Covid-19 Detection Using Convolutional Neural Network

A Report on Project submitted
in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
In
Computer Science and Engineering

By

Tanmay Gupta
1903208

Under the guidance of
Akangjungshi Longkumer
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NAGALAND UNIVERSITY

MAY 2023

NAGALAND UNIVERSITY

SCHOOL OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Dimapur – 797112, Nagaland

BONAFIDE CERTIFICATE

Certified that this project titled **"COVID-19 DETECTION USING CONVOLUTIONAL NEURAL NETWORK(CNN)"** is the bonafide work of **Tanmay Gupta (Roll No. 1903208)** who carried out the work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other student.

Place: Dimapur, Nagaland

Date: 26th May 2023

Chenlep Yakha Konyak	**Akangjungshi Longkumer**
In-Charge	**Supervisor**
Assistant Professor	Assistant Professor
Computer Science and Engg.	Computer Science and Engg.
NU-SET, Nagaland	NU-SET, Nagaland

Professor V.K. Vidyarthi

DEAN

NU-SET

ABSTRACT

The COVID-19 pandemic has created a global health crisis, requiring early and accurate virus identification to effectively contain the virus. The goal of this research is to create a COVID-19 detection system utilizing Convolutional Neural Networks (CNN). The objective is to use deep learning and image analysis techniques to identify COVID-19 instances from chest X-ray pictures.

The research entailed gathering a collection of chest X-ray pictures from credible sources, including both positive and negative COVID-19 instances. Preprocessing procedures were used to improve the data's quality and consistency. To extract significant information and categorize the images, a CNN model with numerous convolutional layers, pooling layers, and fully connected layers was created.

Using the obtained dataset, the CNN model was trained using an appropriate loss function and optimization algorithm. The model's performance was assessed using a variety of criteria, including accuracy, precision, recall, and F1 score. The results demonstrated the CNN model's accuracy in recognizing COVID-19 cases.

The benefits and limitations of the CNN-based COVID-19 detection system were determined through thorough analysis and interpretation of the results. The initiative also highlighted model development issues and suggested potential enhancements for future research.

The project's findings highlight the importance of using CNNs in COVID-19 identification. The created model is a viable technique for assisting healthcare professionals in quickly and accurately diagnosing COVID-19 cases, thereby contributing to global efforts to limit the pandemic's impact.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to everyone who helped me finish my project, "COVID-19 Detection Using Convolutional Neural Network (CNN). "This endeavor would not have been feasible without their assistance, direction, and encouragement.

First and foremost, I want to express my gratitude to my supervisor, Akangjungshi Longkumer, for providing invaluable guidance, unshakable support, and expertise throughout this project. Their intelligent recommendations, quick comments, and constructive criticism helped shape and improve my work.

I am also grateful to the faculty members of the School of Engineering and Technology for their ongoing encouragement and support. Their knowledge, skill, and dedication to education have served as a constant source of inspiration for me.

I would like to express my appreciation to my fellow classmates and friends for their help and support throughout this endeavor. Their input, conversations, and support have been crucial in overcoming obstacles and achieving progress.

Finally, I am grateful to my family for their consistent support, understanding, and patience. Their love, encouragement, and faith in my abilities have been the driving forces behind my achievement.

I would like to thank everyone mentioned above, as well as anybody else who has helped with this project in any way, directly or indirectly. Your advice and support have been tremendous, and I am glad for the opportunity to work on this project.

DECLARATION

I, Tanmay Gupta, hereby declare that the project titled "COVID-19 Detection using Convolutional Neural Network (CNN)" is my original work. I have conducted this project under the supervision of Akangjungshi Longkumer at the School of Engineering and Technology.

I affirm that this project is the result of my independent research and that all the information, data, and sources used have been duly acknowledged and cited. Any contributions or assistance received from individuals or organizations in the completion of this project have also been duly acknowledged.

I further declare that this project has not been submitted for any other degree or examination in any other institution. The findings and conclusions presented in this project are based on my own analysis and interpretation of the data, and any opinions or views expressed are solely my own.

I take full responsibility for the accuracy and authenticity of the contents presented in this project. In the event of any discrepancies or errors identified, I am willing to provide the necessary clarifications and corrections.

I understand that any violation of academic integrity or plagiarism is a serious offense and can lead to severe consequences, including the rejection of my project and disciplinary action by the institution.

