Department of Electronics and Communications Engineering

EAST WEST UNIVERSITY

Dhaka, Bangladesh

June 2022

Automatic Covid-19 Detection Model from Chest X-ray using CNN, VGG16 and Best Fit Accurate Model

This Research Project has been submitted of the requirements for the degree of Bachelors of Science in Information and Communications Engineering

Course Code: ICE 498

Submitted by

Sheikh Shahed Ahmed Shaon

ID: 2015-3-50-005

Supervised by

S. M. Raiyan Chowdhury

Declaration

The Thesis entitled ‘Automatic Covid-19 Detection Model from Chest X-ray using CNN, VGG16 and Best Fit Accurate Model’ is considered under the guidance and supervision of S. M. Raiyan Chowdhury Sir. This report is the requirement for the successive competition of B.Sc. in Information and Communications Engineering under the department of Electronics and Communications Engineering.

I hereby declare that this thesis represents my own work which has been done after registration for the degree of B.Sc. at East West University.

Student’s Name and Signature:

_____

Sheikh Shahed Ahmed Shaon

ID: 2015-3-50-005

Approval

In partial compliance with the criteria, the report entitled ‘Research on **Automatic Covid-19 Detection Model from Chest X-ray using CNN, VGG16 and Best Fit Accurate Model**’ was submitted to the following respected members of the Board of Examiners of Faculty of Science and Engineering.

For a Bachelor’s Science degree in Information and Communications Engineering. The following students were accepted satisfactory in June, 2022.

Name: Sheikh Shahed Ahmed Shaon

ID: 2015-3-50-005

Supervisor

S. M. Raiyan Chowdhury

Lecturer

Department of Electronic and Communications Engineering

East West University, Bangladesh.

Dr. Mohammad Arifuzzaman

Associate Professor and Chairperson

Department of Electronics and Communications Engineering

East West University, Bangladesh.

Acknowledgement

I am really grateful because I manage to complete my thesis paper successfully by my supervisor. The completion of my thesis work brings with it a sense of satisfaction, but it is never compete without thinking everyone who made it possible.

Firstly, I like to express my deep and sincere gratitude to my research supervisor S. M. Raiyan Chowdhury Sir, Lecturer, Department f Electronics and Communications Engineering, East West University, for giving me the opportunity to do research and providing invaluable guidance throughout this research. I am thankful to my supervisor S. M. Raiyan Chowdhury Sir for the guidance and encouragement in finishing this thesis and also for teaching us.

My deepest regards to Dr. Mohammad Arifuzzaman Sir, Associate Professor and Chairperson of the Department of Electronics and Communications Engineering, for encouraging and inspiring us to carry out the thesis work.

I would also thankful to my Advisor and all faculty members and staffs of Department of ECE for providing us with the require facilities and support towards the completion of the thesis.

Finally, I am thankful to all of the people who have supported me to complete the research work directly or indirectly.

Abstract

This paper describes chest X-Ray images classification like Covid-19 infected chest images or normal chest images using various types of deep learning model. These are VGG16(Transfer learning) and CNN (Convolution Neural Network). Here We made a comparison among VGG16, CNN, SVM models and collected results that which deep learning is more accurate to identify Covid-19 or normal. These models were applied for same dataset and dataset was randomly chosen almost 1350 images (Covid-19 and Normal both) from a website. For this work, at first, we have preprocessed the chest X-Ray image. Then we have extracted the distinct features from the chest X-Ray images. After that, these features have trained into various Deep Learning algorithm and finally classify these images into the category. From the experiment, Convolution Neural Network (CNN) model achieving highest accuracy more than others. The CNN models achieving training accuracy of up to 100% and validation accuracy 94.5% and the VGG16 models achieving training accuracy up to 99.1% and validation accuracy 94.2%. Then validating the CNN model how it detects COVID-19 or normal. After that, best fit accurate model can be easily identified.

Keywords- COVID-19 Chest X-Ray, Convolution Neural Network, VGG16, Transfer learning

Contents **Page no**

Declaration i

Approval ii

Acknowledgement iii

Abstract iv

List of Figures vii

List of Tables viii

Chapter 1: Introduction (1-2)

1.1 Motivation 2

1.2 Purpose 2

Chapter 2: Architectures of Deep Learning (3-10)

2.1 Artificial Neural Network 3

2.2 Multi-layer Artificial Neural Networks and Deep Learning 4

2.3 Convolutional Neural Network 4

2.4 VGG16 (Transfer Learning) 5

2.5 Convolutional Layer 7

2.6 Pooling Layer 8

2.7 ReLU Layer 9

2.8 Dropout 9

2.9 Softmax, Loss and Regularization 10

2.10 Epoch 10

Chapter 3: Methodology (11-16)

3.1 Hardware and Software 11

3.2 Dataset collection 11

3.3 Data Preprocessing and Training of Parameter Settings 13

3.4 Network Designing 14

Chapter 4: Results and Validations (17-26)

4.1 CNN Model Summary and Results 17

4.2 VGG16 Model Summary and Results 20

4.3 Validation of COVID-19 Detection 23

Chapter 5: Conclusion and Future Goal 27

References 28

Appendix (i-vii)

List of Figures Page no

Figure 2.1: Artificial Neural Network architecture 3

Figure 2.2: Architecture of CNN 5

Figure 2.3: VGG16 architecture 6

Figure 2.4: Comparative diagram between ML & VGG16 7

Figure 2.5: Convolutional layer 8

Figure 2.6: Max-pooling Layer 9

Figure 2.7: ReLU Layer 9

Figure 2.8: Dropout Layer 10

Figure 3.1: Few samples of dataset Chest X-Ray (a) COVID-19 (b) Normal 12

Figure 3.2: The overall architecture 13

Figure 3.3: (a) Design of CNN, (b) Design of VGG16 16

Figure 4.1: (a) Training loss and Validation loss, (b) Training Accuracy and Validation Accuracy 19

Figure 4.2: (a) Training loss and Validation loss, (b) Training Accuracy and Validation Accuracy 22

Figure 4.3: Test input image 1 for detection 23

Figure 4.4: Test input image 2 for detection 24

Figure 4.5: Test input image 3 for detection 25

List of Tables Page no

Table 3.1: Details of Chest X-Ray Dataset 12

Table 3.2: Preprocessing and training phase parameters 14

Table 4.1: Deep Neural Networks Covid-19 classification evaluation accuracy score 17

Table 4.2: Summary of CNN Model 18

Table 4.3: Summary of VGG16 Model 20

Chapter 1

Introduction

Covid-19 has become an impactful issue in today's world. This virus has created a pandemic situation all over the world. The outbreak began in Wan, Hubei Province, China, in late December 2019 and has since spread around the world, including to Bangladesh. The first case of the virus was discovered in Bangladesh on March 8, 2020, and 10 days later, on March 18, the first person was infected. After that, the incidence of infection in Bangladesh rapidly grew. [1]. The patients have been infected with the Covid-19, their lungs have been largely devastated, and they will never fully recover, according to health professionals. The lungs of Covid-19 are badly injured, with severe shortness of breath, coughing, and exhaustion being the most typical symptoms. An X-ray of an infected person's chest indicates the state of their lungs and the extent of their damage. There has been a lot of study on Covid-19 chest X-Ray images in the last year, including X-Ray image classification, lung image, tumor classification, blood cell detection, and so on. Machine learning techniques are currently being used in a lot of research on coronavirus infected lung pictures. [2].

In this paper, we describe automatic COVID-19 detection by using the public database of COVID-19 cases with chest X-ray images. We believe that this database can dramatically improve identification of COVID-19. Notably, this would provide essential data to train and test Few Deep Learning based system, likely using some form of Convolution Neural Network (CNN) Vgg16(transfer learning). These tools could be developed to identify COVID-19 characteristics as compared to other types of pneumonia or in order to predict survival. Currently, all images and data are released under the following URL:

https://data.mendeley.com/datasets/8h65ywd2jr/3

Here we randomly chose 720 images of COVID-19 positive and 602 images of normal chest X-Ray. Because the dataset is huge and almost 17000 images in that website.

