

An Efficient Deep Learning Approach to Detect COVID-19 Infected Lungs Using Image Data

by

Asif Rezwan Kabir

18301230

Shutirtha Roy

18301028

Nusrat Zerine

18101533

Sheikh Sharia Afrin

18101528

Anika Jahan Choudhury

18301016

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2022

© 2022. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

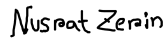
Student's Full Name & Signature:



Asif Rezwan Kabir
18301230



Shutirtha Roy
18301028



Nusrat Zerin
18101533



Sheikh Sharia Afrin
18101528



Anika Jahan Choudhury
18301016

Approval

The thesis/project titled “An Efficient Deep Learning Approach to Detect COVID-19 Infected Lungs Using Image Data” submitted by

1. Asif Rezwan Kabir (18301230)
2. Shutirtha Roy (18301028)
3. Nusrat Zerine (18101533)
4. Sheikh Sharia Afrin (18101528)
5. Anika Jahan Choudhury (18301016)

Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 16, 2022.

Examining Committee:

Supervisor:
(Member)



Dr. Md. Ashraful Alam, PhD
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)



Md. Tanzim Reza
Lecturer
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The beginning of 2020 will always be a dreadful chapter in human history. Even with all the recent advancements in the medical sector, the COVID-19 virus proved to be a major challenge for doctors all over the world. The virus affected different people in different ways. One of its deadliest symptoms can be observed in our lungs. COVID-19 can cause various complications in the lungs such as pneumonia, acute respiratory distress syndrome (ARDS), sepsis, etc. This pandemic, being highly contagious, can spread and affect a large number of the population in a very short period. This results in many patients not receiving proper treatment at the appropriate time. Our proposed CNN model will be able to automate the entire detection and classification process. It will be trained using large amounts of X-ray images of lungs, which will provide it with the necessary feature knowledge to distinguish between an infected lung and a healthy one.

Keywords: COVID-19; CNN; Supervised Learning; X-ray image; Tensorflow

Acknowledgement

Firstly, all praise to the Great Almighty for whom our thesis have been completed without any major interruption.

Secondly, to our advisor Dr. Md. Ashrafal Alam sir and co-advisor Md. Tanzim Reza sir for their kind support and advice in our work. They helped us whenever we needed help.

And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Nomenclature	viii
1 Introduction	1
1.1 Motivation	1
1.2 Research Problem	2
1.3 Research Objectives	3
2 Literature Review	4
2.1 COVID-19 & its Effects on Lungs	4
2.2 CNN Architecture	5
2.3 Supervised Learning	5
2.4 Different Types of CNN Models	6
2.4.1 Resnet	6
2.4.2 Inception Net	8
2.5 Related Works	10
3 Workplan	11
4 Methodology	13
4.1 Input data	13
4.2 Data pre-processing	14
4.3 CNN Modeling	15
5 Implementation & Results	18
5.1 Implementation	18
5.1.1 Building the model	18

5.2	Results	20
5.2.1	CNN	20
5.2.2	Inception v3	22
5.2.3	ResNet-50	24
5.2.4	Comparative Analysis of the different models	26
6	Limitations	27
7	Conclusion	28
8	Future Works	29
8.1	COVID-19 App	29
8.2	Federated Learning	31
8.3	SMOTE: Synthetic Minority Over-Sampling Technique	33
	Bibliography	35

List of Figures

2.1	Architecture of Artificial neuron and Multilayered Artificial neural network.[3]	5
2.2	Diagram shows how Input X is added to Output	6
2.3	Resnet-50 Architecture	7
2.4	Inception Layer	8
2.5	Inception Modules[29]	9
3.1	The flow chart of the proposed COVID-19 detection model using CNN	12
4.1	Sample COVID-19 Positive X-ray image from the dataset	13
4.2	Sample COVID-19 Negative X-ray image from the dataset	14
4.3	Convolution Neural Network	16
5.1	Determining the training and validation loss	21
5.2	Determining the training and validation accuracy	21
5.3	Determining the training and validation loss	23
5.4	Determining the training and validation accuracy	23
5.5	Determining the training and validation loss	25
5.6	Determining the training and validation accuracy	25
5.7	Comparative Analysis between different models	26
8.1	User Interface Inside the App	30
8.2	Our Future Plans of Implementing Federated Learning	32

List of Tables

4.1	Number of Images Per Class	14
-----	--------------------------------------	----

Chapter 1

Introduction

1.1 Motivation

After the world survived pandemics such as the ones caused by Influenza, Cholera, and the Bubonic Plague, humans have gotten better at identifying and treating these life-threatening diseases. As better and more effective medicines and medical techniques were being invented, we felt much more confident in our endeavor to be protected from any further pandemics.

COVID-19 arrived in 2020 against all odds. It was one of the most unexpected phenomena in modern times, unpredicted by anybody. In a very short period, it started spreading uncontrollably, infecting people by the millions. Daily life was in complete disarray, and lockdowns were enforced all over the world in an attempt to contain the spread of the virus. However, as of May 2021, the world is still struggling to find some stability with newer and deadlier strains of the virus being reported every month. We are in complete uncertainty of when the situation will become normal once again, and if we can ever get rid of this deadly virus.

Since this challenge cannot be faced by humans alone, we have been trying to use deep learning algorithms to assist us in the process. One of the implementations of deep learning is using Convolutional Neural Networks (CNN) for identifying COVID-19 infected lungs from X-ray images. Convolution layers, pooling layers, and fully connected layers are fundamental building elements in a convolutional neural network[1]. Several neurons are used to create fully connected neural networks. An artificial neural network is a mathematical model based on the motivation of the biological brain. It comprises certain properties of the human brain, particularly the biological neuron of the brain and how it functions as a link between them. The concept of interconnected neurons is implemented in a neural network to handle specific problems. A neural network is composed of three layers: input, hidden, and output. The data from the input is received by the layer of input neurons, and the processed result is received by the layer of output neurons. Between them are hidden layers that include numerous neurons in diverse patterns. Weights are applied to each neuron and multiplied by the input [2]. For the model with numerous layers, an activation function is used; otherwise, a linear regression model would be constructed, with restricted performance. To decrease error in huge data sets, the notion of backpropagation is employed to move back from output layers to preceding layers [3]. The concept of backpropagation is used to travel back from output layers to preceding levels to reduce errors in large data sets. Following the discovery of

mistakes, the weights may be modified using the gradient descent. The output from the final Pooling or Convolutional Layer is the input of the fully connected layer [1]. It is flattened and then fed into the fully connected layer [1].

In deep learning, the deeper the neural network, the more data training is required to better suit the parameters of each layer. Otherwise, overfitting is quite easy which leads to a poor generalization ability. In response to this issue, transfer learning is a viable option [4]. Transfer learning aims to transfer knowledge from the source domain to the target domain by relaxing the assumption that the training data and the test data must be independent and distributed in the same way. This will have a significant positive impact on a variety of domains that are difficult to enhance due to a lack of training data [5].

1.2 Research Problem

From the early days when the severity of COVID-19 was being realized by the general populace, countless attempts have been made in search of a cure for it. Being a virus, it does not truly have a cure, but rather vaccines can be used to prevent further cases of COVID-19. Therefore, the race to discover the vaccine for COVID-19 was first initiated.

While the research was ongoing to discover the vaccine by studying the inner structure of the virus, it continued spreading to the far reaches of the world. Even developed countries started to notice that they do not possess the necessary equipment and protective measures required to prevent the virus. Even before contracting the virus, humans were unaware of the protective measure which should be implemented to reduce the risk of contracting the disease. An inadequacy of personal protective equipment (PPE), masks, and medicines are some of the major causes of death in both developing and developed countries [6]. The usage of PPE was previously limited to hospitals only, and producers were unable to cope up with the increased demand.

Another problem started to arise when a person suspected that they might have been exposed to the virus and wanted to be certain by going through a test. To do that, one testing kit was required per person, and the inadequacy of raw materials was a serious hindrance in producing to meet the rising demand. Additionally, the global lockdown and travel restrictions also caused bottlenecks in the supply chain, resulting in the factors of production being inefficiently utilized [7].

The final problem started being apparent once mass people started getting infected with the virus. Since a lot of the virus's symptoms were similar to the ones caused by pneumonia doctors had some ideas about the steps to follow to alleviate the patients. What they were not prepared for is the sheer volume of the patients that they were required to diagnose. They bravely rose to face the challenges but treating infected patients had the adverse effect of infecting some of the doctors in return. As time progressed, the discrepancies between the number of available trained surgeons and the number of infected people started to widen [8].

After careful consideration of the previously mentioned problems, our research aims to answer the following question:

“How to efficiently and accurately detect the severity of COVID-19 infected patients with the least human intervention to determine who requires the most urgent treatment?”

Our research will explore how we may use transfer learning to modify previous pneumonia-detection models to now be able to identify COVID-19 in X-ray scans and judge their severity based on CXR scores.

1.3 Research Objectives

This analysis focuses on advancing a COVID-19 detection system to discover infections from the lungs caused by COVID-19 using a Convolutional Neural Network (CNN) on top of X-ray images. At the preprocessing stage, abnormalities in data transmitted by X-ray pictures may be recognized and deleted. The objectives of this research are:

1. To deeply understand CNN, and how it works.
2. To deeply understand image processing techniques.
3. To develop a model for COVID-19 detection based on X-ray images and train with its other data.
4. To evaluate the model.
5. To offer recommendations on improving the model.

Chapter 2

Literature Review

The COVID-19 detection model is changing rapidly due to the CNN models that have the advantage to detect infection patterns in the lungs. CNN is one of the most useful deep neural networks in which several hidden layers execute convolution and sampling to extract modest to considerable amounts of input data. These layers are known as convolutional layers, which contain featured input and output. Here, different channels are intertwined with the information in each layer [9].

2.1 COVID-19 & its Effects on Lungs

COVID-19 is a contagious disease that was initially discovered in Wuhan, China in December of 2019. Attempts were made to contain the disease around the source, however, it started to spread globally around March 2020 and has caused a worldwide pandemic situation causing lockdowns as of the first half of 2021.

COVID-19 is mainly caused by a virus named severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The main symptoms include breathing problems, dizziness, fever, nausea, vomiting, diarrhea, cold, cough, and loss of taste and smell. In some moderate cases, we may notice dry cough, sore throat, inflammation in the trachea and the lungs, and in some people, this may turn into pneumonia [10].

Our COVID-19 detection model will mainly be focusing on X-ray scans to identify the changes which can be observed in a pair of lungs that are affected by COVID-19 compared to a pair of healthy ones. The severity stages of this disease can be identified using a metric known as a chest X-ray (CXR) scoring system. Using this system, we can categorize patients into different stages of infection based on the correlation of their CXR scores with their age, sex, etc [11].

Some distinct visual changes can be noticed in our lungs. These changes are then analyzed and their contrast with healthy lungs can give us an idea of the severity of the infection. In the scan of a pair of normal lungs, we can see the markings inside the lungs. However, in infected lungs, these markings are hidden by some cloudy substance, which is caused by increased lung density [12]. Our model can identify this white pattern and predict the chance of COVID-19 and its severity from it.