Date: 26th May 2023

TANMAY GUPTA

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The new Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), which produces the disease known as COVID-19, is a virus that causes respiratory illness in humans and was first discovered in 2019 in China's Wuhan Province. It kept the entire world on edge during the first few months of 2020. It provoked the closing of the borders of many countries and the confinement of millions of citizens to their homes due to infected people. At present (May 2023), it is an ongoing global pandemic that has caused more than 765 million confirmed cases and 6.9 million deaths worldwide, making it one of the deadliest in history. Even though COVID-19 vaccines have been approved and widely distributed all over the world since December 2020, and many more preventive measures have been taken, the effect of the virus is still prevailing significantly around the world, causing havoc. One of the main reasons for this is the mutative nature of the viruses. Even though other symptoms of the disease vary from mutant to mutant to some extent, the most serious illness caused by this virus is related to the lungs, such as pneumonia; thus, these cases can most commonly be diagnosed using the X-ray imaging analysis for the abnormalities.

The test to detect COVID-19 is based on taking samples from the respiratory tract. It is conducted by a health care professional at home, generally when the case study is asymptomatic or symptoms are mild, or in a health center or hospital if the patient is admitted for a serious condition. Conducting as many tests as possible has proven to be the key tool for stopping the virus in countries like Germany or South Korea. Spain was not able to conduct so many tests; therefore, it is important to research and develop alternative methods to perform these tests in a quick and effective way.

The rapid spread of COVID-19 has motivated scientists to quickly develop countermeasures using technologies such as cognitive computing, deep learning, artificial intelligence, machine intelligence, cloud-based collaboration, and wireless communication.

Cognitive computing simulates human thought processes and is extensively used in fields such as finance and investment, healthcare and veterinary medicine, travel, and mobile systems. The Internet of Things (IoT) is a network of networked electronic items that use unique identifiers (UIDs), such as computers, smartphones, coffeemakers, washing machines, and wearable gadgets. The IoT, along with cloud computing, Artificial Intelligence (AI), Machine Learning (ML), and deep learning, could be a powerful tool to combat COVID-19, and 4th generation (4G) and 5th generation (5G) wireless communication technologies have the potential to revolutionize many sectors, including healthcare. China has already used 5G technology to fight the COVID-19 pandemic by monitoring patients, collecting, and analyzing data, and tracking viruses.

Most developing countries utilize wireless technologies, laboratory-based trials, and radiological investigations to recognize and diagnose COVID-19. A standard method is real-time reverse transcription polymerase chain reaction (qRT-PCR), but false-negative results can occur due to asymptomatic patients, and mistakes may also affect its role in identifying COVID-19. In the initial stages, imaging technologies such as CT scans, MRIs, and X-rays might play a vital role in detecting COVID-19 patients.

Radiology-based chest scanning has been employed to investigate pneumonia.

An artificial intelligence (AI)-based tool was developed to automatically detect, quantify, and monitor COVID-19 and to differentiate affected and normal patients. A deep learning-based approach was developed to automatically segment the entire lung for infection sites under chest CT. Similarly, an early screening system based on deep learning can discriminate influenza (viral pneumonia) from vigorous cases and COVID-19. A deep learning-based approach can extract graphical features from CT images of COVID-19. These features provide prior medical analysis and pathogenic testing and have been claimed to save crucial time for disease investigation. However, most people only consider one radiological domain, such as X-ray or CT.

This work builds a deep learning approach to detect COVID-19 from various X-Ray images. The model is a Convolutional Neural Network (CNN) whose 14 layers include input, convolutional, max-pooling, dropout, flatten, dense, and output. The input layer accepts input grayscale images of size 150×150 and uses 32 filters of size 3×3. A ReLU activation function is employed in the input layer and all hidden layers. Following the max-pooling layer is a dropout layer to avoid overfitting. This drops out different neurons in the hidden layer. The percentage of neurons to drop should be specified when using the dropout function. We dropped 20%. Next are three convolutional layers, with 64, 128, and 256 filters, respectively, of size 3×3, a 2×2 max-pooling layer, and a dropout layer to drop 20% of neurons. After that, we flatten the input data into a column matrix. Two dense layers are added to the model with the ReLU activation function, with the first dense layer having 256 units and the second dense layer having 128 units. The dropout layer is added after the dense layer with a dropout of 50%. The final output layer is added with a single unit and a sigmoid activation function. At last, the model's hyperparameters, such as loss function, optimizer, and evaluation metrics, are set using the 'compile' method.

Therefore, the goal of this project was to show how neural networks, specifically CNN classification, can be used to distinguish between a healthy human lung and one that is COVID-19-infected using these X-ray pictures. The proposed model was developed to provide precise binary classification (COVID-19 positive or negative) predictions on the key elements of X-ray pictures. For binary classes, our model generated a normal grouping exactness of 98%.

1.1 WHY USE CNN IN THE PROJECT?

CNN is employed in our project because, compared to other neural networks, it provides excellent accuracy. Additionally, because CNN is primarily used for image classification, it may also be applied to computer vision in a number of other sectors, including medicine. The information required to train a model to recognize and categorize the provided data by self-learning what is what can be provided with the aid of a dataset.