1.1 Motivation

In this part, we will introduce the COVID-19 problem. There are very less amount of papers COVID-19 related tasks that have been published in various international journals. COVID-19 pandemic is very challenging of our daily lives. Deep learning regression algorithm help to predict the corona virus cases in Bangladesh. In deep learning, it can be utilized huge data to predict the breakout of the disease and they used remote cloud model to prediction the corona virus. In this paper, we have discussed about some deep learning models like CNN, VGG16 that we designed to predict number of COVID-19 cases, the affected cases and find out which model will give the best accuracy.

1.2 Purpose

The purpose of this paper is to generate an experimental deep learning model like Convolutional Neural Network, with VGG16 (Transfer learning) model which has default layers and training them by same dataset. Then making a comparison of their accuracy, loss and checking the training accuracy and validation accuracy of them to identify which model will give better performance. At last, checking validation of COVID-19 detection of that model because can this model show proper output or not.

Chapter 2

Architectures of Deep Learning

2.1 Artificial Neural Network

An artificial neuron is a computational model that inspired by the biological neurons and artificial neural network (ANN) is a set of layers of neurons (It's called units or nodes). In the case of a fully connected ANN, each unit in a layer is connected to each unit in the next layer. Natural neurons receive signals through synapses located on the dendrites or membrane of the neuron and artificial neuron [3]. ANNs are more effective to solve problems related to pattern recognition and matching, clustering and date classification using mathematical algorithms are well suited for linear programming, arithmetic and logic calculations [4]. There is an input layer, hidden layers and output layers.

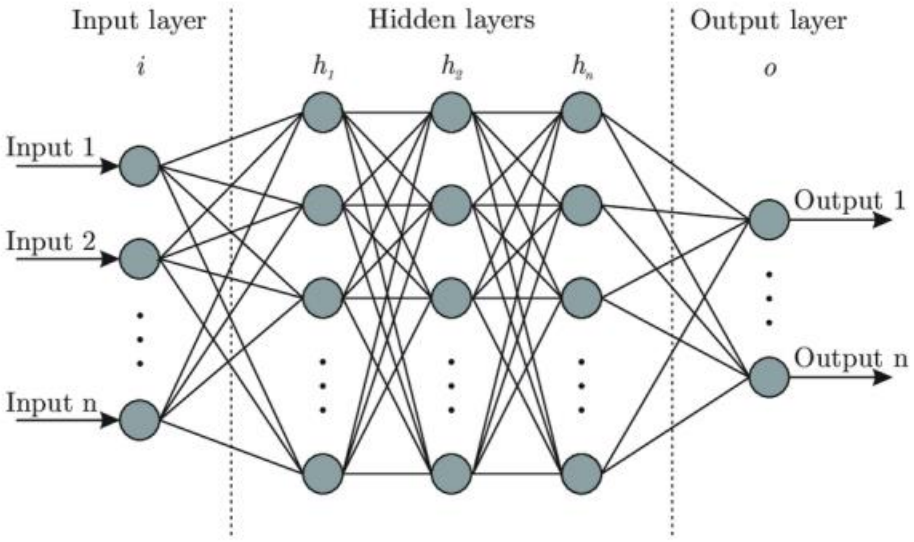


Figure 2.1: Artificial Neural Network Architecture

2.2 Multi-layer Artificial Neural Networks and Deep Learning

Multilayer ANN are used to called Deep learning. Current applications of DNN concern image and speech recognition, text recognition in images for real-time translation. The basic idea of DNN is partially inspired to the hierarchical models of our visual system. This consists of neural networks distributed with a layered topology. The complex pathways in our brain goes from the retinas, to the visual cortex, to the occipital cortex, and finally they reach high-level associative areas. Along these paths, the receptive fields at one level of the hierarchy are constructed by combining inputs from units at a lower level. There are various types of Deep Neural Networks and among the networks addressed to supervised learning, one of them is Convolutional Neural Network which is most uses network [4].

2.3 Convolutional Neural Network (CNN)

Convolutional neural networks are well suited to tasks such as object recognition, image classification, and text analysis. In 1989 CNN have been first introduced and in recent years to the increasing of GPU power and to the availability of huge datasets for training, have been largely used in computer vision tasks. The key observation is that many natural signals are a composition of low-level features [5]. The structures of CNN inspired by the visual cortex in animals [6], where groups of cells are sensitive to a small subregion of the input image. Therefore, the image is not processed as a single block but as a composition of smaller features. The absence of completely connected layers in the initial and central parts of the architecture feed-forward neural networks. Only fully connected layers are employed to construct the output, the classification probability distribution. From a computational perspective, this translates into models that require a smaller number of weights even for a large number of layers, and are therefore more tractable from the point of view of memory occupation. The training process is executed with backpropagation as in the feed-forward neural network [7].

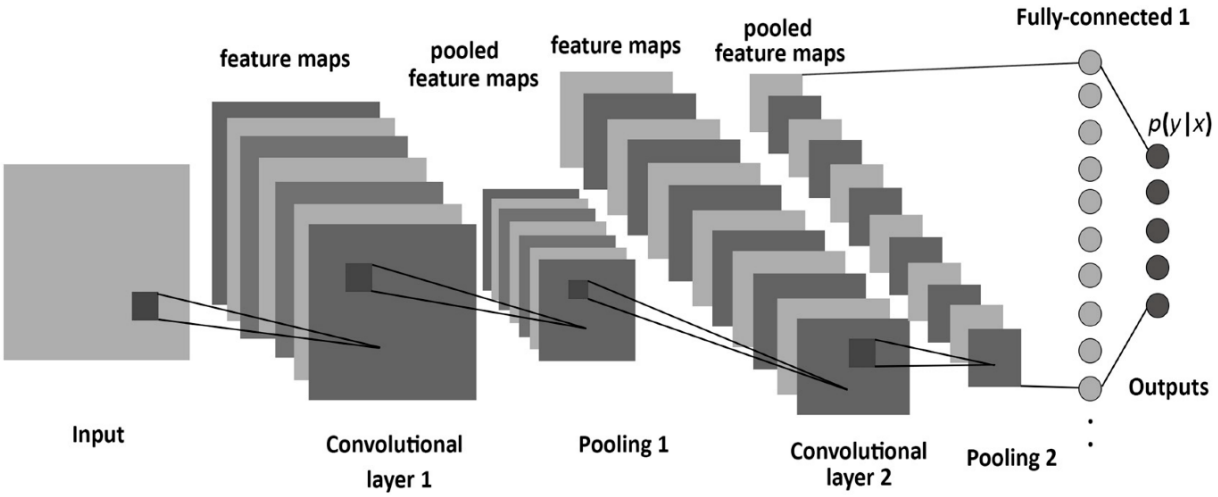


Figure 2.2: Architecture of CNN

2.4 VGG16 (Transfer Learning)

VGG16 is one kind of Convolutional Neural Network. VGG16 was created by an investigation of the effect in accuracy in increasing the depth of a convolutional network, using mostly 3x3 convolution filters. This work showed an important improvement on the prior-art architectures increasing the depth to 16 weight layers [8]. This Transfer learning improves learning by transferring knowledge from related tasks that have been learned, i.e., transferring learned and trained parameters to a new model to help with its training. The architecture of deep learning models is complex and data dependent requiring much data to train them. Much COVID-19 data are published online, but the number of samples is small, making it difficult to train a deep learning model from start to finish. Transfer learning can facilitate the training of such a small sample dataset to achieve the research purpose [9]. Transfer learning would significantly improve the performance of learning. The main idea behind transfer learning is to borrow labelled data or knowledge extracted from some related domains to help a machine learning algorithm to achieve greater performance in the domain of interest [10].