2.2 CNN Architecture

CNN is one of the most amazing deep neural networks in which several hidden layers execute convolution and sampling to extract modest to considerable amounts of input data. These layers are known as convolutional layers, which contain featured input and output. Here, different channels are intertwined with the information in each layer [3]. Convolution layers, pooling layers, and fully connected layers are amongst the building components of the CNN architecture. Usually, an architecture contains one or more fully connected layers, followed by a stack of multiple convolution layers and a pooling layer. In our CNN model, there are a total of 384 2D convolutional layers, 64 1D convolutional layers, another set of 32 1D convolutional layers, a flatten layer, a 128 neuron dense layer, a 64 neuron dense layer and a two neuron output layer. This results in a total of 484 layers.

Forward propagation refers to the process of transforming input data into output data using these layers. Pixel values are stored in a two-dimensional (2D) grid, such as an array of numbers, in digital images, and a small grid of parameters called kernel, an enhanced feature extractor, is applied at each image position. This can create CNNs highly systematic for image processing because a feature can emerge anywhere in the image. Although the convolution and pooling operations described in this section are for 2D-CNN, they are also applicable to three-dimensional (3D)-CNN [3].

2.3 Supervised Learning

In neural networks, teaching patterns can be categorized into three sections such as supervised learning, unsupervised learning and reinforcement learning. Hence, supervised learning takes an input vector which is presented with a set of responses at the output layer. To put it simply, it uses labeled input and output datasets to train algorithms which can later classify data or outcomes accurately. However, a forward pass is done and also it calculates the errors between preferred and actual responses for each node in the output layer. Later, the values are used to determine weight changes according to its existing network [3].

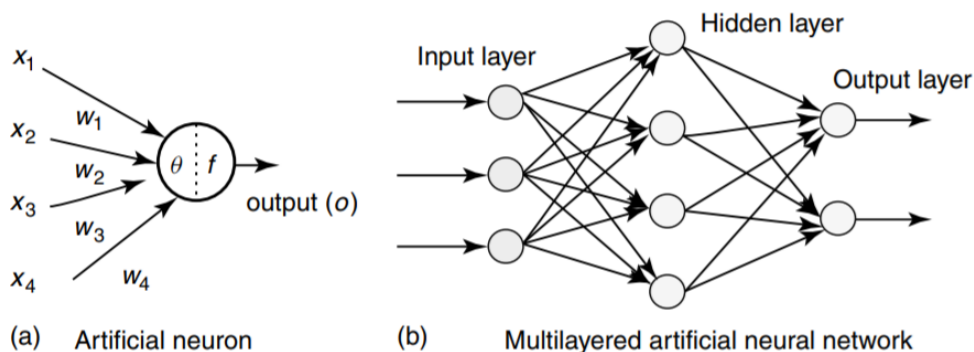


Figure 2.1: Architecture of Artificial neuron and Multilayered Artificial neural network.[3]

2.4 Different Types of CNN Models

2.4.1 Resnet

Resnet is one of the models of the CNN which uses identity concepts to skip the connection by adding a particular input to output directly. It bypasses a few stages of training and joins directly to the output, referred to as skip connections. If we consider X as input, the data travels from layer to layer. Skip connections will not add any extra parameters or complexity to the process.

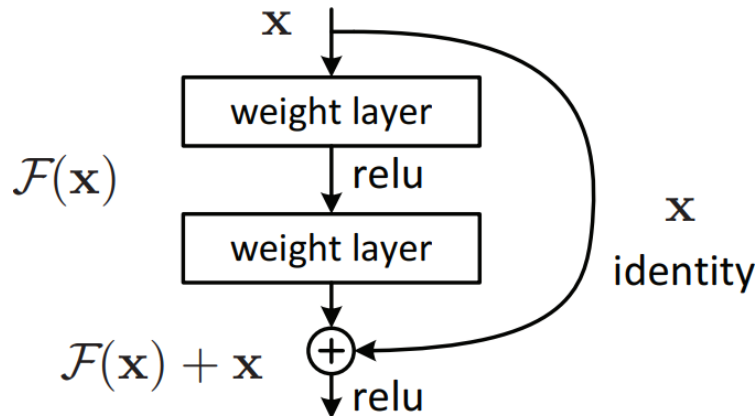


Figure 2.2: Diagram shows how Input X is added to Output

We can consider it as $F(X)$ on the basis of input. In CNN Networks it would be $Y = F(X)$, since X is connected directly with the output we are considering $Y = F(X) + X$. Since we want to get $Y = X$ in the residual network we have to convert $F(X)$ to 0. If the input is not equal to the output a convolution block is added in the skip connection to match it.

There are many variations of Resnet, out of which Resnet-50 was used for the main implementation as it brought better accuracy, precision when compared to other CNN models. If we compare Resnet of 18 layers and 34 layers we can see that Resnet of 34 layers works better than Resnet of 18 layers. This is due to Resnet 34 layers having lesser error during training and has better understanding of the validation data which also helps us to understand better accuracy gains. Due to that we thought to use Resnet-50 as it had more layers and it was seen to provide better accuracy gains.

Resnet-50 has a kernel size of $7 * 7$ and 64 different kernels with a stride of size 2. Here 1 layer is found. In the next layer we can see max pooling of size $3 * 3$ of stride size of 2. In the next convolution there is a $1 * 1, 64$ different kernels, a $3 * 3, 64$ different kernels and $1 * 1, 256$ different kernels. All these layers are repeated thrice, so we get 9 layers. After that, in the next convolution there is a $1 * 1, 64$ different kernels, a $3 * 3, 64$ different kernels and $1 * 1, 256$ different kernels. All these layers are repeated four times, so we get 12 layers. In the next convolution there is a $1 * 1, 64$ different kernels, a $3 * 3, 64$ different kernels and $1 * 1, 256$ different kernels. All these layers are repeated four times, so we get 16 layers. In the next convolution there is a $1 * 1, 64$ different kernels, a $3 * 3, 64$ different kernels and $1 * 1, 256$ different kernels. All these layers are repeated four times, so we get 9 layers. After

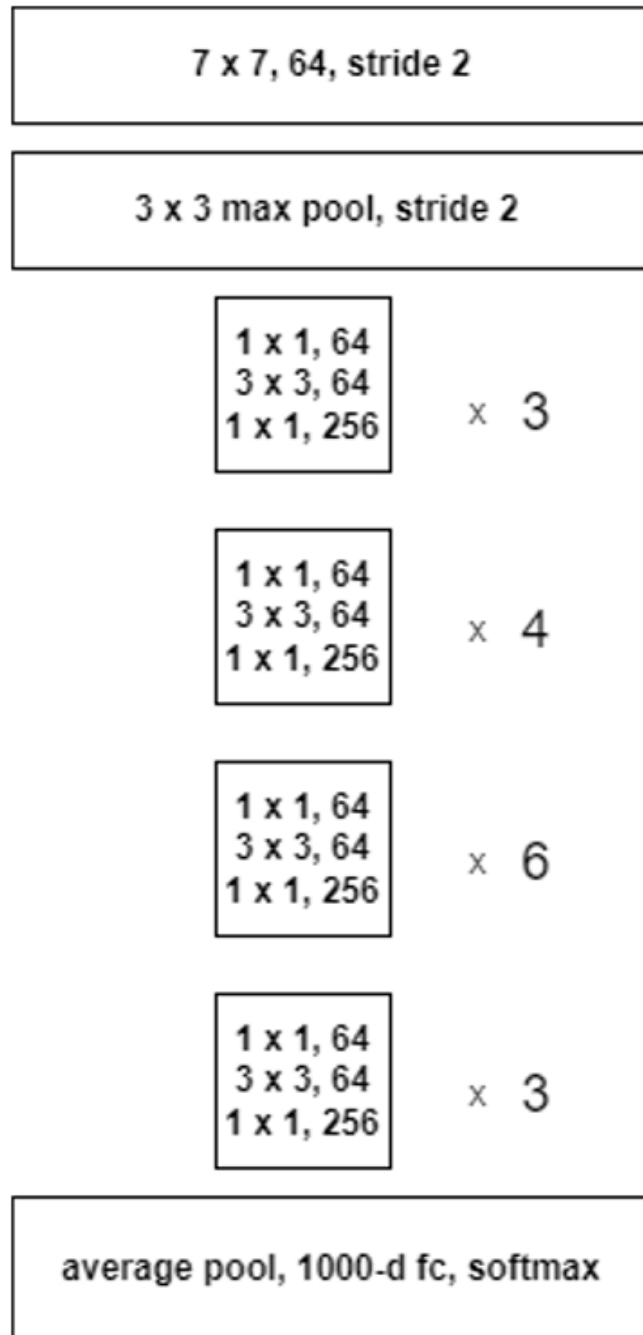


Figure 2.3: Resnet-50 Architecture

that we do an average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us 1 layer. Altogether we get a total of 50 layers.

Many image applications other than image classification, including object identification and face recognition have improved due its powerful representational ability. It also reduces the calculation process of the computer [13].

2.4.2 Inception Net

In our CNN model there can be a lot of problems when the model is big and there are a lot of parameters. If the model is big, the model cannot perform accurately due to overfitting. If there are a lot of parameters in the model, it is put up a huge load while performing computation. So if the model is changed to a thinly dispersed connected manner layer instead of a fully connected layer it would solve the issue. InceptionNet follows that particular architecture.

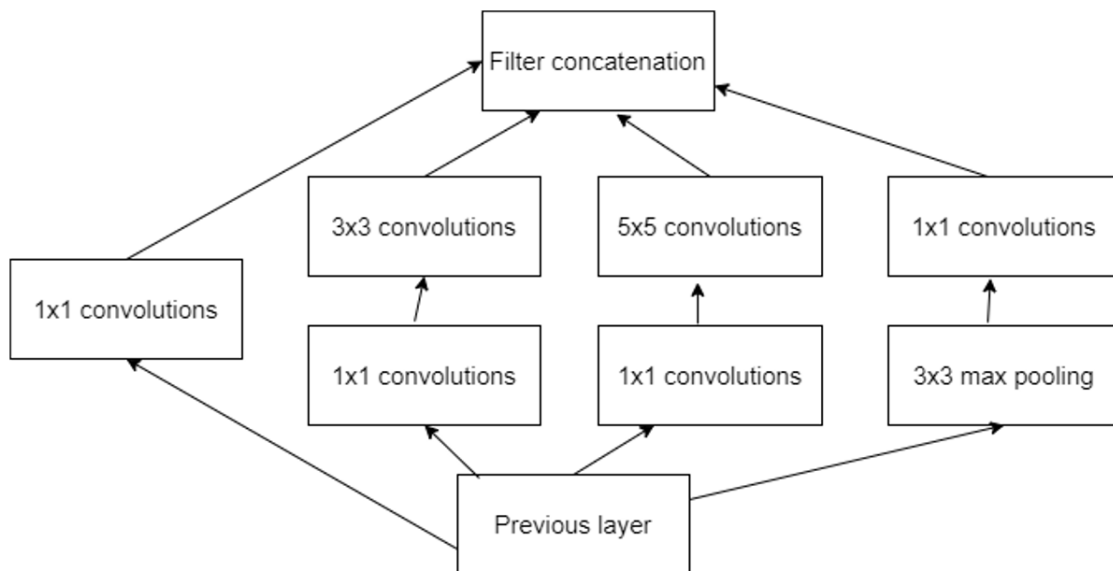


Figure 2.4: Inception Layer

The Inception Layer is made up of three convolutional layers: 1x1 Convolutional, 3x3 Convolutional, and 5x5 Convolutional, with their output filter banks combined into a single output vector that serves as the input to the second process. The 1x1 Convolutional layer is added before being added to another layer for dimensionality reduction, along with the previous convolution layers. The Max Pooling layer is connected in parallel, giving the inception layer another choice.