1.2 RELATION BETWEEN CNN, AI, AND DEEP LEARNING

In general, artificial intelligence involves developing a computer to mimic specific human behaviors. Machine learning is a subset of Al that comprises strategies or tactics that allow computers to extract knowledge from a given set of data and pass it along to Al for processing. Deep Learning, on the other hand, is a subset that gives computers the ability to recognize patterns to solve complex problems.

Al leverages deep learning in CNN's sophisticated image processing system to conduct both generative and descriptive tasks known as machine vision, which falls under the category of image and video recognition along with language processing. These are all the procedures and applications used in the medical industry to make better and more accurate decisions and to help people live better lives.

CHAPTER 2

METHODOLOGY

2.1 PROPOSED METHODOLODY

Due to the growing number of issues like COVID-19, pneumonia, and many more in the field of lung illnesses alone, one of the main concerns the healthcare sector has is identifying and providing the proper diagnosis to ill patients at the earliest possible time at a cheap cost. It takes a lot of time to analyze the medical records and determine the patient's accurate diagnosis, which might be quite important.

We have suggested employing deep learning as a strategy to address this type of problem when identifying COVID-19-concerned patients. The time needed for report analysis can be decreased if a Convolutional Neural Network (CNN) like the one shown in Fig. 2.1 is trained to accurately identify the issue from chest X-ray images of ill patients. This will facilitate quicker patient problem identification, enabling early diagnosis and treatment that could potentially save a life.

The goal of the project is to construct an n-layer convolutional neural network that will be trained using the training dataset and get network weights that can be used on the test dataset to analyze the performance of the CNN model and improve its accuracy. Once completed, we should be able to leverage that network of medical applications to perform binary classification to identify an unhealthy lung.

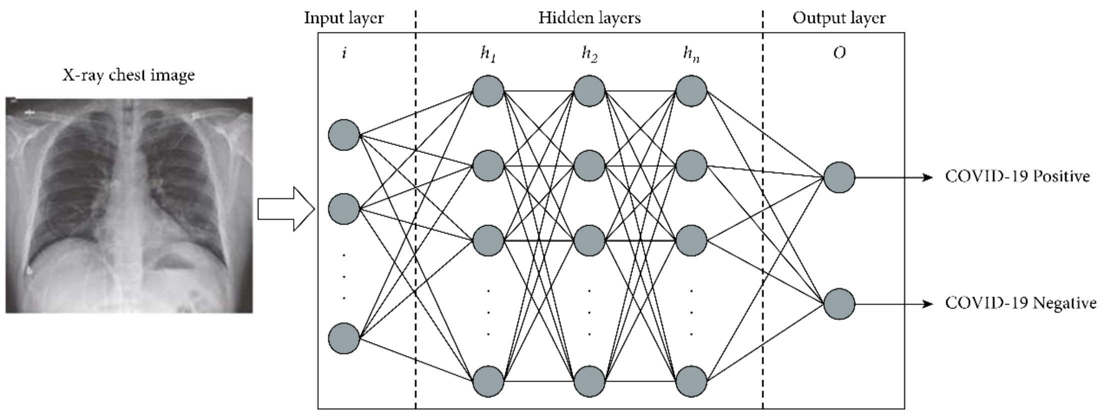


Figure 2.1: Pictorial Representation of proposed approach against X-Ray Images

2.2 NEURAL NETWORK

A neural network is a network of neurons, or in the modern sense, it refers to a network of artificial neurons made up of artificial neuron nodes. It is comparable to the structure of a biological neural network, which consists of a collection of chemically linked neurons with a functioning related neuron. To create a network, a single neuron may connect to a large number of other neurons. An artificial neural network (ANN) models neuron connections as weights between nodes.

2.3 ARTIFICIAL NEURAL NETWORK

Figure 2.2 shows a generic representation of a neural network. If ‘x’ is the input and ‘y’ is the output, then all the additional components in between these two that contribute to determining ‘y’ using ‘x’ are referred to as hidden layers. An artificial neural network is created by combining all these networks.

ANNs are made up of multiple hubs that look like natural neurons in the human cerebrum. The neurons are linked together and interact with one another. The nodes can take input data and execute simple code. The result is passed on to other neurons, and each node's output is activation (or node valves).