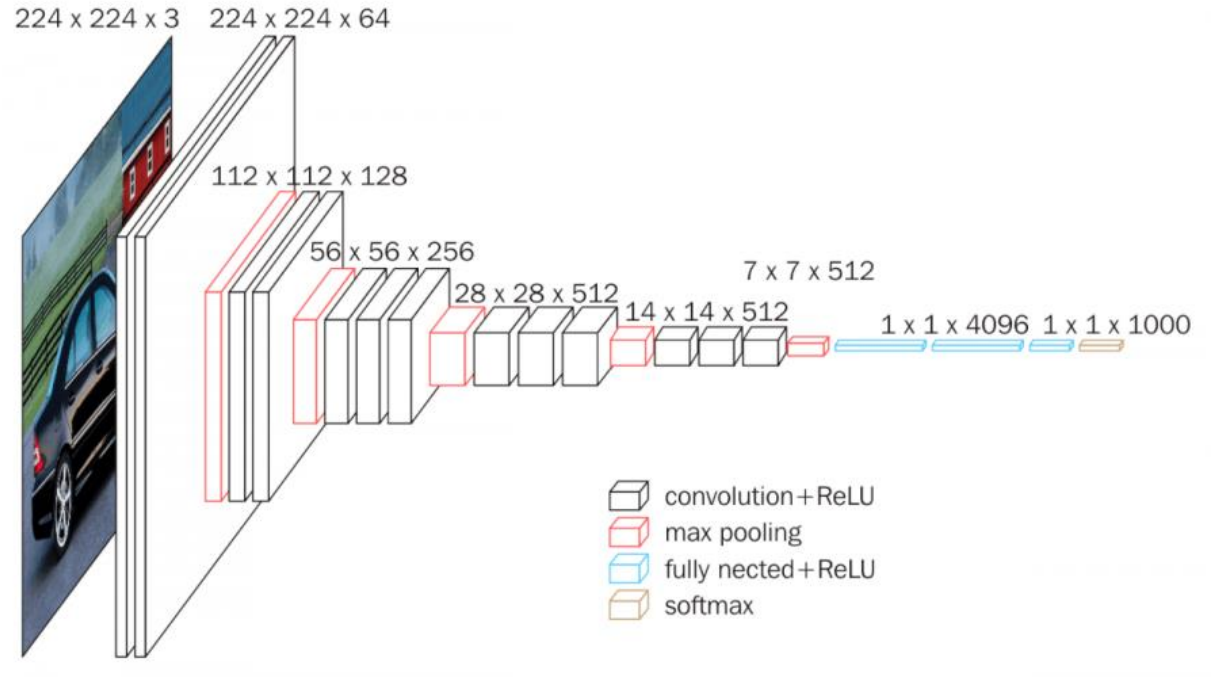


Figure 2.3: VGG16 architecture

The contrast between the processes of conventional machine learning and transfer learning. As we can see in a conventional machine learning, it tries to learn each disparate task separately with different learning system, while transfer learning tries to extract the knowledge from previous source tasks to a target task where the latter has very few labelled data for supervised learning. [10]

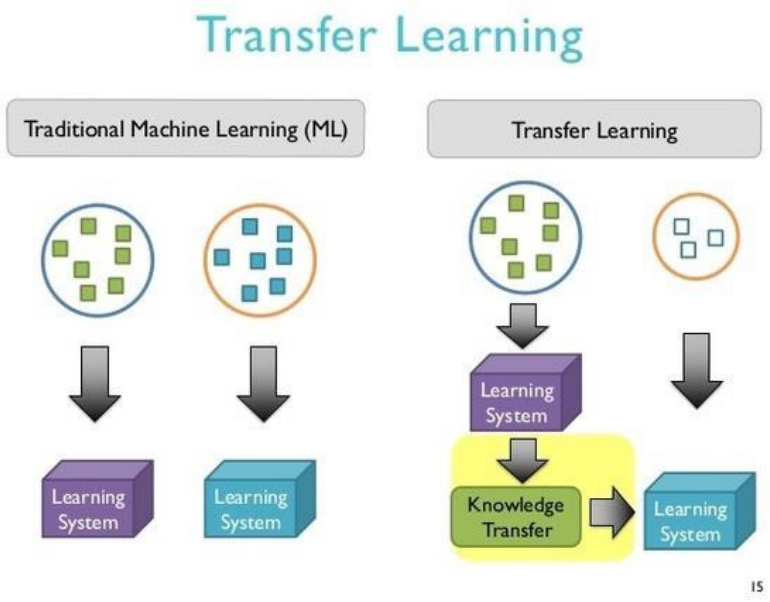


Figure 2.4: Comparative Diagram between ML and Transfer Learning

2.5 Convolutional layer

Convolutional layers are important building blocks which become the backbone of CNNs. A convolution is the straightforward use of a filter to an information that results in an activation [11]. Convolutional is the first layer to extract features from an input image. The convolutional layer plays an essential role in how CNNs operate and the parameters of the layer revolve around the usage of learnable kernels. These kernels are usually small in spatial dimensionality, but spreads along the entirety of the depth of the input. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map [12].

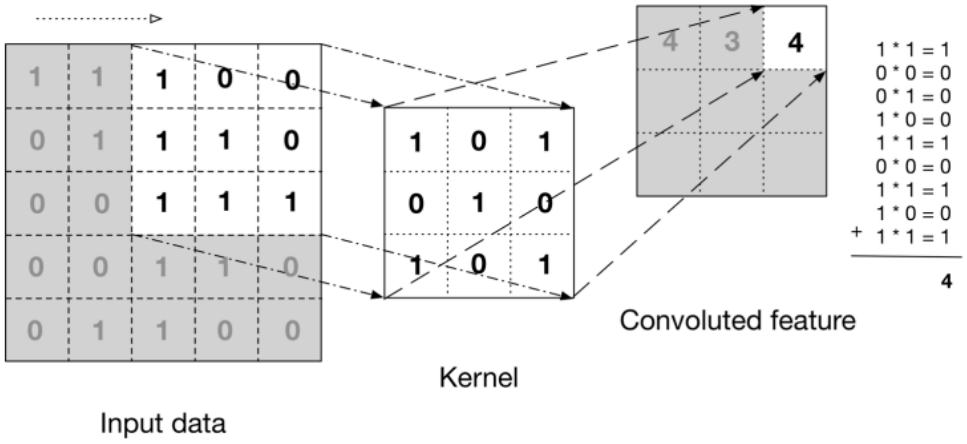


Figure 2.5: Convolutional layer [11]

2.6 Pooling layer

Pooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number of parameters and the computational complexity of the model. The pooling layer operates over each activation map in the input, and scales its dimensionality using the "MAX" function. In most CNNs, these come in the form of max-pooling layers with kernels of a dimensionality of 2 × 2 applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25% of the original size - whilst maintaining the depth volume to its standard size [12]. Max pooling outputs the maximum value of the elements in the portion of the image covered by the filter, while average pooling returns the average value. Max pooling is better at extracting dominant features and therefore considered more performance [13].

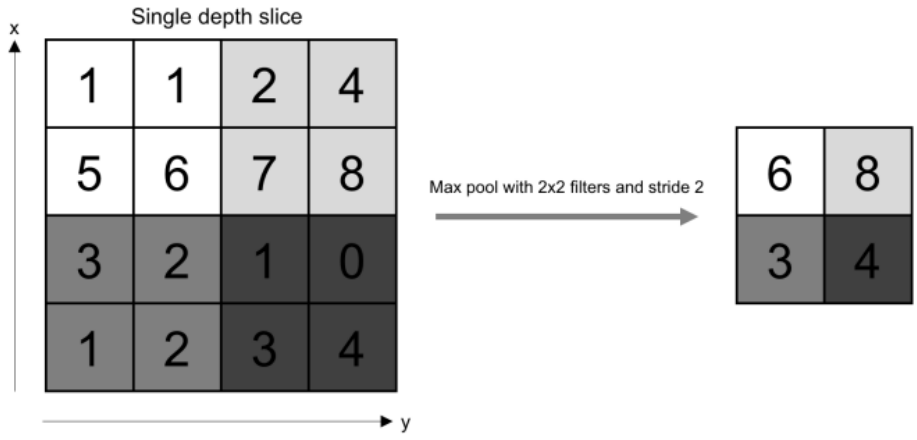


Figure 2.6: Max-pooling Layer [5]

2.7 ReLU layer

The Rectified Linear Unit or ReLU has become very popular in the last few years as activation function. This layer is placed always after a convolution layer. It computes the function There are some positive and negative aspects in using ReLU as activation function. Instead of using more complex functions such as tanh or sigmoid, ReLU simply make a threshold activation at zero. Its linear form for x>0 also performs a better propagation than a saturated non-linear function and so accelerates the training convergence using the stochastic gradient descent [8][13].