Out of the four versions of Inception Net we used Inception-v3 for our model which is an extension of the figure of Inception Layer. Inception-v3 is also a 48 layer network . In Inception-v3 we can factorize into smaller convolutions. Next change we can do is to add Factorization into Asymmetric Convolutions which helps to reduce the parameters [14]. For factorization convolutions it explains the reason for the reduction of computation power and decreases the number of parameters. When we decrease the size of the convolutions it trains quicker than earlier. Asymmetric

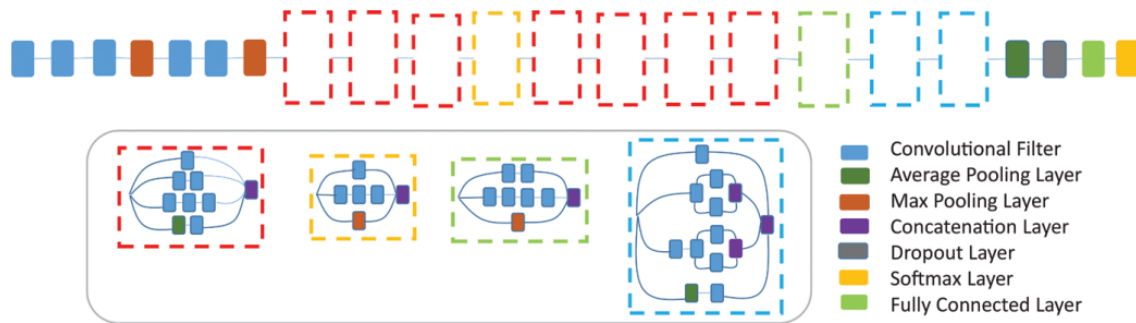


Figure 2.5: Inception Modules[29]

Convolutions is performed by converting 1 two-dimensional kernel into 2 one dimensional convolution kernels. For example, 1×3 and 3×1 can be calculated from 3×3 convolution.

2.5 Related Works

This section of the report is an analysis of the previous work that has been conducted in the field of detecting COVID-19 infected lungs using image data. Since COVID-19 is a relatively new phenomenon, the amount of research conducted specifically on it is quite limited. However, there has been other related research work on identifying similar diseases such as pneumonia from which we can draw inspiration and use as a foundation for comparisons and improvements.

Ali et al. [15] created a deep convolutional neural network (CNN) for the classification of COVID-19 Chest X-ray images into normal and COVID-19 classes using ResNet50, InceptionV3, and Inception-ResNetV2 models. The CT imaging results and the PCR technique had excellent image results, they reported.

Prabira et al. [16] suggested a technique based on deep feature and support vector machines (SVM) for detecting COVID-19 in X-ray images. They gathered images of X-rays from GitHub, Kaggle, and the Open-I repository. Despite the short number of pictures utilized in their analysis, they recovered the deep feature maps of multiple CNN models and concluded that ResNet50 performs better.

Maghdid et al. [17] suggested a basic CNN with only 16 layers to identify COVID-19 using X-ray and CT images and found decent results, however, the dataset utilized was small-scale.

Using the Resnet18 model and image patches focusing on regions of interest, Xiaowei et al. [18] developed an early prediction model to differentiate COVID-19 pneumonia from Influenza-A viral pneumonia and healthy patients using lung CT scans.

Shuai et al. [19] used CT scans to predict COVID-19 cases, with an accuracy of 89.5 percent, a specificity of 88.0 percent, and a sensitivity of 87.0 percent using the Inception transfer-learning model. Several CNN architectures, which are currently utilized for various medical image classifications, were examined across a dataset of X-ray images to differentiate coronavirus patients from pneumonia and normal cases [20]. CNN's were used on a dataset of 224 images of COVID-19, 700 of non- COVID19 pneumonia, and 504 normal where they report overall accuracy of 97.82. Wang and Wong [21] explored COVIDx, a dataset, and COVID-Net, a neural network architecture developed for the identification of COVID-19 cases from open-source chest X-ray radiography images.

In [22], Narin et al. used multiple CNN architectures to categorize normal X-ray images with COVID-19 X-rays and they showed remarkable classification accuracy, sensitivity, and specificity.

The research conducted by our predecessors gives us a working plan in improving on our model, implementing their ideas, and expanding on them to create models with better performance, trained and tested with larger datasets.

Chapter 3

Workplan

The primary objective of our project is to display definitive data on whether a patient is infected by COVID-19 based on their X-ray scans. Initially, we collect various image databases as our input data. After pre-processing those images to filter out unnecessary data we move on to splitting the data for training and testing purposes. We have decided to use 80% of the pre-processed data for training purposes, 10% on validation and the remaining 10% on testing our model for performance metrics. Afterward, we use the training data to create and train the CNN model which is the core of our COVID-19 detection system. After training is complete, move on to the classification stage where we test the remaining input data to identify whether they are infected by COVID-19 or not. If COVID-19 is detected, the model displays the information and administers the patient for further diagnosis.

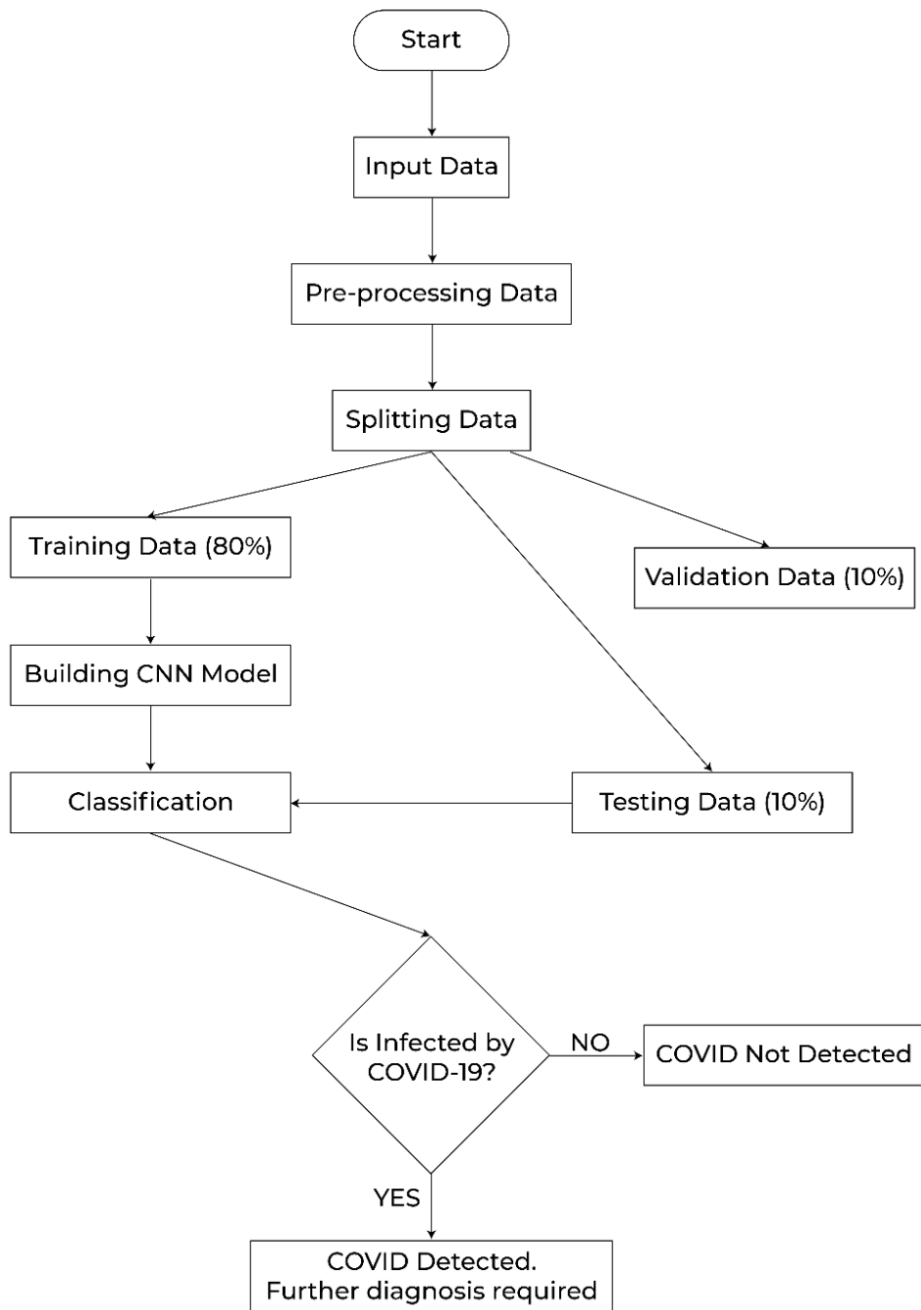


Figure 3.1: The flow chart of the proposed COVID-19 detection model using CNN

Chapter 4

Methodology

Despite the fact that there are a huge number of infected COVID-19 patients across the world, the number of publicly available online chest X-ray pictures are trivial and dispersed. As a result, CNN has been used in this study which uses the generalization capabilities of large datasets to train models, then moves on to smaller chest X-Ray datasets to continue training. Pretrained models are frequently used in this design technique which is why deep evolution neural networks are used to create these pre-trained models. Also, a CNN model can acquire a knowledge of extracting important picture characteristics and select the most phenomenal display features of data for initial training.

4.1 Input data

In this study, a total of 13,808 chest X-ray images have been obtained from different sources which are publicly available datasets, online sources, and published articles [23], [24]. This repository consists of chest X-ray images for COVID-19 positive cases along with negative cases. 10,192 normal (COVID-19 negative) and 3616 COVID-19 positive chest X-ray images were selected for this study.

Here, some sample images from the dataset for COVID-19 positive chest X-ray images has been shown in figure 1.

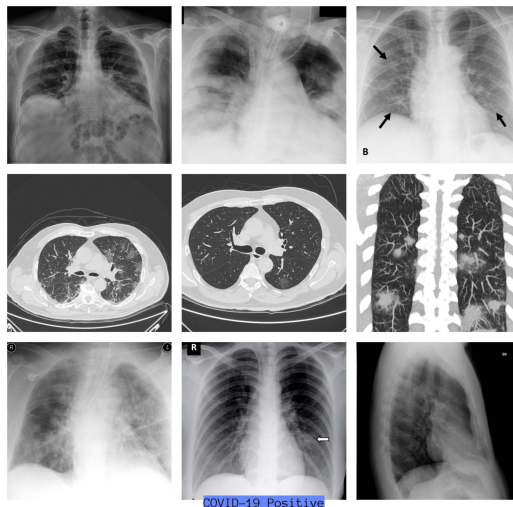


Figure 4.1: Sample COVID-19 Positive X-ray image from the dataset

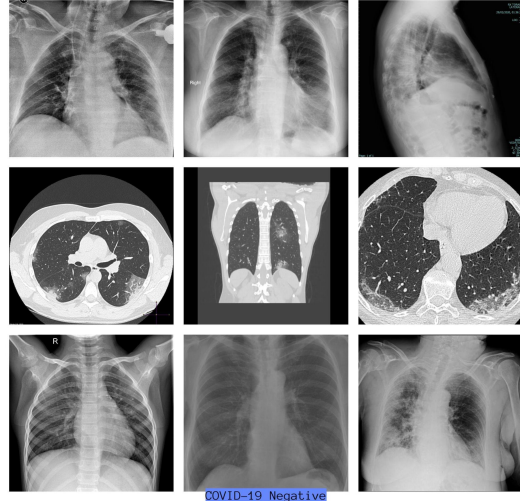


Figure 4.2: Sample COVID-19 Negative X-ray image from the dataset

From the figure 2, we can see the chest X-ray images of COVID-19 negative patients. So, the lung opacity in X-ray images caused by COVID-19 and other lung-related diseases is the primary key to differentiate between COVID and non-COVID categories.