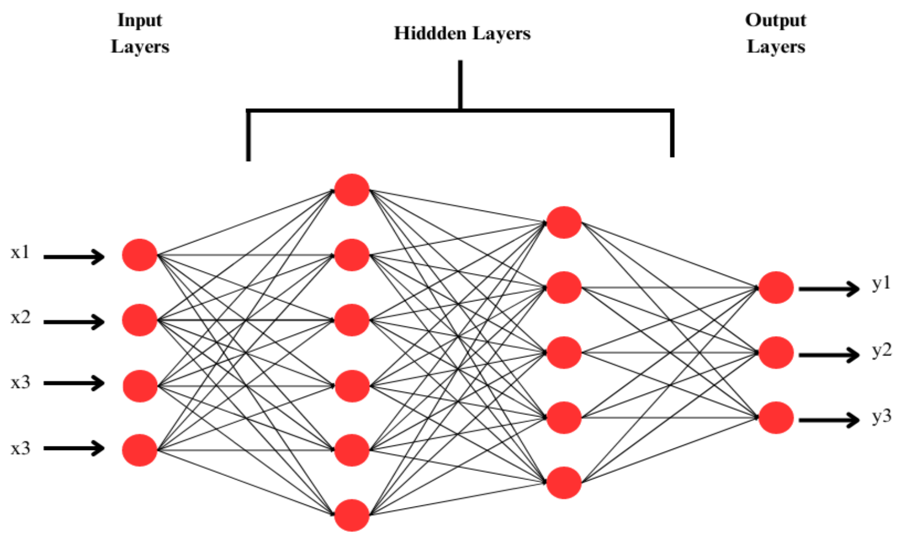


Figure 2.2: General Artificial Neural Network

ANNs are made up of multiple hubs that look like natural neurons in the human cerebrum. The neurons are linked together and interact with one another. The nodes can take input data and execute simple code. The result is passed on to other neurons, and each node's output is activation (or node valves).

2.4 CONVOLUTIONAL NEURAL NETWORK

A CNN is a type of ANN that is used in image processing and recognition and is specifically designed to handle pixel information. CNN is a strong image processing technique that can do both generative and descriptive tasks, and it is frequently used by machine learning to analyze images and videos.

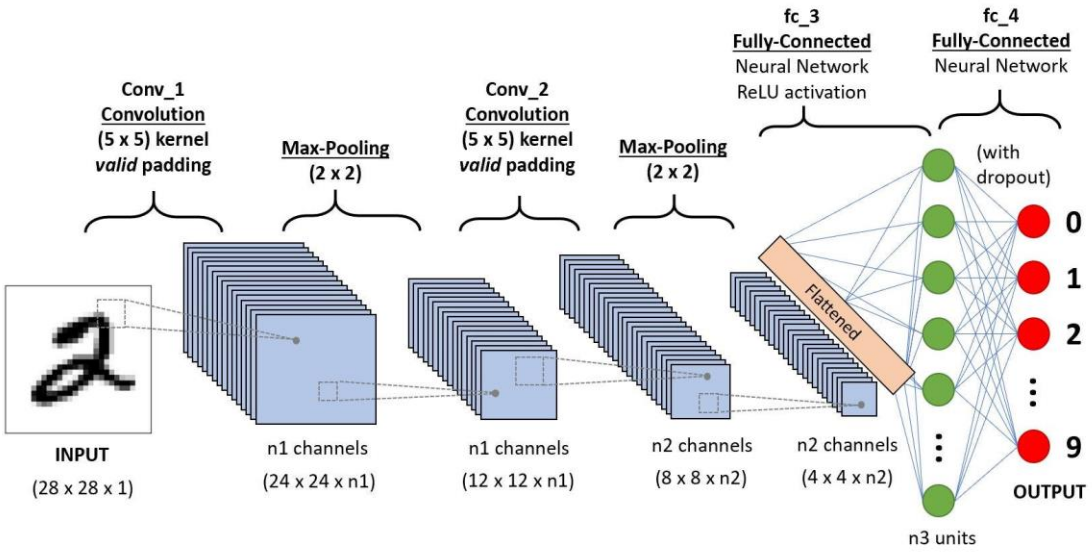


Figure 2.3: General Convolutional Neural Network

In CNN, we first transform the input to lower order using various mathematical methods, then flatten it and feed it to a traditional ANN network for further analysis.

2.5 BLOCK DIAGRAM OF THE PROPOSED SYSTEM

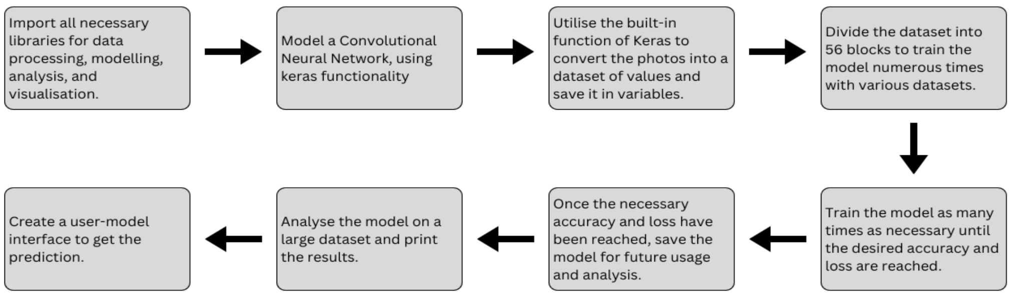

Import all necessary libraries for data processing, modelling, analysis, and visualisation.
Model a Convolutional Neural Network, using keras functionality
Utilise the built-in function of Keras to convert the photos into a dataset of values and save it in variables.
Divide the dataset into 56 blocks to train the model numerous times with various datasets.
Train the model as many times as necessary until the desired accuracy and loss are reached.
Once the necessary accuracy and loss have been reached, save the model for future usage and analysis.
Analyse the model on a large dataset and print the results.
Create a user-model interface to get the prediction.