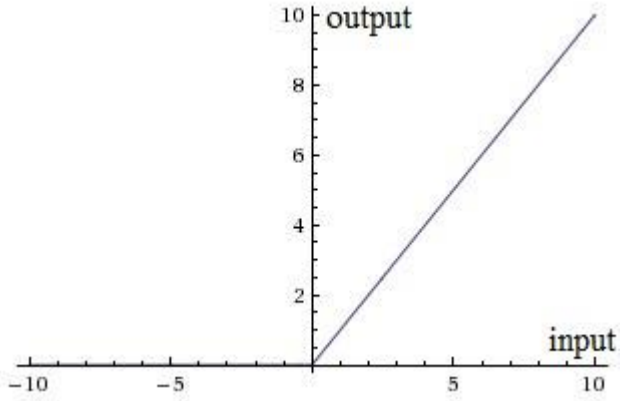


Figure 2.7: ReLU layer

2.8 Dropout layer

The dropout layer is useful to eliminate the overfitting of the data during the training process and it is applied to the fully connected layers at the end of a neural network. In practice it drops out randomly an amount of neuron units proportional to its setting parameter. For example, a dropout of 0.5 connects only 50% of the units and applying this during training makes that different units are trained randomly for each iteration. This can prevent to overfit the data. At inference, all units are present in the network, but the output weights are scaled by the probability of its presents during training, so in the example 0.5*wi. for other details take reference to the paper [8].

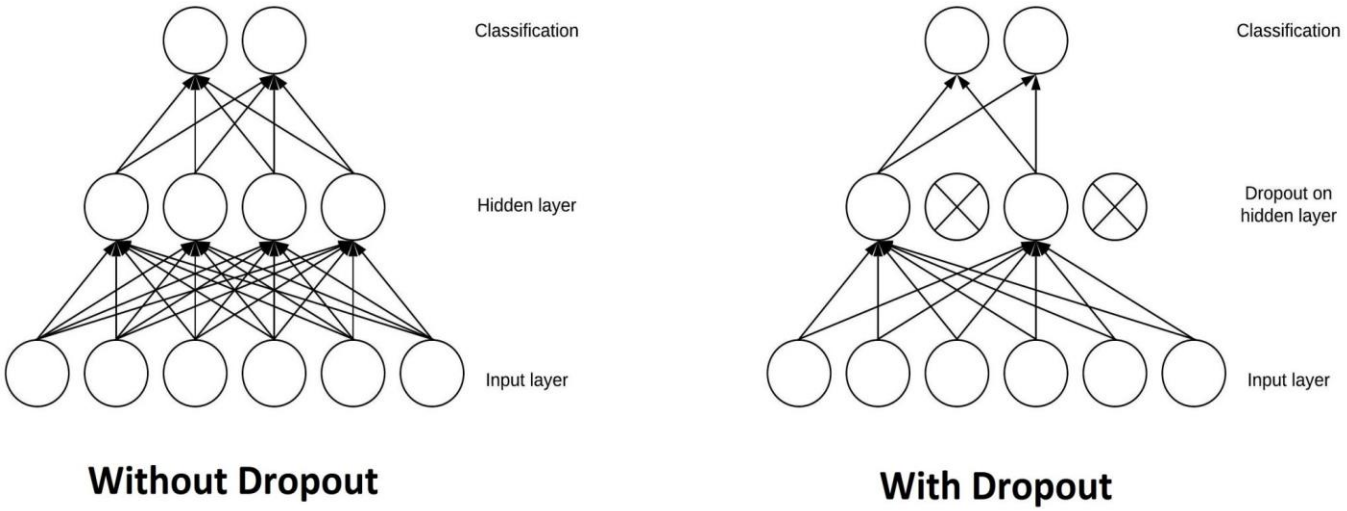


Figure 2.8: Dropout Layer

2.9 Softmax, Loss and Regularization

The Softmax, also called multinomial logistic regression, is a math function that normalize the predicted scores of the classes. The scores are the output values of a convolutional neural network. They are placed in a vector form as outputs of the classifier, where each position corresponds to a class, and so in an inference test the maximum value of the score vector should correspond to the correct output class [8]. Softmax layer uses to classify the features extracted from the FC-layers. For the softmax layer which is the last dense layer, the unit number depends on the number of categories. The softmax layer outputs the multinomial distribution of the probability scores based on the classification performed [14].

2.10 Epoch

An epoch is made up for one or more batches and one epoch is done when the whole dataset for every training sample is fed forward and backward through the neural network only once [13].

Chapter 3

Methodology

The detection of COVID-19 chest X-Ray in this paper requires several stages, as shown in Figure 3.2. The original X-ray image is preprocessed, including size adjustment, rotation, position translation, cross-cutting transformation, scaling, and flip processing at CNN and VGG16. The dataset is then divided into training and validation (test) sets. The preprocessed data are used to extract the modal feature information of the X-ray images through pretraining models by transfer learning, and this is input to the fully connected (FC) layer and trained after fusion. At CNN, 256 convolutional layers included at first convolution layer. Then at VGG16, the first two layers of the FC layer contain 512 hidden units, followed by the ReLU activation function, and the last layer contains a hidden unit, followed by the sigmoid activation function, which is used to detect COVID-19. The performance of the system is evaluated by indices such as training accuracy, validation accuracy, precision, and loss score.

3.1 Hardware and Software

The thesis used own GPUs from Personal Computer. The processor used in the project is Intel core i7 8^{th} generation processor, 8GB RAM with GPU Nvidia Geforce 4GB graphics. The deep learning framework used by TensorFlow with Keras API. All code was written by Python programming language at Jupyter notebook in Anaconda Navigator software.

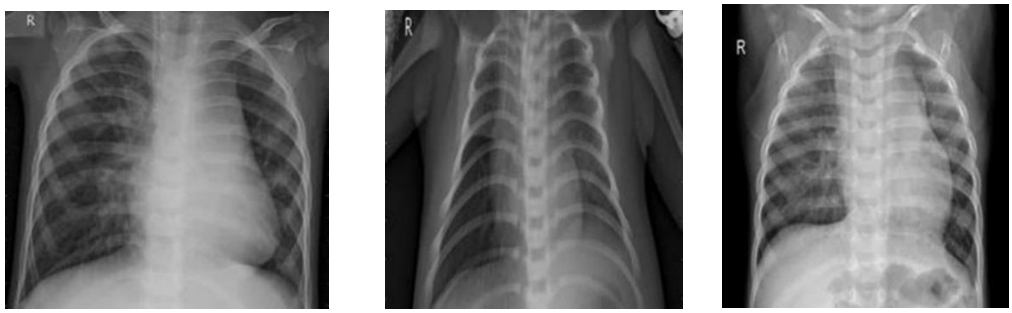
3.2 Dataset Collection

Dataset collection is the first step to developing any Deep learning network. Usually COVID-19 chest X-rays or CT scans designed to be used for computational analysis. We found a huge database of CT scan and X-Ray chest images from a website. Almost 17000 images of CT scan and X-Ray. But we only use dataset of X-Ray chest images to developing deep learning models.

We took 1322 images of chest X-Ray, where COVID-19 positive images are 720 and Normal images are 602. At figure 3.1:(a) is displaying few samples of COVID-19 positive and figure 3.2:(b) is displaying few samples of Normal chest X-Ray of dataset. At table 3.1 represents label of COVID-19 positive is 0 and label of Normal images 1.

The link of the website is https://data.mendeley.com/datasets/8h65ywd2jr/3

(a)



(b)

Figure 3.1: Few samples of dataset (a) Chest X-Ray of Covid-19 (b) Chest X-Ray of Normal

Table 3.1: Details of Chest X-Ray Dataset

Class Name	**Label**	**No. of images**
COVID-19 positive	0	720
Normal	1	602
Total		1322

3.3 Data Preprocessing and Training of Parameter Settings

We discuss and analyze the experimental results. Before training the model, we normalized the training and validation (test) datasets to decimal values between (0, 1) or (1, 1), so as to make training more convenient and faster. Since the experiment used small samples, the training and validation datasets were scaled. In the experiment, all images were rotated, zoomed, cut, and reversed in an anticlockwise direction to facilitate training. For supervised learning it is necessary to identify, instead, a set of examples consisting of appropriate inputs and corresponding outputs to be presented to the network so that it learns from them. The parameter settings for preprocessing and training are shown in Table 3.2. Here we propose the use of the softmax outputs of our neural networks as estimators [15]. We let the training of our neural networks accomplish the non-trivial task of determining a highly-optimized estimator.