4.2 Data pre-processing

In deep learning, data preprocessing is a critical step that improves data quality and facilitates the extraction of useful insights from the data. For now, we have only used genuine X-Ray data in this study and will not consider the development and use of synthetic data.

Dataset images were labelled into 2 classes which are COVID-19 Negative (0) and COVID-19 Positive (1). A random 80% of the dataset was used for training, 10% was used to test, and the remaining 10% was used to validate the data in the study’s experiments. But, before being used as input to the networks, chest X-ray pictures were resized. The chest X-ray pictures were resized from 299×299 to 100×100 pixels in order to meet the CNN model’s compatibility criteria. Also, the images were transformed to grayscale. The pre-trained model standards were used to normalize all of the images. The table below summarizes the number of images per class used for training, validation, and testing in this study.

Types	Binary classification	Total number of X-ray images	Training set	Testing set	Validation
COVID-19 Positive	1	3616	2892	362	362
Normal	0	10192	8154	1019	1019

Table 4.1: Number of Images Per Class

4.3 CNN Modeling

The CNN model type that has been used in the study is the sequential model. In Keras, the simplest way to build a model is sequential. It allows the model to be built layer by layer. The ‘add()’ function is used to add layers to the model. As it is mentioned before, the first layer of our custom CNN model is the convolutional layer which is mainly used to extract features from input images. It contains a set of filters with height and width which are smaller than input images. Moreover, the CNN model consists of 3 types of convolutional layers, max pooling layers, fully connected layers and additional activation functions. In the convolutional layers, mathematical operations of convolution are performed between the input image and filters of different sizes with strides 1 and no padding. Then, the filter is slid across over the input image, and the dot product is used to compute an output layer of that particular convolution. As the depth starts increasing, inner convolutional layers help in detecting edges from the output of the previous layer. This will generate a feature map to pass through the activation function for learning more about the X-ray images in depth used in the dataset. The activation function used is ReLU or rectified linear unit as it is easier to train and give better performance by cancelling out the negative values. It is the most important parameter as it adds non-linearity to the learning.

Max pooling is the process of extracting windows from the input feature maps and displaying the channel’s maximum value. It’s similar to convolution in general, except that instead of using a trained linear transformation (the convolution kernel), local patches are modified using a hard-coded max tensor operation. Max pooling differs from convolution in that it generally uses 2×2 windows and stride 2 to down-sample the feature maps by a factor of two. Convolution, on the other hand, is usually done with 3×3 windows and no stride (stride 1) [25]. By convolving filters across the convolutional layer, max pooling combines the features of the layer. It helps to avoid overfitting by decreasing the computational cost by limiting the number of parameters. To prevent overfitting and make the model computationally effective, a 2×2 Max pooling layer is added after the convolutional layer in each of the three layers [26]. We can notice that after each MaxPooling2D layer, the size of the feature maps is reduced to half. For example, the feature map is 48×48 before the initial MaxPooling2D layers, but the max-pooling process reduces it to 24×24 . So, max pooling’s objective is to aggressively down-sample feature maps, similar to strided convolutions.

The fully connected layer (FC) or also known as classifier is used to mainly classify the image more accurately. As convolution and pooling already reduced the size of the input images to keep unique characteristics of the classification, we then used flattening to increase dimension of the vector and passed it to FC. FC performs mathematical functions to classify the given model accurately. Here, instead of using sigmoid, we have opted to use “Softmax ” activation function to generate our final yhat values because we are dealing with multiple classification rather than binary. In the terminal dense layer, the expected dimension is 128 which passes through our softmax function to output based on exponents of probability for each classification. However, overfitting can occur while connecting all of the FC layer and there is the issue of relying heavily on one single feature. To solve these problems, dropout (dropout_1, dropout_2, dropout_3) each with value 0.5 that is, half of the neurons se-

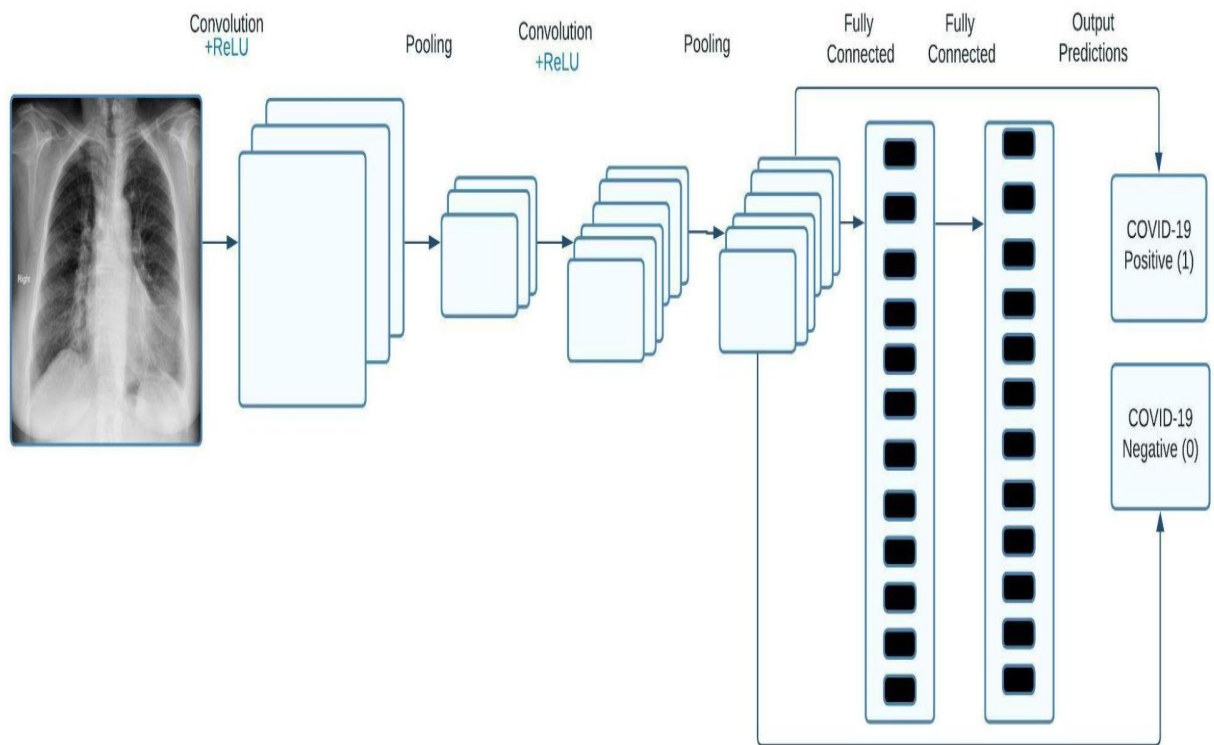


Figure 4.3: Convolution Neural Network

lected randomly will be shut off during training process for further improving our model's performance.

The summary of the model is given below:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
model_1 (Model)	(None, 50, 50, 384)	11008
conv2d_4 (Conv2D)	(None, 48, 48, 64)	221248
activation_1 (Activation)	(None, 48, 48, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 24, 24, 64)	0
conv2d_5 (Conv2D)	(None, 22, 22, 32)	18464
activation_2 (Activation)	(None, 22, 22, 32)	0
max_pooling2d_2 (MaxPooling2D)	(None, 11, 11, 32)	0
flatten_1 (Flatten)	(None, 3872)	0
dropout_1 (Dropout)	(None, 3872)	0
dense_1 (Dense)	(None, 128)	495744
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dropout_3 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 2)	130

Total params: 754,850

Trainable params: 754,850

Non-trainable params: 0

Chapter 5

Implementation & Results

For image classification, CNN is a dynamic model because of its features to categorize image characteristics. This section describes how this suggested model works. At first, the layers are first arranged in a logical order with width, height, and depth dimensions. The neurons in a particular layer do not attach completely with all of the next layer neurons.

However, a Jupyter notebook was used in order to implement and test this model which is implemented in the following stages: pre-processing of input data, classification, normalizing, training, testing and validation.

In this section, the results of implementation of CNN in the given x-ray image dataset is obtained.

5.1 Implementation

Anaconda, the most popular open source Python data science platform, was used to write the program, while Jupyter was used as the Python development environment (Julia, Python and R). Jupyter is a free notebook, open-source, dynamic web application that allows researchers to integrate software code, computational output and other resources in a single document.

Keras, a high-level library for the TensorFlow machine learning framework, was used to construct the suggested technique using the Python programming language which is done in Jupyter notebook. Besides Keras, tensorflow and jupyter, other libraries which we used are Pillow, numpy, scikit-learn, matplotlib, opencv-python and pandas. A Ryzen 3600 CPU and 32 GB RAM powered the machine.

5.1.1 Building the model

After downloading, we can see that the dataset is divided into two folders which are- Covid19 Positive and Covid19 Negative. Among them, 3616 images belonged to label 0 i.e. Covid19 Positive and 10192 images belonged to label 1 i.e. Covid19 Negative. To prepare the pictures for processing, the photographs were kept in the frequently used PNG format and the order of the color channels was altered from the default RGB to GRAYSCALE.

The input data folder is read by the following lines of code:

```
data_path='dataset'  
categories=os.listdir(data_path)
```

```
labels=[i for i in range(len(categories))]
label_dict=dict(zip(categories,labels))
```

Here, the main folder contains two sub folders namely COVID and Normal. These two sub folders are read by the codes given above.

Then the images are resized and the labels are categorized into the dataset using the following code:

```
data.append(resized)
target.append(label_dict[category])
```

This whole sequential model structure includes two convolution 2D layers conv2d_4 and conv2d_5, two maxpooling2D layers max_pooling2d_1 and max_pooling2d_2. In this case, we create a pool size of 2x2 for max pooling. In between each conv2D and maxpooling2D layer, an activation function is used. The activation functions are activation_1 and activation_2 . The next phase is using the flatten function. All of the pooled feature maps are combined into a single vector in this phase. All extracted features are flattened into a single column using the Flatten function. Finally, dense gives the neural network a completely linked layer. Dropout is placed in between flatten and dense phase layers to create a dropout for avoiding data over fitting. To add layers to our model, we use the 'add()' function. It is done by the following lines of code:

```
model = Sequential()
model.add(conv_model)
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(32,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2,input_dim=128,activation='softmax'))
```

For training all of our models, we have used Adam optimizer. It is a deep learning model training algorithm that replaces stochastic gradient descent which is simple to set up, and the default configuration parameters work well for the majority of problems. It uses a first-order gradient-based optimization approach which is based on adaptive lower-order moment estimates. Besides, Adam optimizer requires less memory space, it is easy to implement in the output layer, it works well with large data sets as we use almost 13K images. Most importantly, as it is computationally efficient because it is the combination of gradient descent with momentum algorithm and RMS(Root Mean Square) Prop algorithm and so most models just like our model prefer to use this optimization algorithm in the output layer for compilation. It uses a single learning rate for all of the weight updates of our dataset that do not change throughout the training.

```
model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy',keras.metrics.Precision(), keras.metrics.Recall()])
```

5.2 Results

5.2.1 CNN

In this section, we demonstrate the relevance of our proposed method by providing the experimental results. Here, the ratio of splitting the training-testing-validation dataset is (0.8 : 0.1 : 0.1). In this case, 80% of the dataset is used for training, 10% for testing and 10% for validation. We trained on 11184 samples and validated on 1243 samples of our dataset.