Figure 2.4: Block Diagram of the Proposed System

2.6 DATA GATHERING AND SEGMENTATION

Data collection is one of the most crucial components of developing a model because the quality and quantity of the dataset have a significant impact on the model's output. A balanced dataset is required to work on the presentation of the suggested CNN models for COVID-19 detection. We collected a total of 562 lung X-ray images of two types: 281 COVID-19 positive lung X-ray images and 281 normal lung X-ray images. These 562 photos are separated into two groups: training images (242 positive, 242 normal) and test images (39 positive, 39 normal). These X-ray images were downloaded from Kaggle. Data segmentation is the process of splitting the existing dataset into groups so that it may be used to train the model with distinct datasets for each epoch.

Table 2.1: Data Distribution Table

	COVID-19 Positive X-Ray Image	**Normal X-Ray Image**
Training Dataset	242	242
Test Dataset	39	39

2.7 PROCESSING THE DATA

Image-to-data conversion, convolutional (Conv2D), max-pooling, flatten layers, the ReLU activation function, the sigmoid activation function, and the Adam optimizer will all be covered here.

2.7.1 IMAGE TO DATA CONVERSION

Figure 2.5 depicts the process of transforming a picture (RGB image) into a matrix of shape (width, height, depth) dataset with pixel values ranging from 0 to 255.

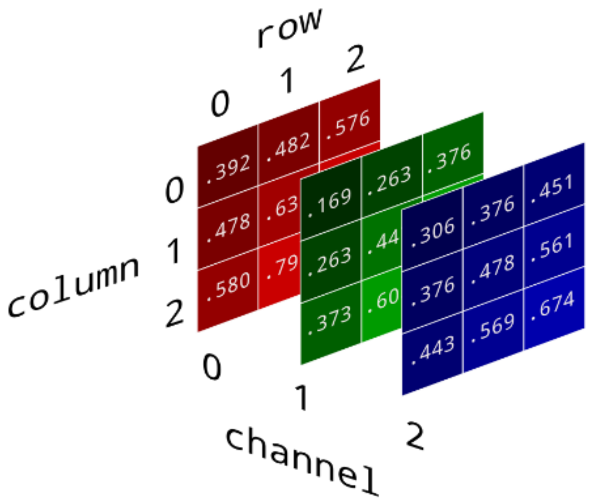


Figure 2.5: Image Conversion from RGB to RGB Matrix

2.7.2 CONVOLUTIONAL (CONV2D)

Convolutional layers are used to apply filters to the source image or to other feature maps, as seen in Figure 2.6. Reusing the same filter on the same piece of data produces a feature map, which shows the area and strength of an identified element in the input, much like a picture.

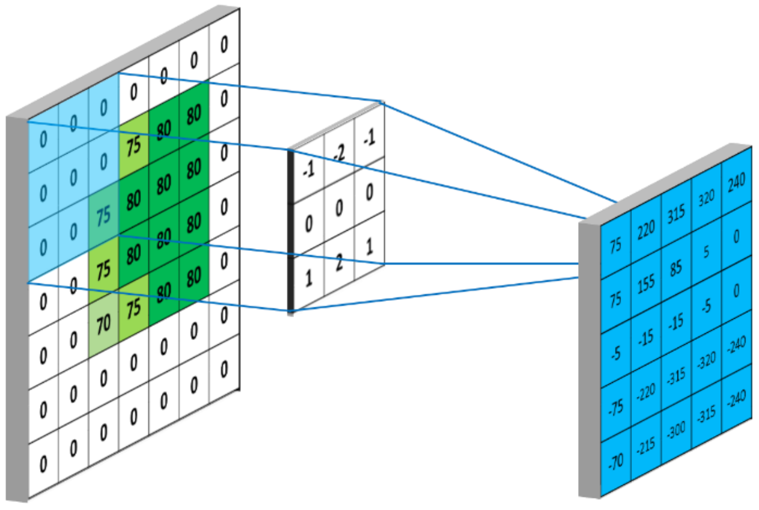


Figure 2.6: Convolutional (Conv2D)

