved a maximum accuracy of 96.4% and recall of 99.62%. Various researches have been performed on this method. In the light of COVID, researchers also developed a transfer learning approach to detect COVID. The authors in [18], the authors worked on an X-ray dataset of people with confirmed COVID-19, microorganism respiratory disorder, and normal diseases. Authors used transfer learning with CNN, attaining accuracy of 96%.

Later on, new algorithms were developed to detect COVID-19. In [19], the authors developed an algorithm based on a deep learning approach i.e., Mask-RCNN. This algorithm was used to identify and localize pneumonia in chest X-rays. In this study, three types of X-rays were used i.e., abnormal, lung opacity, and normal. The authors in [20], developed a CheXNet model which detects COVID-19 using chest X-rays. In [21], the authors introduced COVIDiagnosis-Net which is based on deep SqueezeNet using Bayes Optimization. This method was used in the detection of COVID-19 with an accuracy of 98.26%. The authors in [22], the authors performed COVID-19 detection. They used an age-based selection of chest X-rays for training and testing of Artificial Intelligence (AI) based models. The images were classified as pediatric images

(COVID-19 vs normal vs pneumonia) from adult images.

After going through all the views and proposed work by different researchers, we have decided to build four Convolutional Neural Network models. VGG19, ResNet50, InceptionV3, and Xception were trained separately on Chest X-rays. We have used 80% of the images were used for training the models and the rest 20% of the data for testing the accuracy of the models. Lastly, we compare the results and accuracy of the four models and determine the best model.

3. Materials and Methods

3.1 Dataset Description

To test the results of our models, we have used chest X-rays of patients who have COVID and who are healthy. We have gathered the dataset from GitHub open repository. Our dataset has a total of 940 images of chest X-rays. Out of these images, 505 chest X-rays are of healthy patients and 435 chest X-rays are of COVID-19 affected patients. The dataset will be divided into two sets. 80% for training the model and 20% for testing the model. The following images show sample chest X-rays of COVID-19 affected patients and healthy people.

(a)

(b)

Fig. 1 Sample x-ray of (a) COVID infected patient (b) non-COVID person.

All the images are in different dimensions, so we resized them to 224 x 224 and are normalized.

3.2 Xception Model

Xception is a convolutional neural network that has 71 layers of depth. The architecture of the Xception model is a linear stack with depth-wise separable convolution layers and residual connections. This model's architecture includes a deep CNN architecture that involves depth wise separable convolutions. The Xception model was developed by Google researchers. This model is an expansion of inception design. The parameter size of Xception is similar to that of Inception net, but it performs quite better when compared to the Inception net.

It is a fundamental hypothesis of the mapping of spatial correlations and cross-channel correlations can be entirely decoupled. The model is structured into 14 modules and all of those have linear residual connections surrounding them, except for the last and first modules.

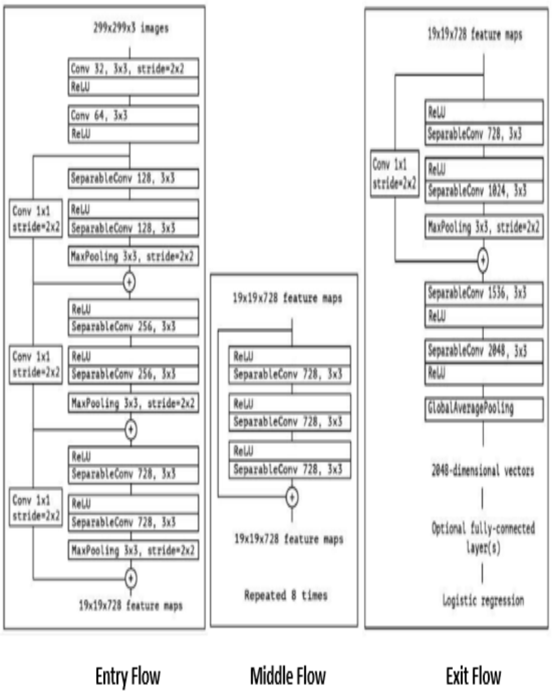


Fig. 2 Xception Model Architecture

SeparableConv is the improved depth wise separable convolution. SeparableConvs are treated as Inception Modules and placed throughout the whole deep learning architecture.