The summary of training our model using CNN is given below:

```
Train on 11184 samples, validate on 1243 samples
Epoch 1/10
11184/11184 [=====] - 877s 78ms/step - loss: 0.4614 - accuracy: 0.7725 - val_loss: 0.3493 - val_accuracy: 0.8399
Epoch 2/10
11184/11184 [=====] - 871s 78ms/step - loss: 0.3130 - accuracy: 0.8670 - val_loss: 0.2421 - val_accuracy: 0.8978
Epoch 3/10
11184/11184 [=====] - 855s 76ms/step - loss: 0.2600 - accuracy: 0.8974 - val_loss: 0.2314 - val_accuracy: 0.9043
Epoch 4/10
11184/11184 [=====] - 829s 74ms/step - loss: 0.2058 - accuracy: 0.9193 - val_loss: 0.2070 - val_accuracy: 0.9091
Epoch 5/10
11184/11184 [=====] - 824s 74ms/step - loss: 0.1713 - accuracy: 0.9345 - val_loss: 0.1768 - val_accuracy: 0.9308
Epoch 6/10
11184/11184 [=====] - 813s 73ms/step - loss: 0.1454 - accuracy: 0.9455 - val_loss: 0.1487 - val_accuracy: 0.9445
Epoch 7/10
11184/11184 [=====] - 808s 72ms/step - loss: 0.1381 - accuracy: 0.9497 - val_loss: 0.1492 - val_accuracy: 0.9372
Epoch 8/10
11184/11184 [=====] - 814s 73ms/step - loss: 0.1120 - accuracy: 0.9596 - val_loss: 0.1494 - val_accuracy: 0.9453
Epoch 9/10
11184/11184 [=====] - 809s 72ms/step - loss: 0.0925 - accuracy: 0.9666 - val_loss: 0.1185 - val_accuracy: 0.9533
Epoch 10/10
11184/11184 [=====] - 808s 72ms/step - loss: 0.0913 - accuracy: 0.9662 - val_loss: 0.1106 - val_accuracy: 0.9646
```

We trained the model with an epoch of 10. After 10 epochs, we got a training loss of 0.0913 or, 9.13% , a validation loss of 0.1106 or, 11.06%, a validation accuracy of 95.87%.

The following figures indicate the above mentioned information:

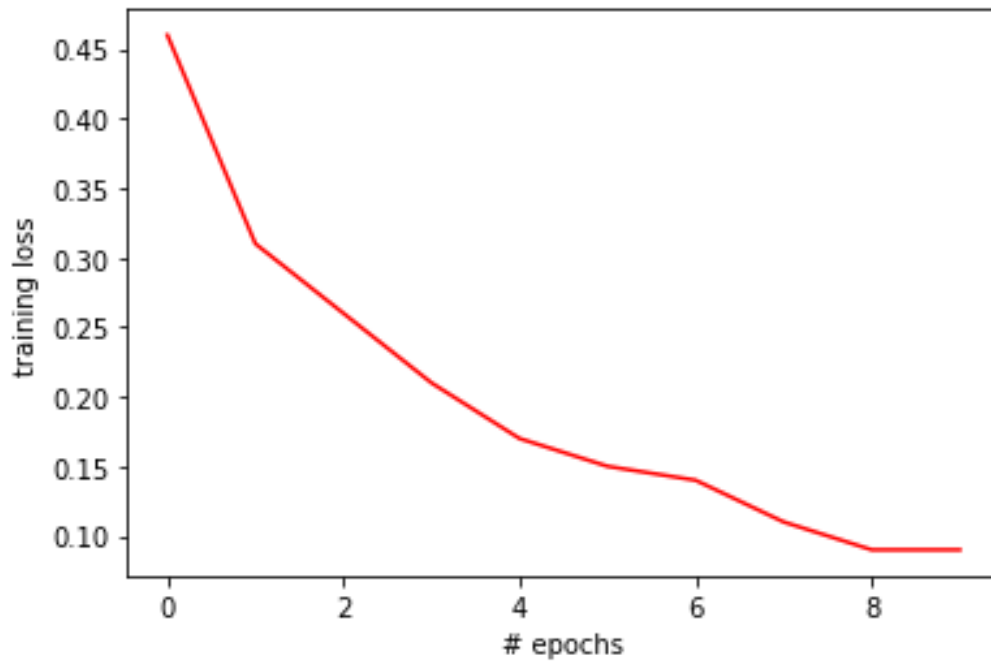


Figure 5.1: Determining the training and validation loss

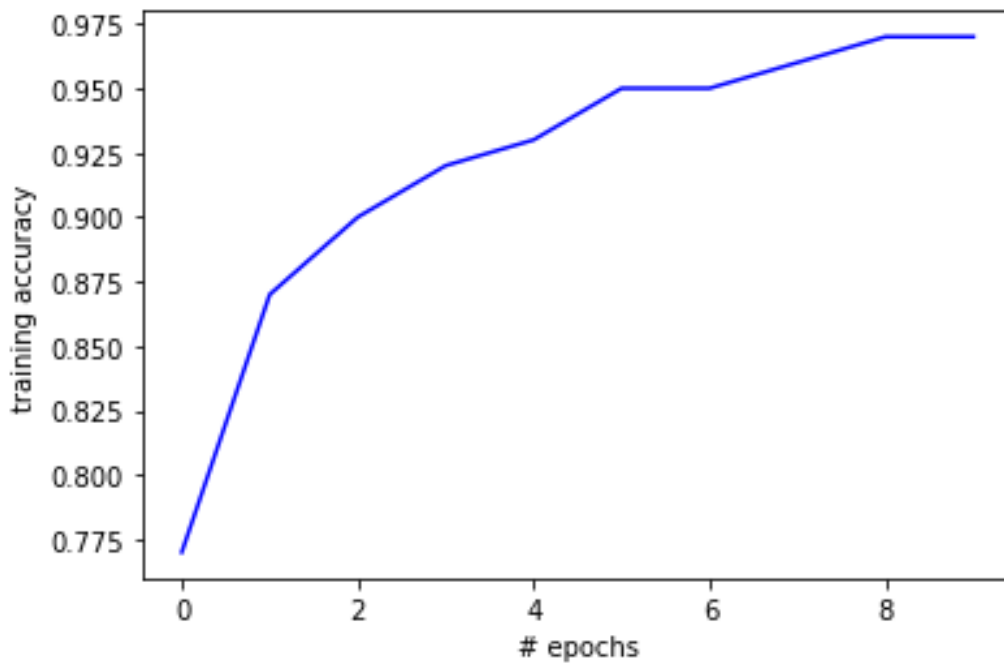


Figure 5.2: Determining the training and validation accuracy

Figure 5(b) depicts the accuracy analysis of the proposed CNN model. It clearly shows that our proposed sequential CNN model achieves a significant accuracy rate. From the results above, we can say that our proposed model is obtaining 96.62% accuracy on our sample dataset based on X-ray images. Therefore, it can be said that our proposed model can efficiently classify the COVID-19 negative and positive patients.

5.2.2 Inception v3

We took the same dataset that was used previously in CNN and used it to train an InceptionV3 model. However, instead of the previous split ratio, we now separated the images in a 0.8 : 0.1 : 0.1 ratio.

The summary of training our model using InceptionV3 is given below:

```

Epoch 1/30
64/64 [=====] - 42s 596ms/step - loss: 103.7736 - accuracy: 0.4653 - precision: 0.5196 - recall: 0.8784
Epoch 2/30
64/64 [=====] - 39s 606ms/step - loss: 16.3270 - accuracy: 0.4626 - precision: 0.5306 - recall: 0.9384
Epoch 3/30
64/64 [=====] - 39s 602ms/step - loss: 2.2461 - accuracy: 0.4878 - precision: 0.5096 - recall: 0.9365
Epoch 4/30
64/64 [=====] - 39s 606ms/step - loss: 0.6197 - accuracy: 0.7642 - precision: 0.5438 - recall: 0.6792
Epoch 5/30
64/64 [=====] - 38s 591ms/step - loss: 0.5154 - accuracy: 0.7646 - precision: 0.5225 - recall: 0.7588
Epoch 6/30
64/64 [=====] - 40s 619ms/step - loss: 0.4494 - accuracy: 0.7646 - precision: 0.4983 - recall: 0.8765
Epoch 7/30
64/64 [=====] - 38s 593ms/step - loss: 0.4805 - accuracy: 0.7242 - precision: 0.4600 - recall: 0.8572
Epoch 8/30
64/64 [=====] - 35s 546ms/step - loss: 0.5083 - accuracy: 0.7549 - precision: 0.4863 - recall: 0.6650
Epoch 9/30
64/64 [=====] - 41s 632ms/step - loss: 0.5644 - accuracy: 0.7314 - precision: 0.4420 - recall: 0.9204
Epoch 10/30
64/64 [=====] - 40s 623ms/step - loss: 0.4260 - accuracy: 0.7330 - precision: 0.4377 - recall: 0.8685
Epoch 11/30
64/64 [=====] - 38s 599ms/step - loss: 0.4408 - accuracy: 0.7256 - precision: 0.4416 - recall: 0.9126
Epoch 12/30
64/64 [=====] - 37s 579ms/step - loss: 0.3953 - accuracy: 0.7627 - precision: 0.4479 - recall: 0.8584
Epoch 13/30
64/64 [=====] - 37s 579ms/step - loss: 0.5070 - accuracy: 0.7545 - precision: 0.4740 - recall: 0.7350
Epoch 14/30
64/64 [=====] - 38s 585ms/step - loss: 0.4367 - accuracy: 0.7046 - precision: 0.4355 - recall: 0.8228
Epoch 15/30
64/64 [=====] - 38s 592ms/step - loss: 0.3944 - accuracy: 0.6685 - precision: 0.4211 - recall: 0.9375
Epoch 16/30
64/64 [=====] - 38s 595ms/step - loss: 0.4378 - accuracy: 0.7158 - precision: 0.4483 - recall: 0.9316
Epoch 17/30
64/64 [=====] - 38s 597ms/step - loss: 0.3543 - accuracy: 0.6895 - precision: 0.4246 - recall: 0.9369
Epoch 18/30
64/64 [=====] - 38s 597ms/step - loss: 0.4116 - accuracy: 0.7427 - precision: 0.4342 - recall: 0.9033
Epoch 19/30
64/64 [=====] - 39s 607ms/step - loss: 0.4444 - accuracy: 0.6289 - precision: 0.3953 - recall: 0.9629
Epoch 20/30
64/64 [=====] - 39s 609ms/step - loss: 0.3964 - accuracy: 0.6650 - precision: 0.4086 - recall: 0.8890
Epoch 21/30
64/64 [=====] - 40s 617ms/step - loss: 0.4003 - accuracy: 0.6685 - precision: 0.4376 - recall: 0.9175
Epoch 22/30
64/64 [=====] - 40s 625ms/step - loss: 0.4010 - accuracy: 0.6689 - precision: 0.4273 - recall: 0.9150
Epoch 23/30
64/64 [=====] - 39s 599ms/step - loss: 0.4695 - accuracy: 0.6294 - precision: 0.4046 - recall: 0.8853
Epoch 24/30
64/64 [=====] - 39s 606ms/step - loss: 0.4065 - accuracy: 0.6685 - precision: 0.4360 - recall: 0.7954
Epoch 25/30
64/64 [=====] - 39s 607ms/step - loss: 0.3929 - accuracy: 0.7144 - precision: 0.4044 - recall: 0.7783
Epoch 26/30
64/64 [=====] - 38s 592ms/step - loss: 0.3738 - accuracy: 0.7120 - precision: 0.4174 - recall: 0.7819
Epoch 27/30
64/64 [=====] - 38s 588ms/step - loss: 0.4321 - accuracy: 0.7531 - precision: 0.4411 - recall: 0.8191
Epoch 28/30
64/64 [=====] - 38s 591ms/step - loss: 0.3897 - accuracy: 0.6323 - precision: 0.3975 - recall: 0.8848
Epoch 29/30
64/64 [=====] - 37s 580ms/step - loss: 0.3745 - accuracy: 0.6264 - precision: 0.3936 - recall: 0.9267
Epoch 30/30
64/64 [=====] - 39s 606ms/step - loss: 0.4121 - accuracy: 0.5112 - precision: 0.3753 - recall: 0.9346