2.7.3 MAX-POOLING

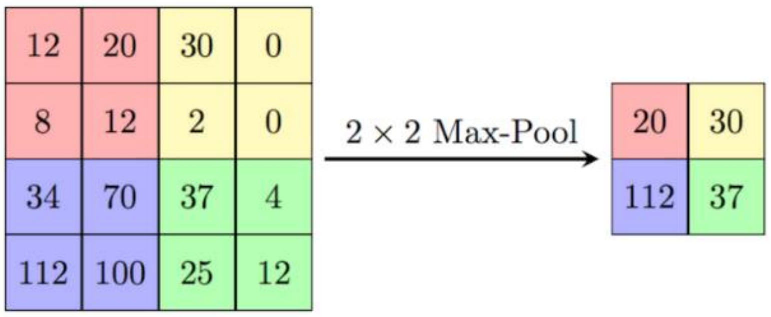


Figure 2.7: Max-Pooling

Max pooling is a pooling action that selects the largest component from the feature map area covered by the channel. As a result of the maximum pooling layer, the final output would be a feature map including the most noticeable parts of the previous feature map. Figure 2.7 depicts one such case.

2.7.4 FLATTEN LAYER

The process of transforming a matrix of order M*N into (M, N*1), that is, a column matrix, is known as flattening layer.

Example: $$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \longrightarrow \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{bmatrix}$$

2.7.5 RELU ACTIVATION FUNCTION

The rectified linear activation function, abbreviated as ReLU, is a linear function that returns zero if the input is less than zero and the same value if the input is larger than zero. Figure 2.8 depicts the ReLU graph.

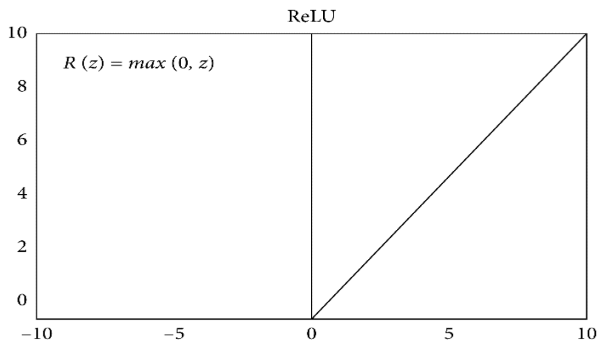


Figure 2.8: ReLU Activation Function

2.7.6 SIGMOID ACTIVATION FUNCTION

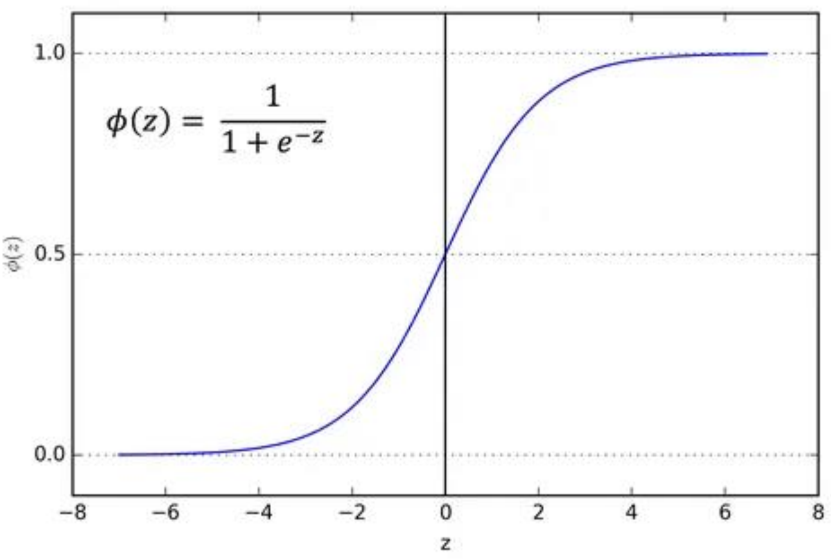


Figure 2.9: Sigmoid Activation Function

The sigmoid function is used because it exists between 0 and 1. As a result, it is particularly useful for neural networks, where we must anticipate the probability as a binary output. Because the probability of anything occurring only between 0 and 1, the sigmoid is the best choice. Because our output is a yes-or-no binary classification at the last step, we make it a sigmoid function in the CNN's final layer. Figure 2.9 shows the graph and function of the sigmoid activation function.