3.3 Inception V3 Model

Inception V3 is the 3rd version of DL convolutional architectures by Google. Inception V3 is a CNN that helps in image analysis and object detection. It mainly focuses on burning less computational power by improving and modifying previous Inception architectures. It was initially trained on an original ImageNet dataset of 1000 classes which was trained on over a million training images. Inception V3 has got RMSProp Optimizer, factorized 7x7 convolutions, BatchNorm in the auxiliary classifiers and Label Smoothing (it is a type of regularizing component that is added to loss formula which prevents the network from turning too confident about a class i.e., prevents overfitting).

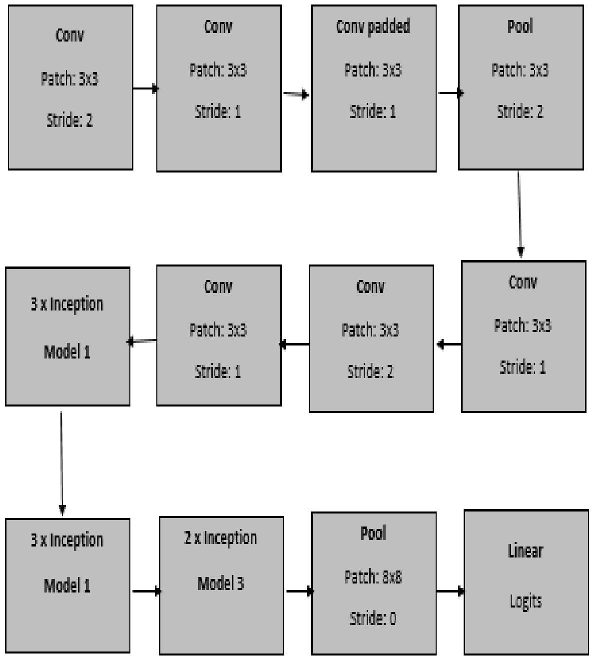


Fig. 3 Inception V3 Architecture

3.4 ResNet50 Model

Compared to other CNNs, ResNets are easy to understand. In Residual neural network, there are lesser filters and lower complexity while 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 about three convolution layers and each one of the identity block has three convolution layers. ResNet-50 has over 23 million trainable parameters.

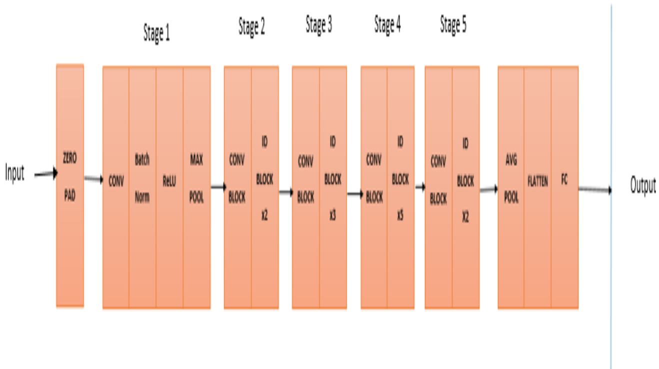


Fig. 4 ResNet50 Model Architecture

3.5 VGG19 Model

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.