```

During this phase, we increased the number of epochs to 30, hoping that it might yield better results compared to the CNN. Unfortunately, the accuracy score that we received was extremely poor, and the model proved to be quite ineffective in properly classifying the images into the various classes.

The overall trend of the loss and accuracy during training is given in the graphs below:

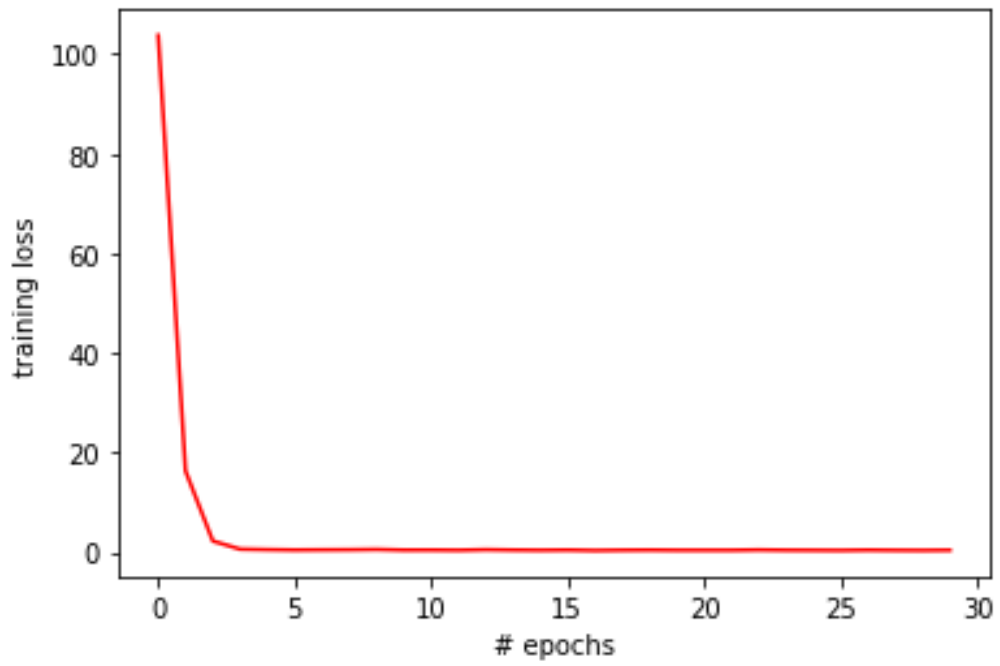


Figure 5.3: Determining the training and validation loss

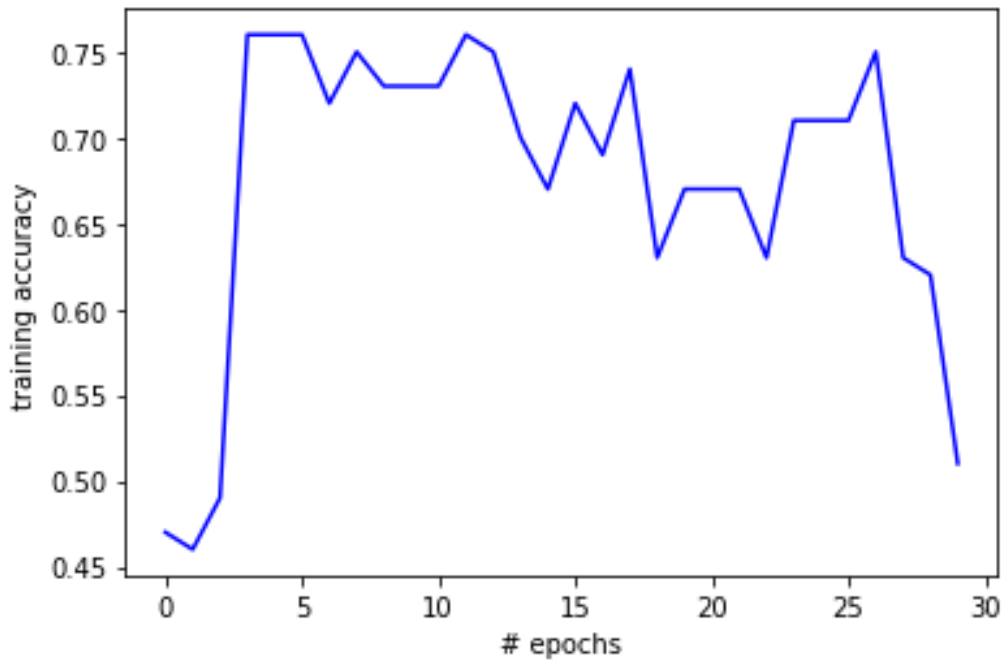


Figure 5.4: Determining the training and validation accuracy

5.2.3 ResNet-50

After InceptionV3's underwhelming performance compared to basic CNN, we decided to employ the power of ResNet50 to help us to properly identify the different categories. We kept the image split ratio the same as InceptionV3 and started the training.

The summary of training our model using ResNet50 is given below:

```
Epoch 1/30
64/64 [=====] - 111s 2s/step - loss: 0.8153 - accuracy: 0.7808 - precision_2: 0.6842 - recall_2: 0.6812
Epoch 2/30
64/64 [=====] - 109s 2s/step - loss: 0.2819 - accuracy: 0.8916 - precision_2: 0.8144 - recall_2: 0.8096
Epoch 3/30
64/64 [=====] - 109s 2s/step - loss: 0.2674 - accuracy: 0.8968 - precision_2: 0.8472 - recall_2: 0.8431
Epoch 4/30
64/64 [=====] - 112s 2s/step - loss: 0.2288 - accuracy: 0.9159 - precision_2: 0.8661 - recall_2: 0.8621
Epoch 5/30
64/64 [=====] - 112s 2s/step - loss: 0.2231 - accuracy: 0.9095 - precision_2: 0.8762 - recall_2: 0.8725
Epoch 6/30
64/64 [=====] - 115s 2s/step - loss: 0.2091 - accuracy: 0.9277 - precision_2: 0.8840 - recall_2: 0.8806
Epoch 7/30
64/64 [=====] - 119s 2s/step - loss: 0.1989 - accuracy: 0.9253 - precision_2: 0.8910 - recall_2: 0.8878
Epoch 8/30
64/64 [=====] - 124s 2s/step - loss: 0.1862 - accuracy: 0.9287 - precision_2: 0.8957 - recall_2: 0.8928
Epoch 9/30
64/64 [=====] - 114s 2s/step - loss: 0.2001 - accuracy: 0.9155 - precision_2: 0.8991 - recall_2: 0.8963
Epoch 10/30
64/64 [=====] - 121s 2s/step - loss: 0.1595 - accuracy: 0.9375 - precision_2: 0.9020 - recall_2: 0.8992
Epoch 11/30
64/64 [=====] - 122s 2s/step - loss: 0.1837 - accuracy: 0.9336 - precision_2: 0.9054 - recall_2: 0.9027
Epoch 12/30
64/64 [=====] - 117s 2s/step - loss: 0.1526 - accuracy: 0.9424 - precision_2: 0.9081 - recall_2: 0.9055
Epoch 13/30
64/64 [=====] - 121s 2s/step - loss: 0.1611 - accuracy: 0.9346 - precision_2: 0.9106 - recall_2: 0.9081
Epoch 14/30
64/64 [=====] - 116s 2s/step - loss: 0.1369 - accuracy: 0.9482 - precision_2: 0.9127 - recall_2: 0.9102
Epoch 15/30
64/64 [=====] - 116s 2s/step - loss: 0.1801 - accuracy: 0.9258 - precision_2: 0.9151 - recall_2: 0.9127
Epoch 16/30
64/64 [=====] - 111s 2s/step - loss: 0.1657 - accuracy: 0.9346 - precision_2: 0.9157 - recall_2: 0.9133
Epoch 17/30
64/64 [=====] - 114s 2s/step - loss: 0.1374 - accuracy: 0.9453 - precision_2: 0.9172 - recall_2: 0.9149
Epoch 18/30
64/64 [=====] - 114s 2s/step - loss: 0.1348 - accuracy: 0.9521 - precision_2: 0.9191 - recall_2: 0.9169
Epoch 19/30
64/64 [=====] - 113s 2s/step - loss: 0.1275 - accuracy: 0.9536 - precision_2: 0.9211 - recall_2: 0.9189
Epoch 20/30
64/64 [=====] - 117s 2s/step - loss: 0.1242 - accuracy: 0.9512 - precision_2: 0.9227 - recall_2: 0.9205
Epoch 21/30
64/64 [=====] - 114s 2s/step - loss: 0.1456 - accuracy: 0.9458 - precision_2: 0.9239 - recall_2: 0.9218
Epoch 22/30
64/64 [=====] - 110s 2s/step - loss: 0.1001 - accuracy: 0.9614 - precision_2: 0.9253 - recall_2: 0.9233
Epoch 23/30
64/64 [=====] - 68s 1s/step - loss: 0.1071 - accuracy: 0.9619 - precision_2: 0.9269 - recall_2: 0.9250
Epoch 24/30
64/64 [=====] - 67s 1s/step - loss: 0.1403 - accuracy: 0.9535 - precision_2: 0.9284 - recall_2: 0.9265
Epoch 25/30
64/64 [=====] - 66s 1s/step - loss: 0.1241 - accuracy: 0.9512 - precision_2: 0.9291 - recall_2: 0.9272
Epoch 26/30
64/64 [=====] - 66s 1s/step - loss: 0.1200 - accuracy: 0.9521 - precision_2: 0.9302 - recall_2: 0.9283
Epoch 27/30
64/64 [=====] - 66s 1s/step - loss: 0.0993 - accuracy: 0.9634 - precision_2: 0.9312 - recall_2: 0.9294
Epoch 28/30
64/64 [=====] - 67s 1s/step - loss: 0.1110 - accuracy: 0.9595 - precision_2: 0.9323 - recall_2: 0.9305
Epoch 29/30
64/64 [=====] - 65s 1s/step - loss: 0.0972 - accuracy: 0.9619 - precision_2: 0.9334 - recall_2: 0.9316
Epoch 30/30
64/64 [=====] - 66s 1s/step - loss: 0.0894 - accuracy: 0.9643 - precision_2: 0.9344 - recall_2: 0.9326
<tensorflow.python.keras.callbacks.History at 0x22f8e76a080>
```

ResNet50 yielded surprisingly accurate results compared to the InceptionV3 model that we used previously. The 96.43% accuracy score even surpassed the basic CNN model's impressive score (95.87%). It was able to correctly identify 30 images that we manually input to test the model's functionality.