2.7.7 ADAM OPTIMIZER

The Adam optimizer is a hybrid of two slope-plunge enhancements. This approach is used to accelerate the gradient descent calculation by taking the exponentially weighted average into account. Using midpoints causes the calculation to merge faster towards the minima.

$$For\ each\ Parameter\ w^j$$

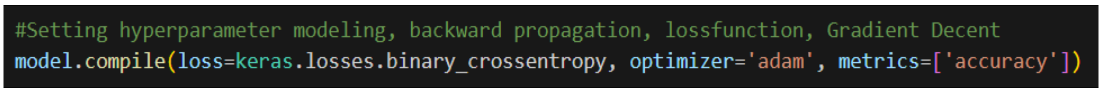
$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

$\eta : Initial\ Learning\ rate$
$g_t : Gradient\ at\ time\ t\ along\ \omega^j$
$\nu_t : Exponential\ Average\ of\ gradients\ along\ \omega_j$
$s_t : Exponential\ Average\ of\ squares\ of\ gradients\ along\ \omega_j$
$\beta_1, \beta_2 : Hyperparameters$

Figure 2.10: Adam Optimizer

2.8 PARAMETER SETUP

Several parameters must be set up before training CNN, such as input size, which is set to (150, 150, 3), implying that any image input will be resized to this configuration; initial, exponential average, learning rate of gradients, exponential average of squares of gradients; and hyperparameters, which are auto configured by the Adam optimizer in the program.



```
#Setting hyperparameter modeling, backward propagation, lossfunction, Gradient Decent
model.compile(loss=keras.losses.binary_crossentropy, optimizer='adam', metrics=['accuracy'])
```

Figure 2.11: Code for Parameter Setup

2.9 MODEL ANALYSIS

Model analysis is the process of examining the accuracy and dependence of the model using the test dataset to gain a better understanding of the training model and visualizing the output model.

2.10 CREATING AN INTERFACE

Creating a method where the user gives his image to get the prediction, whether it is COVID-19 positive or negative, and the prediction is displayed on the screen along with the image.

2.11 PROPOSED CNN MODEL

Modelling CNN is the process of developing a model of a convolutional neural network by specifying the flow of the model as depicted in Figure 2.12 and the required parameters to train the model.

Our CNN model has 16 layers, including four convolutional layers, four Max Pooling layers, four dropout layers for regularization, three fully connected layers, and one output layer. Our CNN model's input shape is (150, 150, 3). The kernel size for all convolutional layers is 3*3; the number of filters used in each convolutional layer is 32, 64, 128, or 256. The Max Pooling layer with a 2*2 size has been used after each con2D layer. The ReLU function is used in the activation layer, while a 20% dropout rate is used in the dropout layer. The result of 65536 neurons in the last Con2D layer is flattened into a column matrix; this flattened layer will be the input for the next 3 dense layers of order 256, 128, and 1, respectively.

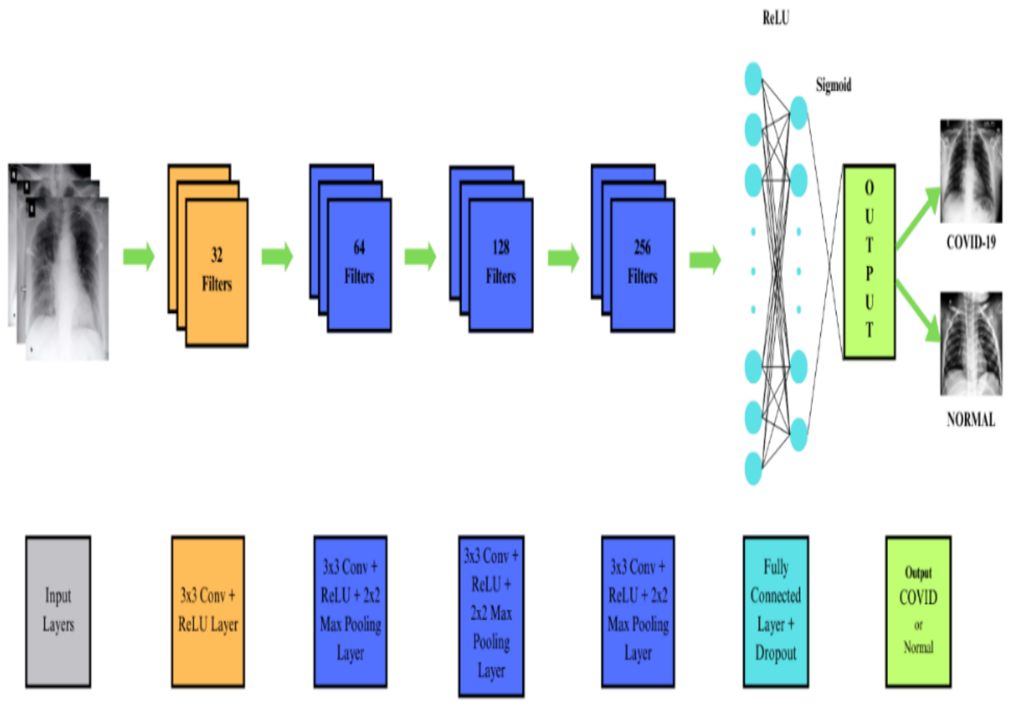

ReLU
Sigmoid
32 Filters
64 Filters
128 Filters
256 Filters
OUTPUT
COVID-19
NORMAL
Input Layers
3x3 Conv + ReLU Layer
3x3 Conv + ReLU + 2x2 Max Pooling Layer
3x3 Conv + ReLU + 2x2 Max Pooling Layer
3x3 Conv + ReLU + 2x2 Max Pooling Layer
Fully Connected Layer + Dropout
Output COVID or Normal

Figure 2.12: Proposed CNN Model

Since only one output node is needed to classify the data into one of two classes, we make use of binary classification by giving the output to a sigmoid activation function. The output given by the sigmoid activation function lies between 0 and 1.