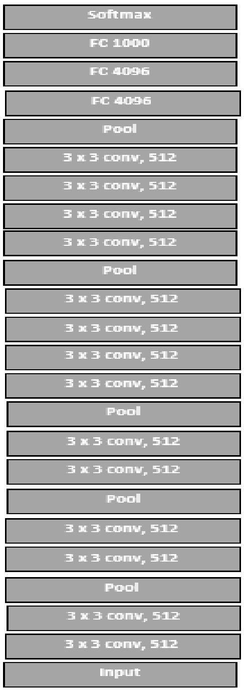


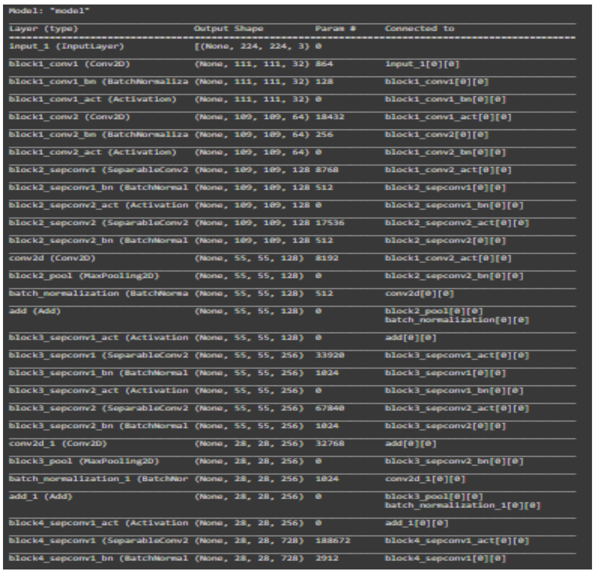
Fig. 5 VGG19 Model Architecture

3.3 Methodology

Firstly, we have taken the dataset from GitHub and used 80% of it for training and 20% for testing. Furtherly, we performed preprocessing on the images by resizing them to 224 x 224 px (fixed size) before they were fed into the deep learning model as images are of different sizes. Then we performed normalization on images. While building the models, we added 3 custom layers, so that it can be travelled on our dataset. Then we added a Flatten layer just to flatten all features and also a Dropout layer to overcome overfitting. Lastly, we compiled the model with the adam optimizer and also used categorical cross entropy as the loss function.

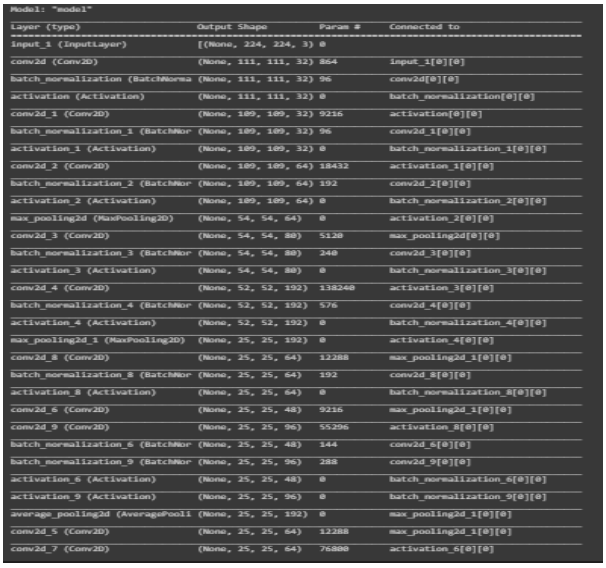
3.4 Experimental Analysis & Results

We used an Image Data Generator to train the models at modified versions of the images, such as at different flips, angles, shifts, or rotations. We have trained the model with 500 epochs and a batch size of 32 images.


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 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. 7 Inception V3 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. 8 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. 9 VGG19 Model Summary

Predictions were generated by running the trained models on the test data. The predictions show how much percentage of COVID-19 is detected in the chest X-rays.