The overall trend of the loss and accuracy during training is given in the graphs below:

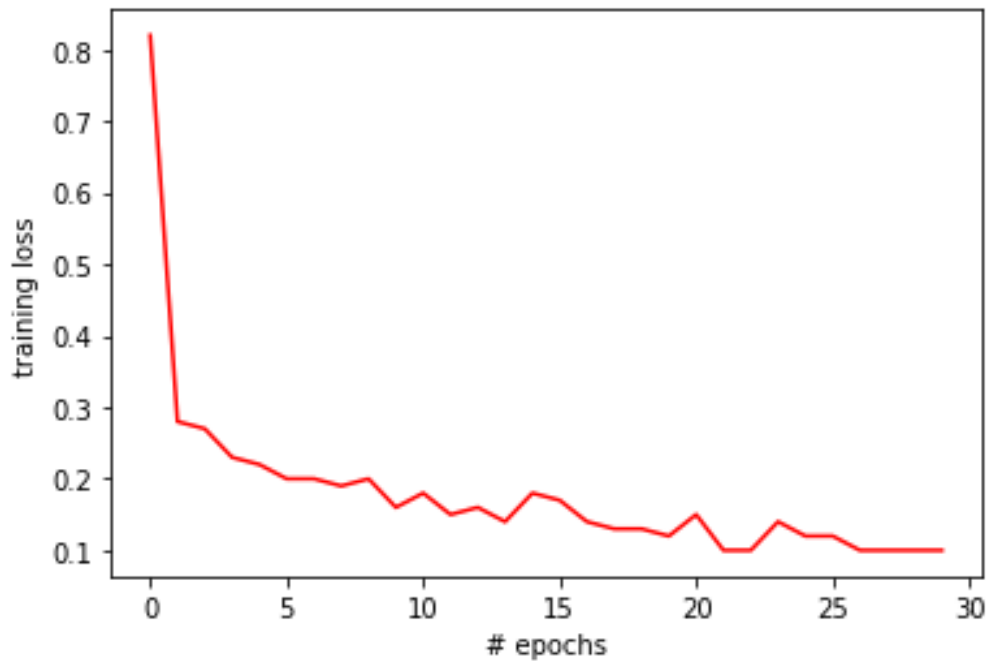


Figure 5.5: Determining the training and validation loss

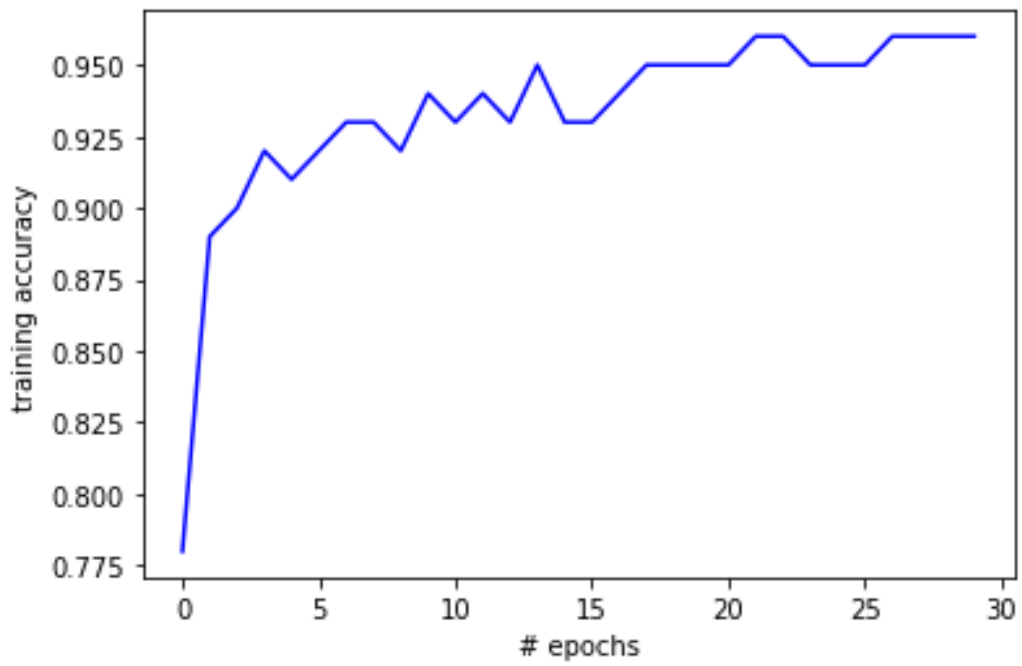


Figure 5.6: Determining the training and validation accuracy

5.2.4 Comparative Analysis of the different models

From the above findings, we are able to notice both differences and similarities between the accuracies of the models. The basic CNN model and ResNet50 had both yielded almost identical results, and they clearly determined which category the image of an X-ray fell under. On the other hand, InceptionV3 seems to be quite ineffective in this particular case, with a subpar accuracy score which cannot be relied upon for such critical medical advice.

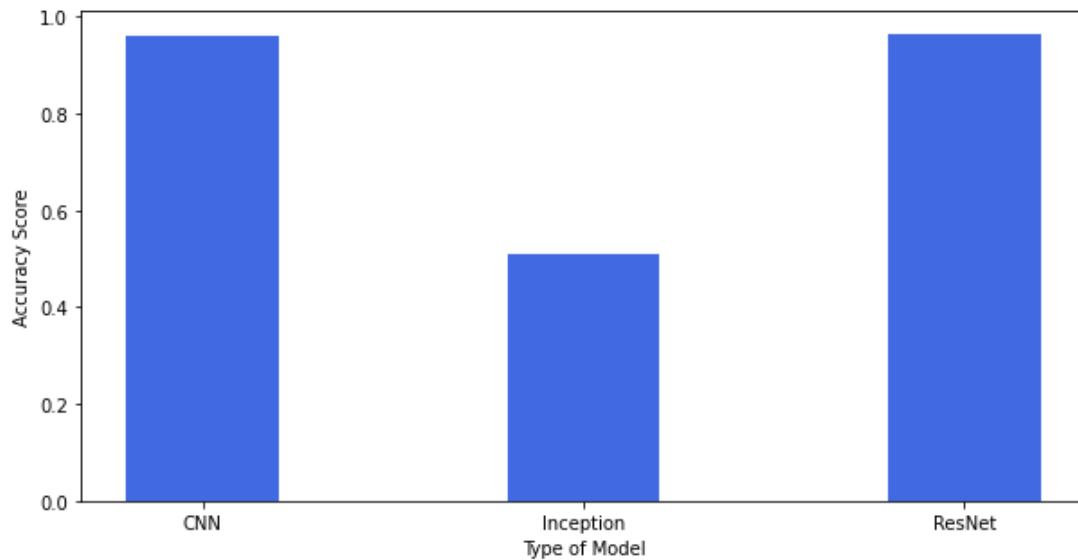


Figure 5.7: Comparative Analysis between different models

Chapter 6

Limitations

For X-ray images, both detection and segmentation are prominent. However, this research has some shortcomings that must be addressed in the near future.

First of all, we could not conduct first-hand research. Also, more COVID-19 samples can be used to train and test our models and resolve the overfitting issue. In recent days, the coronavirus is mutating itself and taking different types of forms which has become a concerning issue all around the world. This can be the cause for our dataset to become obsolete. Furthermore, the current variants of the virus show symptoms similar to Bronchitis which is why our models can be futile since they are more efficient in detecting the virus resembling Pneumonia.

Besides, we have used almost 13,000 images for our model; it would be better if we could use more images to train our model with an increased number of epochs and in that way it would show better outcomes. Moreover, it was quite difficult to label limited images in case of data pre-processing compared to CSV files. In addition, we have faced difficulties to implement our model as different libraries with different versions are incompatible with each other. For instance, we had to use older versions of some of the libraries to be able to run ResNet-50 successfully. Due to the older and incompatible versions of the python library, we could not use GPU. So, we had to use the CPU Ryzen 3600 for training our models. Also, we faced difficulties in converting images from DICOM to JPEG format when we tried to train a different dataset for our model.

Chapter 7

Conclusion

Just like in all other sectors of human lives, the use of technology is now the most viable solution against this global pandemic. Every day newer and better deep learning methodologies are being implemented in our growing arsenal against battling COVID-19. Since this virus is relatively new we have limited research and data concerning it. However, since its symptoms have a similarity to pneumonia, we have the opportunity to use transfer learning methodologies to convert pneumonia-detection models to now be able to detect COVID-19, eliminating the time and resources required to set up a completely new model.

It is possible to state that transfer learning is a deep learning design methodology in which pre-trained models are frequently used. Deep evolution neural networks are used to create these pre-trained models. This method in deep learning includes the initial training of a CNN for a classification issue using large-scale training datasets. Because a CNN model may learn to extract crucial image features, the availability of data for introductory training is a necessary part of successful training. It is determined whether a model is suitable for transfer learning or not based on the CNN's ability to recognize and select the most spectacular display features.

In a research paper, transfer learning is seen to diagnose retinal diseases such as AMD (age-related macular degeneration), diabetic macular edema, etc. Also, with the same framework, this model can distinguish between viral and bacterial pneumonia which ultimately can save a lot of lives [27]. With the development of deep neural networks, it is expected that transfer learning would be widely used to tackle many complex COVID-19 related problems.

The outbreak of COVID-19 started in late 2019 which developed into a global pandemic by March 2020. It has caused a serious negative influence on our daily lives, including the public health system and the global economy. As it is mentioned before, this virus is spreading uncontrollably causing the number of affected patients to grow exponentially. Hence, there is no effective medical equipment or vaccine to eradicate this disease permanently. Also, there is no certainty whether we will find its cure anytime soon. So, in these short periods of time, we need to find more optimum solutions so that we can save as many lives as possible.

Chapter 8

Future Works

The Covid-19 epidemic is spreading rapidly. As of now, we have used two different CNN models in this study to try to characterize Covid-19 afflicted individuals based on their chest X-ray images. Even though the model accuracies are quite good, we propose that the performance be validated using future dataset updates. This may be validated by comparing it to fresh data that will be released in near future. Hence, we may gather more COVID-19 chest X-ray images to validate our proposed model. In further studies, we can implement Data Augmentation techniques which are used to increase the input data set by rotating images from all angles, flipping, cropping, padding and other standard techniques can be used to train the model more efficiently. Also, deeper CNN models might be scrutinized to predict higher accuracy for COVID-19 identification. In addition, the models will be modified so that other lung illnesses such as Pneumonia, Tuberculosis can be detected. These characteristics will be addressed in future research, as well as the creation of a graphics-based user interface to assist radiologists in detecting new types of variants of COVID-19 like Omicron and different imagistic patterns are emerging. For any practical application of our model, it is recommended to consult medical specialists. We want to look into the most cost-effective approaches to battle this disease instead of using flawless detection techniques. Such approaches may be adapted for further study in order to demonstrate the real-world use.