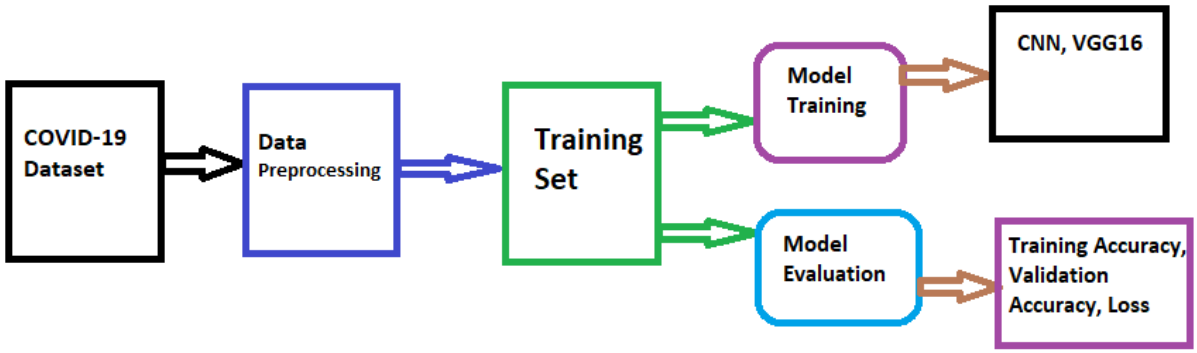


Figure 3.2: The Overall architecture

The overall architecture is at figure 3.1, first COVID-19 chest X-Ray images dataset collection including Normal images. The Data preprocessing is an important part where dataset path location, resize the images, target size, batch size, reshape, zoom range, shear range, shuffle if needed. In CNN, dataset was resized to (100x100), at VGG16 dataset input is fixed (224x224) for the model. The validation split, zoom range, shear range was same for each model but image size, shape vary. At Table 3.2, optimizer, zoom range, shear range, validation split and epoch same for all models.

Optimizer is Adam, zoom range 0.2, shear range 0.2, validation split 0.25, epochs 20. The batch size of CNN and VGG16 is same, that is 64. The loss of CNN is Sparse categorical cross entropy. The loss of VGG16 is Binary cross entropy. After data preprocessing, dataset training is the next step to training every models. The ultimate step is model evaluation. After dataset collection, preprocessing, model training the most important part is how model perform. Here we compare every model score of training accuracy, training loss and validation accuracy, validation loss.

Table 3.2: Preprocessing and training phase parameters

Parameters	**CNN**	**VGG16**
Image size	100,100	224,224
Batch	64	64
Optimizer	Adam	Adam
Loss	Sparse categorical Cross entropy	Binary cross entropy
Epoch	20	20
Validation Split	0.25	0.25
Shear range	0.2	0.2
Zoom range	0.2	0.2

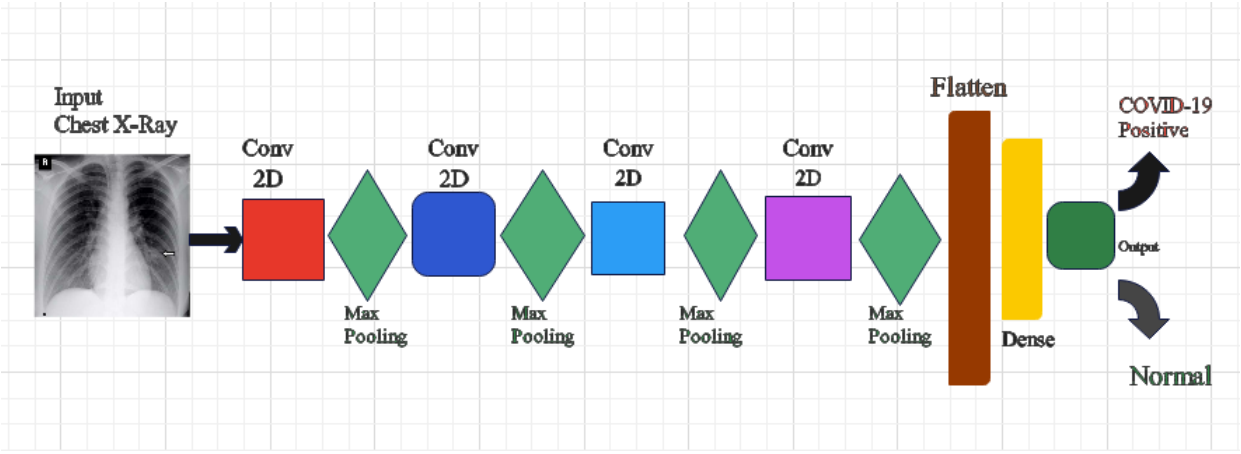
.

3.4 Network Designing

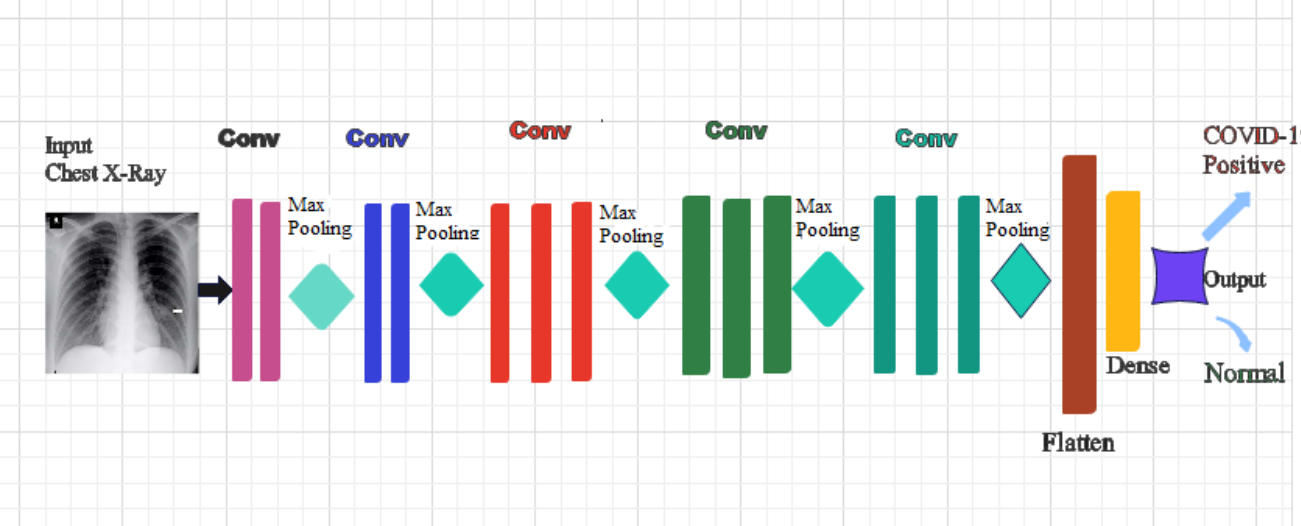
The most significant part of this thesis is Deep Neural Network designing for every individual model COVID-19 chest X-Ray dataset. In figure 3.3:(a) is the designing network of Convolutional Neural Network, 3.3:(b) is the deigning network for Transfer Learning Neural Network.

The designing of Convolutional Neural Network resized (100x100) input chest X-Ray images. Then adding first convolutional layer 2D is (256, 3x3). After Conv 2D layer a maxpooling layer and dropout layer inserted. The second Convolutional layer 2D is (128, 3x3). Then another maxpooling layer and dropout layer inserted. Now third convolutional layer 2D is (64, 3x3). A maxpooling and dropout layer inserted. Here comes the fourth convolutional layer 2D (32, 3x3). Then a maxpooling and dropout layer inserted. Then a flatten layer and dense layer inserted. After all types of convolutional layers then ReLU activation function and fully connected layer show the output COVID-19 or Normal.