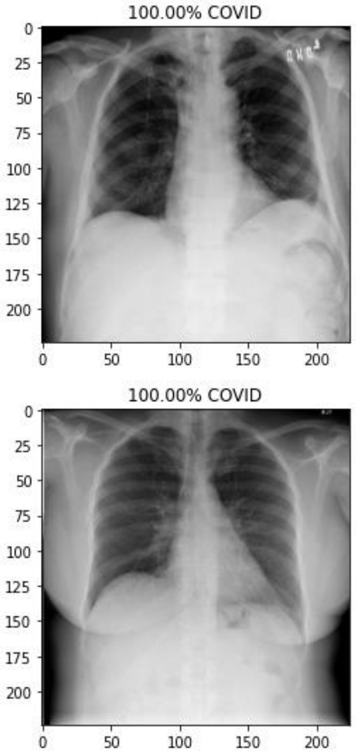


Fig. 10 Xception Model: Visualization of predictions of chest X-rays

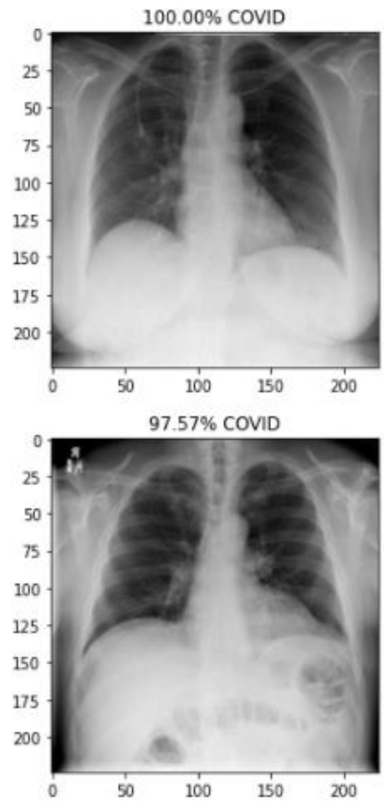


Fig. 11 InceptionV3 Model: Visualization of predictions of chest X-rays

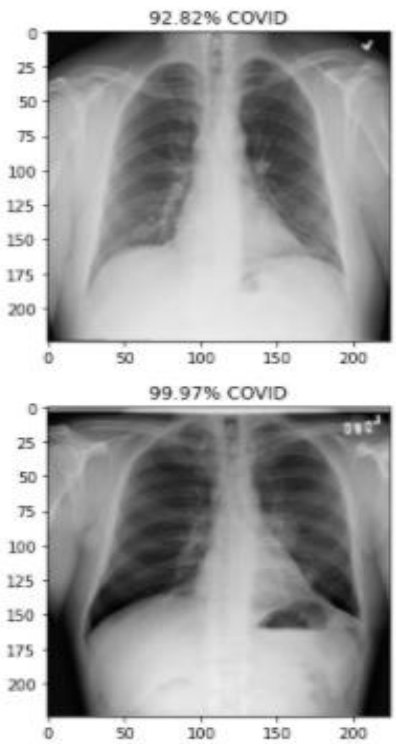


Fig. 12 ResNet 50 Model: Visualization of predictions of chest X-rays

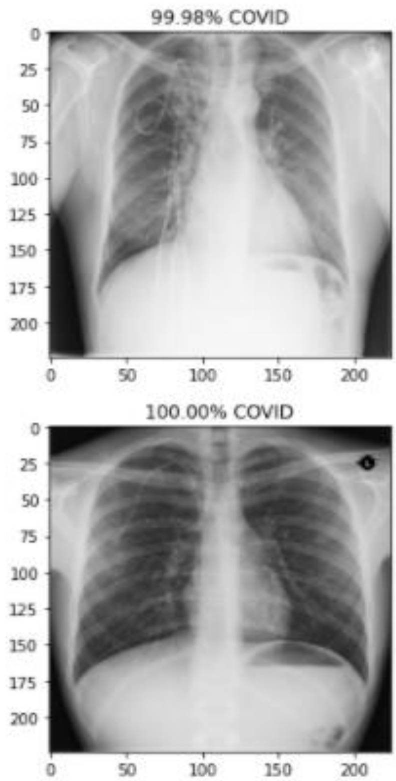


Fig. 13 VGG19 Model: Visualization of predictions of chest X-rays

Then we plotted some results and plots to get an idea about the accuracy of the four models.