8.1 COVID-19 App

In the very near future, we are hoping to start development of a cross-platform application which allows users to upload image scans of their chest X-rays. These images will be tested in the backend using the model that we developed. After testing, the user will be given a message regarding the probability of them having COVID-19. If there is a high enough probability above a certain threshold, the app will notify the user to perform a COVID-19 test to confirm its result. Additionally, it will also suggest some nearby hospitals where the user may appoint a test directly from the app or allow the user to order a self-testing kit.



Figure 8.1: User Interface Inside the App

8.2 Federated Learning

Since we are already constructing an app to receive user data on an individual level, there is an additional functionality that we may integrate onto our system.

Federated Learning is a relatively new machine learning technique. During this process, the trained model is first stored on a cloud-based server. Devices are able to download this global model from the server and store it on their local storage. They are then able to use local training data, which in our case is their X-ray scan to train the model. Such trained models are combined into a single aggregated model, and finally that model is fed back into the global model to improve its performance. Due to this collaborative technique of training models, Federated Learning has been rising in popularity. Additionally, it solves another problem that was present in machine learning, which is data privacy and security. Since Federated Learning does not send the actual data or image into the global model, but rather the metadata or parameters of the locally trained model, users are able to remain anonymous with no risk of their personal data being stolen. This will undoubtedly encourage more people to participate in the training process, who previously stayed away due to the fear of being compromised.

With the help of Federated Learning, we will be able to enhance our global model with training data from all over the globe. The collaborative methodology will undoubtedly improve our model and yield more concrete results.

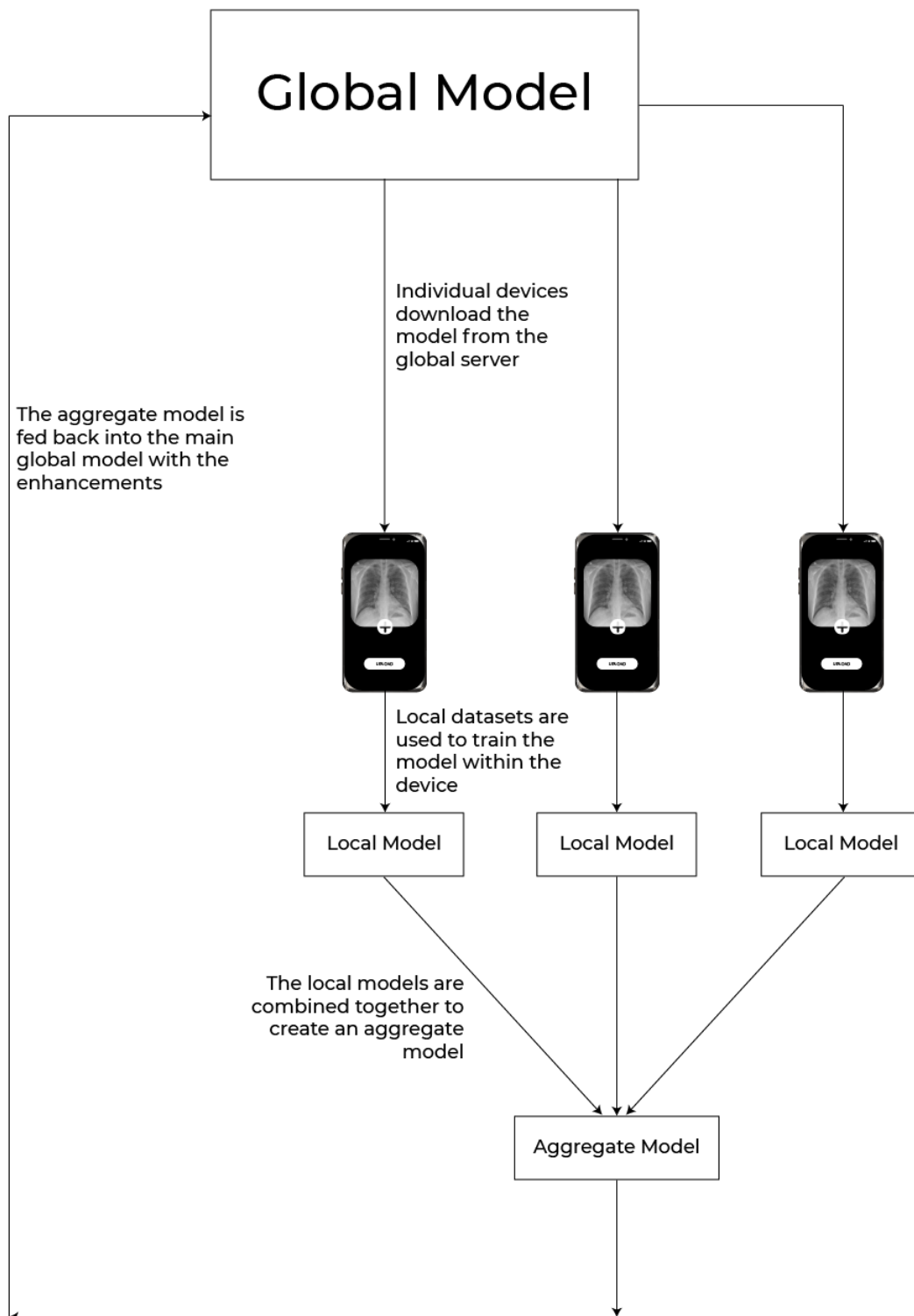


Figure 8.2: Our Future Plans of Implementing Federated Learning

8.3 SMOTE: Synthetic Minority Over-Sampling Technique

In our research, the dataset that we used is quite imbalanced as it did not contain equal numbers of COVID-19 Positive and Negative images. If the classification categories are not equally represented, the dataset is said to be unbalanced. Rather than over-sampling using replacement, we suggest an over-sampling strategy in which the minority class is over-sampled by providing "synthetic" samples which is the 'SMOTE' algorithm. The amount of over-sampling is a parameter of this system. By randomly deleting samples from the majority class population, the majority class is under-sampled until the minority class reaches a quantity approximately equal to the majority class. This will allow us to encounter varied degrees of under-sampling, with the minority class having a better representation in the training set at increasing degrees of under-sampling.

Bibliography

- [1] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology,” *Insights into imaging*, vol. 9, no. 4, pp. 611–629, 2018.
- [2] S. B. Maind, P. Wankar, *et al.*, “Research paper on basic of artificial neural network,” *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 1, pp. 96–100, 2014.
- [3] A. Ajith, *Artificial neural networks: Handbook of measuring system design*, 2005.
- [4] N. Wang, H. Liu, and C. Xu, “Deep learning for the detection of covid-19 using transfer learning and model integration,” in *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, IEEE, 2020, pp. 281–284.
- [5] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *International conference on artificial neural networks*, Springer, 2018, pp. 270–279.
- [6] D. Buonsenso, C. De Rose, and L. Pierantoni, “Doctors’ shortage in adults covid-19 units: A call for pediatricians,” *European Journal of Pediatrics*, pp. 1–4, 2021.
- [7] C.-Y. Park, K. Kim, and S. Roth, *Global shortage of personal protective equipment amid COVID-19: supply chains, bottlenecks, and policy implications*, 130. Asian Development Bank, 2020.
- [8] M. Hoyler, S. R. Finlayson, C. D. McClain, J. G. Meara, and L. Hagander, “Shortage of doctors, shortage of data: A review of the global surgery, obstetrics, and anesthesia workforce literature,” *World journal of surgery*, vol. 38, no. 2, pp. 269–280, 2014.
- [9] V. Madaan, A. Roy, C. Gupta, *et al.*, “Xcovnet: Chest x-ray image classification for covid-19 early detection using convolutional neural networks,” *New Generation Computing*, pp. 1–15, 2021.
- [10] M. Neha Pathak, “What does covid-19 do to your lungs?,” 2021.
- [11] R. Yasin and W. Gouda, “Chest x-ray findings monitoring covid-19 disease course and severity,” *Egyptian Journal of Radiology and Nuclear Medicine*, vol. 51, no. 1, pp. 1–18, 2020.
- [12] J. Cleverley, J. Piper, and M. M. Jones, “The role of chest radiography in confirming covid-19 pneumonia,” *bmj*, vol. 370, 2020.

- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [14] C. Szegedy, W. Liu, Y. Jia, *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
- [15] T. Ai, Z. Yang, H. Hou, *et al.*, “Correlation of chest ct and rt-pcr testing for coronavirus disease 2019 (covid-19) in china: A report of 1014 cases,” *Radiology*, vol. 296, no. 2, E32–E40, 2020.
- [16] P. K. Sethy and S. K. Behera, “Detection of coronavirus disease (covid-19) based on deep features,” 2020.
- [17] H. S. Maghdid, A. T. Asaad, K. Z. Ghafoor, A. S. Sadiq, S. Mirjalili, and M. K. Khan, “Diagnosing covid-19 pneumonia from x-ray and ct images using deep learning and transfer learning algorithms,” in *Multimodal Image Exploitation and Learning 2021*, International Society for Optics and Photonics, vol. 11734, 2021, 117340E.
- [18] X. Xu, X. Jiang, C. Ma, *et al.*, “A deep learning system to screen novel coronavirus disease 2019 pneumonia,” *Engineering*, vol. 6, no. 10, pp. 1122–1129, 2020.
- [19] S. Wang, B. Kang, J. Ma, *et al.*, “A deep learning algorithm using ct images to screen for corona virus disease (covid-19),” *European radiology*, pp. 1–9, 2021.
- [20] I. D. Apostolopoulos and T. A. Mpesiana, “Covid-19: Automatic detection from x-ray images utilizing transfer learning with convolutional neural networks,” *Physical and Engineering Sciences in Medicine*, vol. 43, no. 2, pp. 635–640, 2020.
- [21] L. Wang, Z. Q. Lin, and A. Wong, “Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images,” *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.
- [22] A. Narin, C. Kaya, and Z. Pamuk, “Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks,” *Pattern Analysis and Applications*, pp. 1–14, 2021.
- [23] M. E. Chowdhury, T. Rahman, A. Khandakar, *et al.*, “Can ai help in screening viral and covid-19 pneumonia?” *IEEE Access*, vol. 8, pp. 132 665–132 676, 2020.
- [24] T. Rahman, A. Khandakar, Y. Qiblawey, *et al.*, “Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images,” *Computers in biology and medicine*, vol. 132, p. 104 319, 2021.
- [25] J. Brownlee, *Deep learning for computer vision: image classification, object detection, and face recognition in python*. Machine Learning Mastery, 2019.
- [26] K. F. Haque and A. Abdelgawad, “A deep learning approach to detect covid-19 patients from chest x-ray images,” *AI*, vol. 1, no. 3, pp. 418–435, 2020.
- [27] D. S. Kermany, M. Goldbaum, W. Cai, *et al.*, “Identifying medical diagnoses and treatable diseases by image-based deep learning,” *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.