The designing of VGG16 (Transfer Learning) starting with fixed (224x224) input chest X-Ray images. Usually, the architecture of VGG16 is default and we just implemented it. After input, a pair of convolutional layers 2D are (64, 3x3). After Conv 2D layer a maxpooling layer layers inserted. The second pair of Convolutional layers 2D are (128, 3x3). Then another maxpooling layer inserted. Then third convolutional layer 2D is thrice (256, 3x3). A maxpooling layer inserted. The fourth convolutional layer thrice of 2D are (512, 3x3). Then a maxpooling inserted. The fifth convolutional layers 2D are also thrice (512, 3x3). Then a maxpooling layer inserted. Then a flatten layer and dense layer inserted. After all types of convolutional layers then ReLU activation function and fully connected layer show the output COVID-19 or Normal.



(a)



(b)

Figure 3.3: (a) Design of CNN, (b) Design of VGG16

Chapter 4

Results and Validations

This project uses CNN, VGG16 models 1 to distinguish between COVID-19 and healthy people. These models were trained and tested under the same conditions (dataset, validation split and epoch). The performance of the network models was compared according to the accuracy, precision, recall, and score of the test set, which, according to Table 4.1, were training accuracy 100%, 99.09%. respectively, for CNN model were higher for other single models and model VGG16 because a single deep learning network will lose some detailed feature information when extracting COVID-19 lung modal feature information, eventually leading less amount of classification results but almost equal to CNN. However, CNN model designed less amounts of layers, less memory and time consuming. This single model to extract the modal features of the COVID-19 lung image in parallel, which can effectively retain the detailed feature information of the image, for a better final classification effect than a single model.

Table 4.1: Deep Neural Networks Covid-19 classification evaluation accuracy score

Deep Neural Networks	**Training Accuracy**	**Training loss**	**Validation Accuracy**	**Validation loss**
CNN	1.00	8.2418e-04	0.9456	0.1909
VGG16	0.9909	0.0390	0.9424	0.1188

4.1 CNN Model Summary and Results

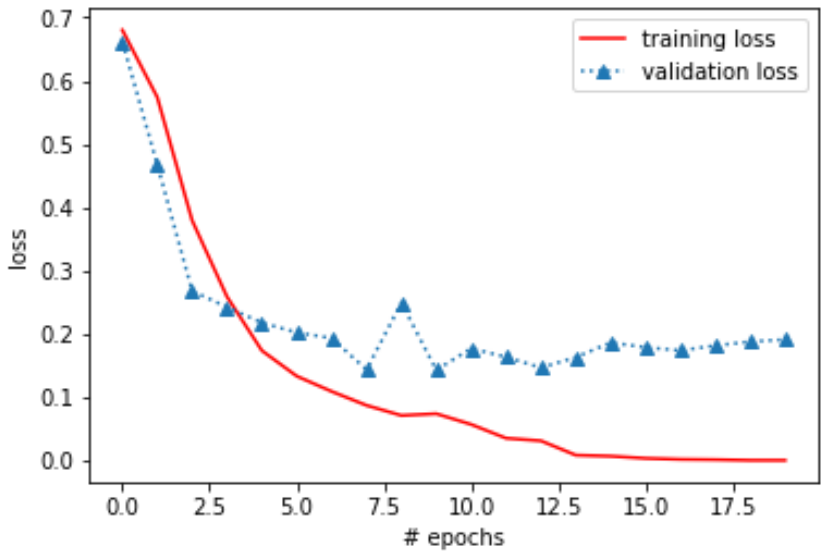
This model is based on the Convolutional Neural Network (CNN) used TensorFlow, Keras API with adjustments. There are four convolutional layers, after each is a max pooling layer, and four dropout layers with the dropout rate of 0.25 added to prevent overfitting. Then a flatten layer with 512 units. After that, there is a dense layer with 2 units. The input image resized to (100x100) and

convolutional layer started with (256x256). The batch size is 64 and the number of epochs is 20 with a validation split 0.25. The optimizer is Adam and loss is sparse categorical binary cross entropy. The total trainable parameters were 395490 and non- trainable parameters were 0.

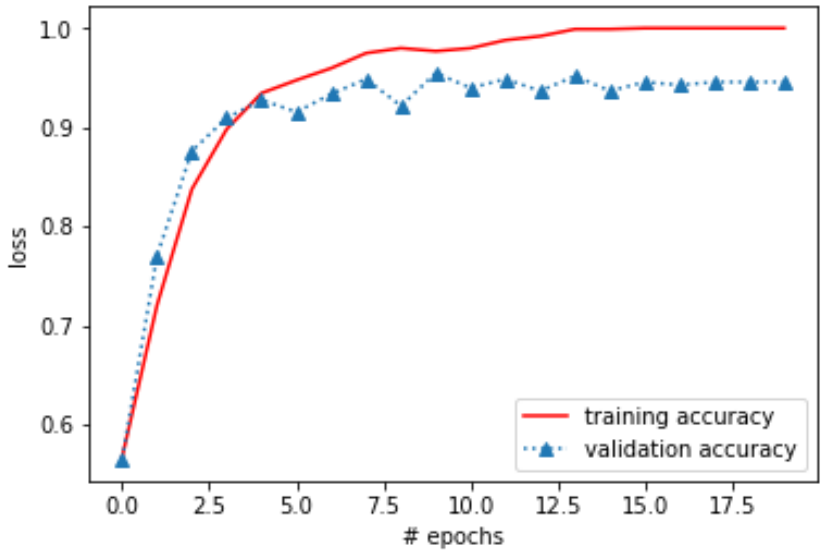
Table 4.2: Summary of CNN Model

Layer Type	**Output Shape**	**Parameter**
Conv2D 1st	(None, 98, 98, 256)	7168
MaxPooling2D 1st	(None, 49, 49, 256)	0
Dropout 1st	(None, 49, 49, 256)	0
Conv2D 2nd	(None, 47, 47, 128)	295040
MaxPooling2D 2nd	(None, 23, 23, 128)	0
Dropout 2nd	(None, 23, 23, 128)	0
Conv2D 3rd	(None, 21, 21, 64)	73792
MaxPooling2D 3rd	(None, 10, 10, 64)	0
Dropout 3rd	(None, 10, 10, 64)	0
Conv2D 4th	(None, 8, 8, 32)	18464
MaxPooling2D 4th	(None, 4, 4, 32)	0
Dropout 4th	(None, 4, 4, 32)	0
Flatten	(None, 512)	0
Dense	(None, 2)	1026
Total Parameters: 395490 Trainable Parameters: 395490 Non-Trainable Parameters: 0		

After summary of CNN, then the evaluation graph of CNN given below at figure 4.1:(a) and 4.2:(b).



(a)



(b)

Figure 4.1: (a) Training loss and Validation loss, (b) Training Accuracy and Validation Accuracy

The graph (a) of figure 4.1 represents training loss and validation loss of Convolutional Neural Network model. The graph's vertical line starting with training loss 0.6801 and to ends up to last 20^{th} epoch 8.2418e-04. The horizontal line represents number of epochs. This graph shows that training loss and validation loss decreasing or increasing after every epoch. The training loss of this model 8.2418e-04 and validation loss of this model 0.1909. The graph (b) of figure 4.1

represents training accuracy and validation accuracy of this CNN model. The red line is training accuracy and blue triangular symbolic line is validation accuracy. This graph shows that training accuracy and validation accuracy decreasing or increasing after every epoch. The training accuracy of this model 1.00 and validation accuracy of this model 0.95.