Table 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 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 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 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

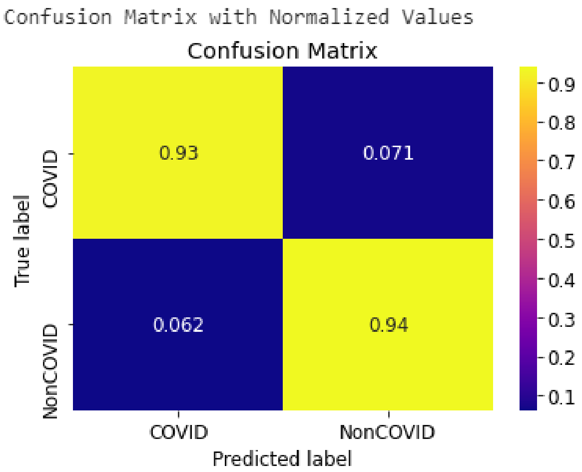


Fig. 14 Confusion Matrix with Normalized Values for Xception Model.

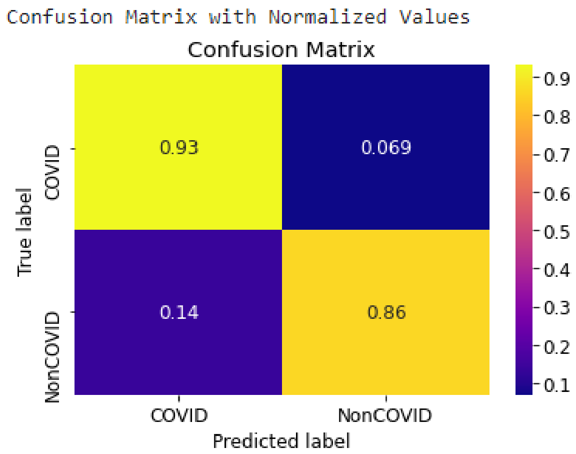


Fig. 15 Confusion Matrix with Normalized Values for InceptionV3 Model.

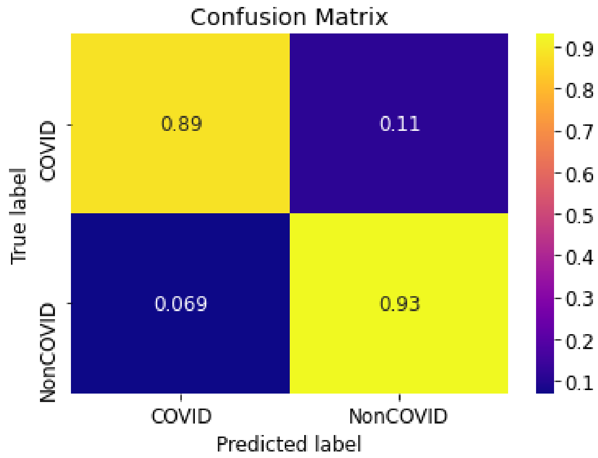


Fig. 16 Confusion Matrix with Normalized Values for ResNet50 Model.