4.2 VGG16 Model Summary and Results

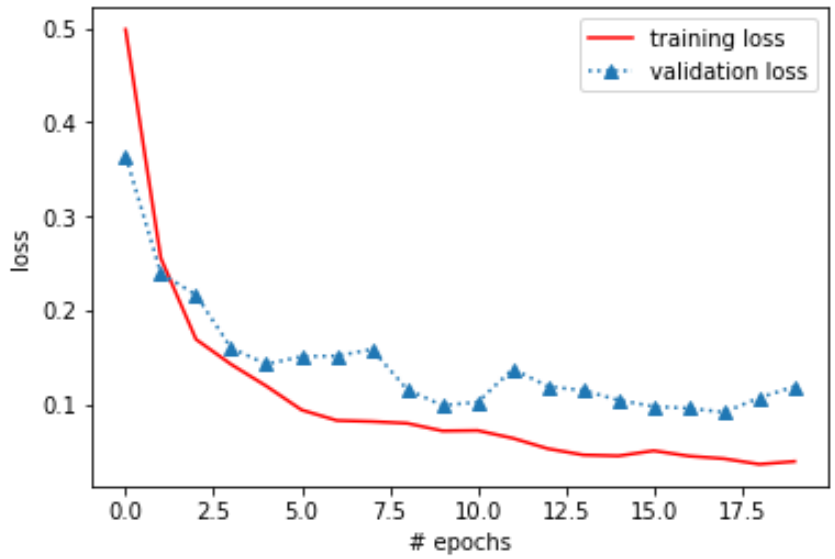
This model is based on the VGG16 (Transfer learning) used TensorFlow, Keras API with adjustments. There are five convolutional layers, after each is a max pooling layer. First two convolution layers are paired. Next three convolutional layers are triply. Then a flatten layer with 25088 units. After that, there is a dense layer with 2 units. The input layer started with (224x224), cause at VGG16 it is fixed size. The convolutional layer also started with (224x224). The batch size is 64 and the number of epochs is 20 with a validation split 0.25. The optimizer is Adam and loss is binary cross entropy. The total trainable parameters were 50178 and non-trainable parameters 14714688.

Table 4.3: Summary of VGG16 Model

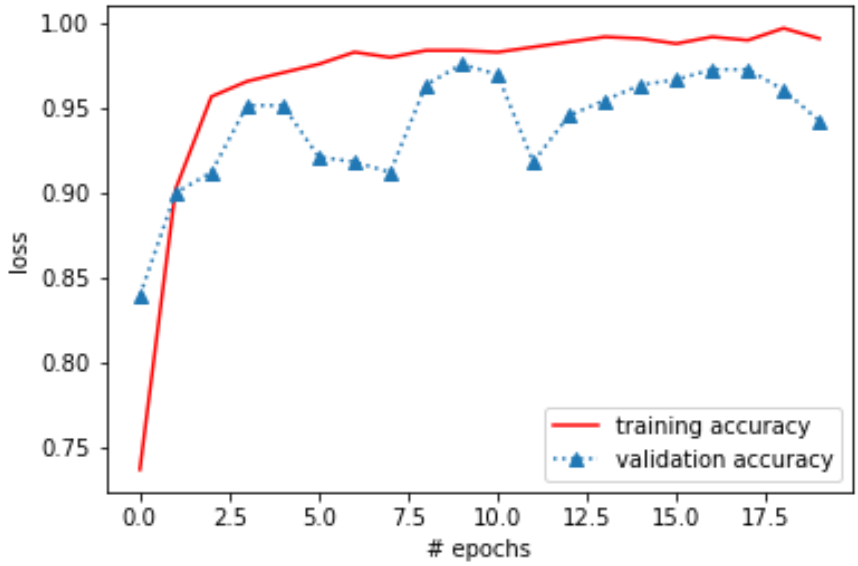
Layer	**Output Shape**	**Parameter**
Input Layer	(None, 224, 224, 3)	0
Conv2D 1st	(None, 224, 224, 64)	1792
Conv2D 2nd	(None, 224, 224, 64)	36928
Maxpooling2D	(None, 112, 112, 64)	0
Conv2D 1st	(None, 112, 112, 128)	73856
Conv2D 2nd	(None, 112, 112, 128)	147584
MaxPooling2D	(None, 56, 56, 128)	0
Conv2D 1st	(None, 56, 56, 256)	295168
Conv2D 2nd	(None, 56, 56, 256)	590080
Conv2D 3rd	(None, 56, 56, 256)	590080

MaxPooling2D	(None, 28, 28, 256)	0
Conv2D 1st	(None, 28, 28, 512)	1180160
Conv2D 2nd	(None, 28, 28, 512)	2359808
Conv2D 3rd	(None, 28, 28, 512)	2359808
MaxPooling2D	(None, 14, 14, 512)	0
Conv2D 1st	(None, 14, 14, 512)	2359808
Conv2D 2nd	(None, 14, 14, 512)	2359808
Conv2D 3rd	(None, 14, 14, 512)	2359808
MaxPooling2D	(None, 7, 7, 512)	0
Flatten	(None, 25088)	0
Dense	(None, 2)	50178
Total Parameters: 14764866 Trainable Parameters: 50178 Non-Trainable Parameters: 14714688		

After summary of VGG16, then the evaluation graph of VGG16 given below at figure 4.2:(a) and 4.2:(b).



(a)



(b)

Figure 4.2: (a) Training loss and Validation loss, (b) Training Accuracy and Validation Accuracy

The graph (a) of figure 4.2 represents training loss and validation loss of Transfer Learning Neural Network model. The graph's vertical line starting with training loss 0.4909 and end up to 0.0390. The horizontal line represents number of epochs. This graph shows that training loss and validation loss decreasing or increasing after every epoch. The training loss of this model 0.0390 and

validation loss of this model 0.1188. The graph (b) of figure 4.2 represents training accuracy and validation accuracy of this transfer learning model. The red line is training accuracy and blue triangular symbolic line is validation accuracy. This graph shows that training accuracy and validation accuracy decreasing or increasing after every epoch. The training accuracy of this model 0.9909 and validation accuracy of this model 0.9424.

4.3 Validation of COVID-19 Detection

In previous section we already evaluate all of the network models. In this section we validate the highest validation accuracy model. After evaluation, the highest validation accuracy we got from CNN and it was 94.5%. That means this model can detect 94.5% true positive and true negative. Other 5.5% probability of this model can detect false positive and false negative. COVID-19 positive detection from this model of input X-Ray chest image of Figure 4.3.

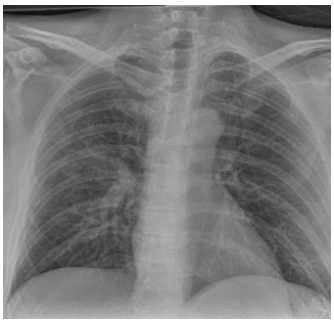


Figure 4.3: Test Input image 1

```
from keras.preprocessing import image
import numpy as np
import random
img_pred=image.load_img(r"F:\_Download\COVID-19 Dataset\X-
ray\COVID\1d435a4b.png",target_size=(100,100))

img_pred=image.img_to_array(img_pred)
img_pred=np.expand_dims(img_pred, axis=0)

rslt=model.predict(img_pred)

print(rslt)
```

```
if rslt[0][0]>rslt[0][1]:
    prediction="Covid Positive"

else:
    prediction="Normal"
print(prediction)
```

Output:

[[1. 0.]]

Covid Positive

In this process, for predicting COVID-19 We import keras, numpy and a function called image.load_img to load Chest X-Ray image. The location of predicting image put into this function and here the target size of image (100x100). We already know these types of models can't predict a raw image file. Then we convert image to array. The index [0][0] is for COVID-19 positive Chest images and index [0][1] is for normal chest images. If the value of index [0][0] is more than [0][1] then it will predict COVID-19 positive and else predict normal. At figure 18, index [0][0] was showing 1 and [0][1] was showing 0 and then it predicted COVID-19 positive.

Figure 4.4: Test input image 2

```
from keras.preprocessing import image
import numpy as np
```

```
import random
img_pred=image.load_img(r"F:\_Download\COVID-19 Dataset\X-ray\Non-
COVID\Non-COVID-19 (111).jpeg",target_size=(100,100))

img_pred=image.img_to_array(img_pred)
img_pred=np.expand_dims(img_pred, axis=0)

rslt=model.predict(img_pred)

print(rslt)
if rslt[0][0]>rslt[0][1]:
    prediction="Covid Positive"

else:
    prediction="Normal"
print(prediction)
```

Output:

[[0. 1.]]

Normal

The input X-Ray chest image of figure 4.4, A normal (COVID-19 Negative) detection from an input chest X-Ray image. Here index [0][1] is more than [0][0]. Index [0][0] was showing 0 and [0][1] was showing 1. Then it predicted Normal (COVID-19 negative).