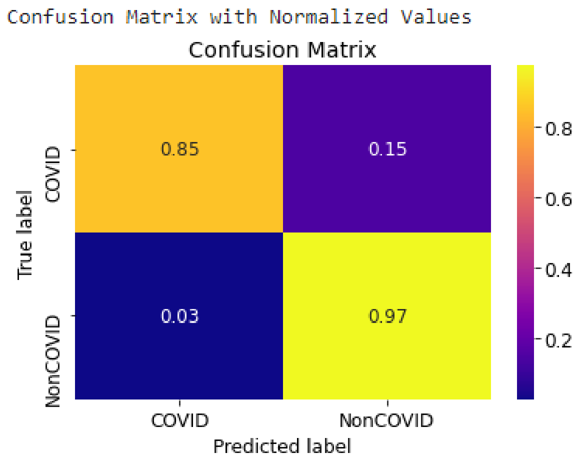
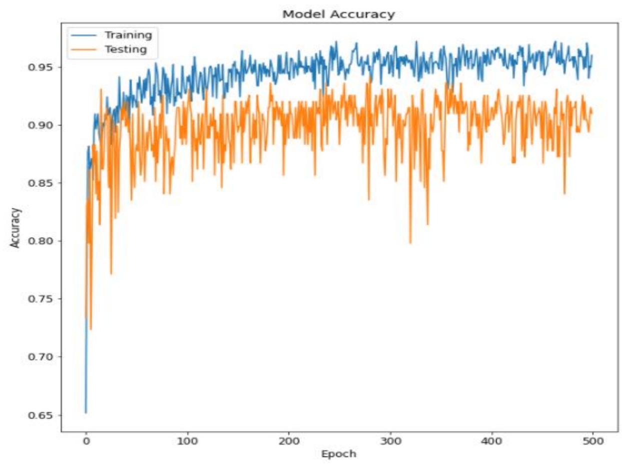
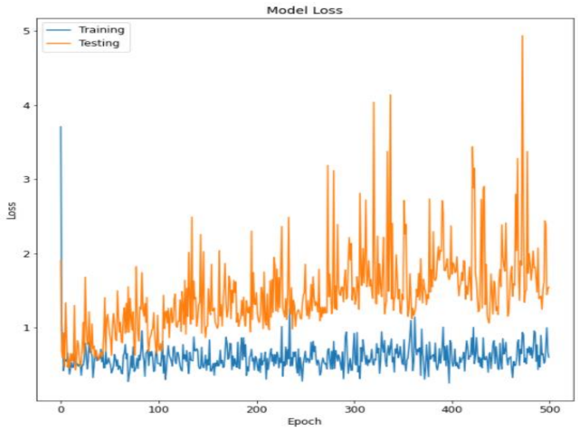


Fig. 17 Confusion Matrix with Normalized Values for VGG19 Model.

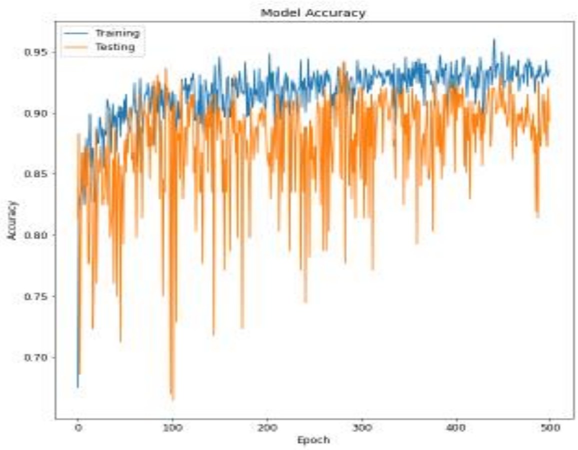


(a)

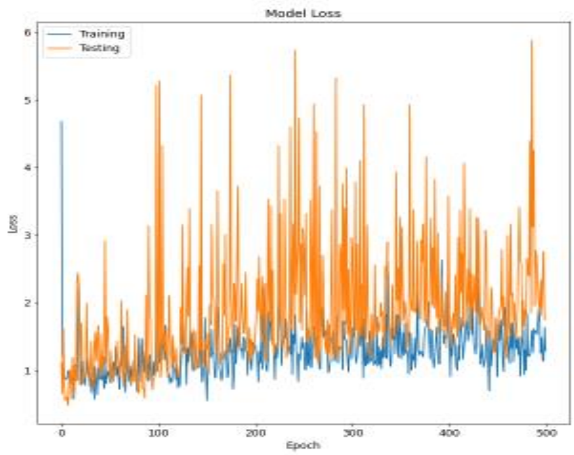


(b)

Fig. 18 Xception Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.