Figure 4.5: Test input image 3

```
from keras.preprocessing import image
```

```
import numpy
as np
import random
img_pred=image.load_img(r"F:\_Download\COVID-19 Dataset\X-ray\non-
COVID\non-COVID-19 (28).jpeg",target_size=(100,100))

img_pred=image.img_to_array(img_pred)
img_pred=np.expand_dims(img_pred, axis=0)

rslt=model.predict(img_pred)

print(rslt)
if rslt[0][0]>rslt[0][1]:
    prediction="Covid Positive"

else:
    prediction="Normal"
print(prediction)
```

Output:

[[$1.0000000e^{+00}$ $1.1630734e^{-11}$]]

Covid Positive

The validation accuracy of this CNN model better than other like VGG16. But it's not 100% validate model, it's validation accuracy 94.5%. There is a probability of 5.5% that sometimes it might be detect false COVID-19 positive or false COVID-19 negative, an example of input normal chest X-Ray image and it detected COVID-19 positive at input chest X-Ray image of figure 4.5.

Chapter 5

Conclusions and Future Goal

Chest X-rays are effective tools for diagnosing and evaluating COVID-19. We used three Deep Learning models to divide X-ray samples into two categories: COVID-19 positive and normal people. We applied these model architectures for feature extraction and classified categories. The results of Experiment showed that, under the same conditions of same dataset, CNN (Convolution Neural Network) is the best fit for classifying COVID-19 and normal people. It could significantly improve classification performance, with scores of training accuracy of 100% and validation accuracy 94.5%. The performance of VGG16 (Transfer Learning) was also good and almost equal to CNN, where scores of training accuracy 99.1% and validation accuracy 94.2%. We discussed and compared our research and recent work. The results showed that Convolution Neural Network is better than other models like VGG16 in classifying and detecting COVID-19 and normal people, can accurately classify them, and can assist doctors in the rapid detection of COVID-19. We concluded from these two aspects that CNN model is well distinguished between COVID-19 patients and healthy people and in future it could help to reduce the workload of doctors in detecting COVID-19 cases.

References

[1] M. K. M. A.-A. S M Abdullah Al Shuaeb, "COVID-19 Outbreak Prediction and Forecasting in Bangladesh using Machine Learning Algorithm," *International Journal of Trend in Scientific Research and Development (IJTSRD),* vol. 5, no. 1, November,December 2020.

[2] A. R. S. M. A. A. S. A. H. Md. Abdul Matin, "Automatic Covid-19 Infected Chest X-Ray image Infected Chest X-Ray Image Classification Using Support Vector Machine," *International Journal of Trend in Scientific Research and Development (IJTSRD),* vol. 5, no. 4, p. 1, May-June 2021.

[3] C. Gershenson, "Artificial Neural Networks for Beginners," September 2003.

[4] P. Dell'Aversana, "ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING. A SIMPLE OVERVIEW".

[5] D. Caschili, "Optimization of CNN-Based Object Detection Algorithms for Embedded Systems," no. 2, p. 19.

[6] J. P. a. A. Gibson, "Deep learning: a practitioners approach," *OReilly,* 2017.

[7] Y. B. a. G. H. Yann Lecun, "“Deep learning”," 2015.

[8] V. A. Luca, "Very deep convolutional neural networks for face identification," pp. 13,14,16,21, July 2016.

[9] Z. Z. Y. Z. Q. Z. Dongsheng Ji, "Research on Classification of COVID-19 Chest X-Ray Image Modal Feature Fusion Based on Deep Learning," *Journal of Healthcare Engineering,* vol. 2021.

[10] S. Tammina, "Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images," *International Journal of Scientific and Research Publications,* vol. 9, no. 10, October 2019.

[11] P. Shah, "DESIGN SPACE EXPLORATION OF CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION," December 2020.

[12] R. N. Keiron O'Shea, "An Introduction to Convolutional Neural Networks," November 2015.

[13] H. Dao, "IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL," 2020.

[14] M. B. H. Chiranjibi Sitaula, "Attention-based VGG-16 model for COVID-19 chest X-ray image classification," 17 November 2020.

[15] H. C. ,. Z. ,. R. S.-N. J. G. A. K. K. L. K. Mahdi Mahdavi, "A machine learning based exploration of COVID-19 mortality risk," 2 July 2021.

Appendix

i. Necessary Code for Convolutional Neural Network:

```
import numpy as np

import cv2

import os

import random

import matplotlib.pyplot as plt

DIRECTORY=r"C:\Users\HP\Desktop\Vgg16_Covid\Dataset"

CATAGORIES=['COVID','Non-COVID']

data=[]


for categories in CATAGORIES:

    folder=os.path.join(DIRECTORY,categories)

    label=CATAGORIES.index(categories)


    for img in os.listdir(folder):

        img=os.path.join(folder,img)

        img_arr=cv2.imread(img)

        img_arr=cv2.resize(img_arr,(100,100))


        data.append([img_arr,label])
```

```
random.shuffle(data)

x=[]

y=[]


for features,label in data:

    x.append(features)

    y.append(label)

x=np.array(x)

y=np.array(y)

x=x/255

x.shape

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation

model=Sequential()

model.add(Conv2D(256,(3,3),input_shape=x.shape[1:],activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))

model.add(Conv2D(128,(3,3),input_shape=x.shape[1:],activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))

model.add(Conv2D(64,(3,3),input_shape=x.shape[1:],activation='relu'))
```

```
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))

model.add(Conv2D(32,(3,3),input_shape=x.shape[1:],activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(2,activation='softmax'))

model.summary()

model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

r=model.fit(x,y,epochs=20,validation_split=0.25)

from matplotlib import pyplot as plt


plt.plot(r.history['loss'],'r',label='training loss')

plt.plot(r.history['val_loss'], '^:',label='validation loss')

plt.xlabel('# epochs')

plt.ylabel('loss')

plt.legend()

plt.show()


plt.plot(r.history['accuracy'],'r',label='training accuracy')

plt.plot(r.history['val_accuracy'], '^:',label='validation accuracy')

plt.xlabel('# epochs')

plt.ylabel('loss')
```

```
plt.legend()

plt.show()
```

##Prediction

```
from keras.preprocessing import image

import numpy as np

import random

img_pred=image.load_img(r"F:\_Download\COVID-19\Dataset\Xray\COVID\1d435a4b.png",target_size=(100,100))


img_pred=image.img_to_array(img_pred)

img_pred=np.expand_dims(img_pred, axis=0)

rslt=model.predict(img_pred)

print(rslt)

if rslt[0][0]>rslt[0][1]:

   prediction="Covid Positive"

else:

   prediction="Normal"

print(prediction)
```

ii. Necessary Codes for VGG16 (Transfer Learning):

```
import tensorflow as tf

import os
```

```
import numpy as np
base_dir=r"C:\Users\HP\Desktop\Vgg16_Covid\Dataset"
IMAGE_SIZE=224
BATCH_SIZE=64
train_datagen=tf.keras.preprocessing.image.ImageDataGenerator(

    rescale=1./255,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.25)

validation_datagen=tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.25
)
train_generator=train_datagen.flow_from_directory(
    base_dir,
    target_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='training'
 )
validation_generator=validation_datagen.flow_from_directory(
    base_dir,
```

```
    target_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    subset='validation'
 )
from tensorflow.keras.layers import Input,Flatten,Dense
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Sequential
from glob import glob
IMAGE_SIZE=[224,224]
vgg=VGG16(input_shape=IMAGE_SIZE+[3],weights='imagenet',include_top=False)
vgg.output
for layer in vgg.layers:
    layer.trainable=False
folders=glob(r"C:\Users\HP\Desktop\Vgg16_Covid\Dataset\*")
print(len(folders))
x=Flatten()(vgg.output)
prediction=Dense(len(folders),activation='softmax')(x)
model=Model(inputs=vgg.input,outputs=prediction)
model.summary()
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
epoch=20
```

```
history=model.fit(train_generator,
                  steps_per_epoch=len(train_generator),
                  epochs=epoch,
                  validation_data=validation_generator,
                  validation_steps=len(validation_generator)
)

from matplotlib import pyplot as plt
plt.plot(history.history['loss'],'r',label='training loss')
plt.plot(history.history['val_loss'],'^:',label='validation loss')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()

plt.plot(history.history['accuracy'],'r',label='training accuracy')
plt.plot(history.history['val_accuracy'], '^:',label='validation accuracy')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```