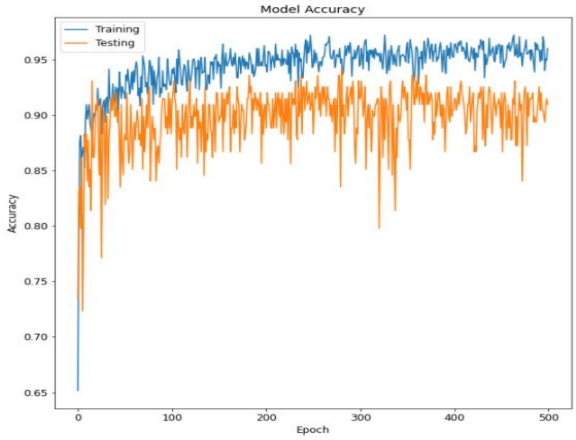


(a)

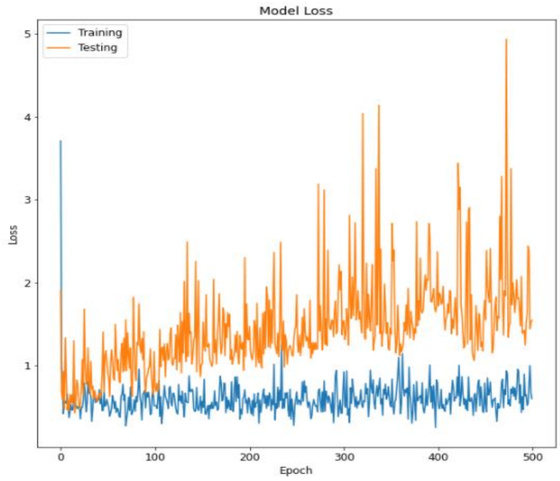


(b)

Fig. 19 InceptionV3 Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.

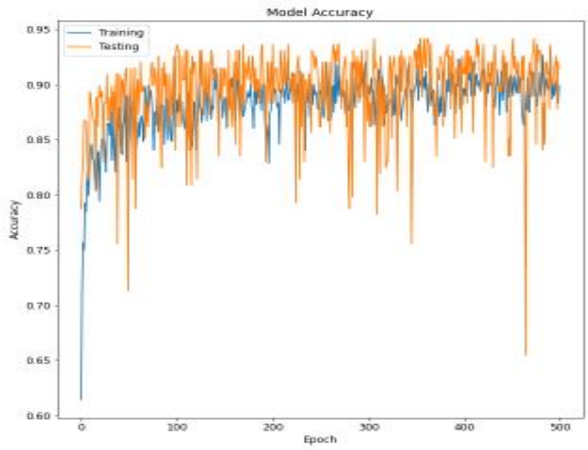


(a)

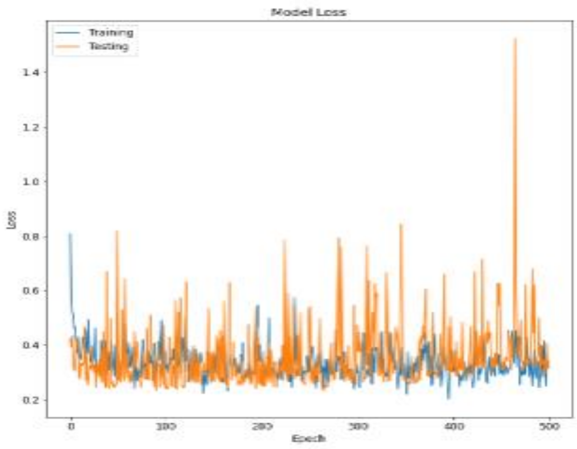


(b)

Fig. 20 ResNet50 Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.



(a)



(b)

Fig. 21 VGG19 Model (a) Accuracy plot for training and testing. (b) Loss plot for training and testing.

4. Conclusion & Future Scope

COVID-19 is burgeoning very quickly. With the increase in the number of cases and deaths day by day, there is an intense need for quick detection of COVID cases. In this work, we have analyzed four CNN models to classify people who are affected with COVID using their chest X-rays. Our Xception model achieved the best accuracy with 93%. VGG19 & ResNet50 models achieved an accuracy of 91%. Inception V3 model achieved an accuracy of 89%. We have compared all our models using plots and confusion matrices. We successfully classified COVID-19 scans, and it shows the possible scope for the future to automate diagnosis tasks.

Our future goal is to use it on a larger dataset to see if we can still achieve this high accuracy and make better predictions as in machine learning, if you train a model on large data, the predictions on unseen data are more accurate. As this is not clinically approved, it can be consulted with a professional to get a clear picture of how far the models are practically viable. As this is just an economically feasible solution, we can see that this research is put into practice in near future.

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. Hartenstein, A., Lübbe, F., Baur, A. D., Rudolph, M. M., Furth, C., Brenner, W., ... & Penzkofer, T. (2020). Prostate cancer nodal staging: using deep learning to predict 68 Ga-PSMA-positivity from CT imaging alone. *Scientific reports*, *10*(1), 1-11.

8. 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.

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*.

11. Jabber, B., Lingampalli, J., Basha, C. Z., & Krishna, A. (2020, December). Detection of Covid-19 Patients using Chest X-ray images with Convolution Neural Network and Mobile Net. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 1032-1035). IEEE.

12. Barstugan, M., Ozkaya, U., & Ozturk, S. (2020). Coronavirus (covid-19) classification using ct images by machine learning methods. *arXiv preprint arXiv:2003.09424*.

13. Cohen, J. P., Dao, L., Roth, K., Morrison, P., Bengio, Y., Abbasi, A. F., ... & Duong, T. (2020). Predicting covid-19 pneumonia severity on chest x-ray with deep learning. *Cureus*, *12*(7).

14. El Asnaoui, K., & Chawki, Y. (2020). Using X-ray images and deep learning for automated detection of coronavirus disease. *Journal of Biomolecular Structure and Dynamics*, 1-12.

15. Jain, R., Gupta, M., Taneja, S., & Hemanth, D. J. (2021). Deep learning based detection and analysis of COVID-19 on chest X-ray images. *Applied Intelligence*, *51*(3), 1690-1700.

16. 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*, *121*, 103792.

17. Chouhan, V., Singh, S. K., Khamparia, A., Gupta, D., Tiwari, P., Moreira, C., ... & De Albuquerque, V. H. C. (2020). A novel transfer learning based approach for pneumonia detection in chest X-ray images. *Applied Sciences*, *10*(2), 559.

18. Narin, A., Kaya, C., & Pamuk, Z. (2020). Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *arXiv preprint arXiv:2003.10849*.

19. Jaiswal, A. K., Tiwari, P., Kumar, S., Gupta, D., Khanna, A., & Rodrigues, J. J. (2019). Identifying pneumonia in chest X-rays: A deep learning approach. *Measurement*, *145*, 511-518.

20. Haritha, D., Pranathi, M. K., & Reethika, M. (2020, October). COVID Detection from Chest X-rays with DeepLearning: CheXNet. In *2020 5th International Conference on Computing, Communication and Security (ICCCS)* (pp. 1-5). IEEE.

21. Ucar, F., & Korkmaz, D. (2020). COVIDiagnosis-Net: Deep Bayes-SqueezeNet based diagnosis of the coronavirus disease 2019 (COVID-19) from X-ray images. *Medical Hypotheses*, *140*, 109761.

22. Sharma, A., Rani, S., & Gupta, D. (2020). Artificial intelligence-based classification of chest X-ray images into COVID-19 and other infectious diseases. *International journal of biomedical imaging*, *2020*.