Table 2.2: Summary of the Model

Model: sequential		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
conv2d_1 (Conv2D)	(None, 146, 146, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 73, 73, 64)	0
dropout (Dropout)	(None, 73, 73, 64)	0
conv2d_2 (Conv2D)	(None, 71, 71, 128)	73856
max_pooling2d_1 (MaxPooling 2D)	(None, 35, 35, 128)	0
dropout_1 (Dropout)	(None, 35, 35, 128)	0
conv2d_3 (Conv2D)	(None, 33, 33, 256)	295168
max_pooling2d_2 (MaxPooling 2D)	(None, 16, 16, 256)	0
dropout_2 (Dropout)	(None, 16, 16, 256)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 256)	16777472
dense_1 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
dense_2 (dense)	(None, 1)	129
Total params: 17,198,913 Trainable params: 17,198,913 Non-trainable params: 0		

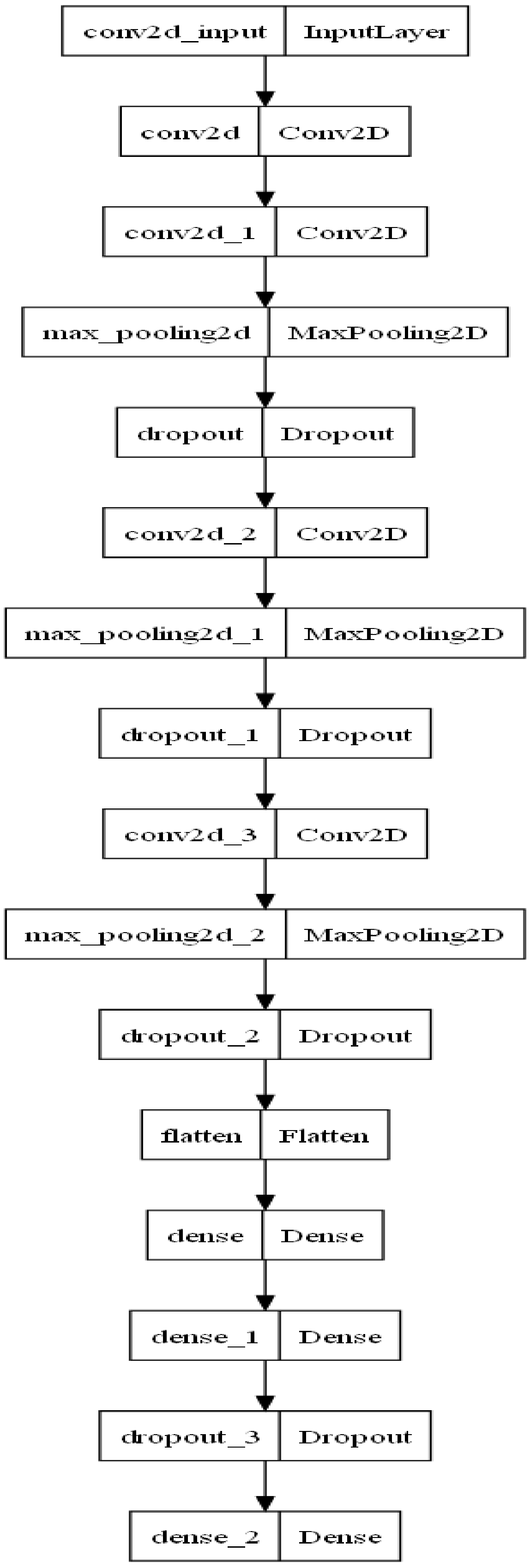


Figure 2.13: Flowchart of the CNN Model

2.12 HAREWARE USED

OS Version: Windows 10

Processor: 9th Gen Intel(R) Core (TM) i7-9750H CPU @ 2.60Ghz

GPU: NVIDIA GeForce GTX 1660 Ti

SSD: 1TB

RAM: 16GB

2.13 SOFTWARE USED

IDE Used: Visual Studio Code

Programming Language Used: Python3

CHAPTER 3

EXPERIMENT (IMPLEMENTATION)

In this chapter, we will talk about the implementation of our project. It entails the creation and training of a convolutional neural network (CNN). To extract features from the input photos, the code used the Sequential model architecture and numerous layers of convolution, pooling, and dropout. For binary classification, the model was built with an appropriate loss function and optimizer.

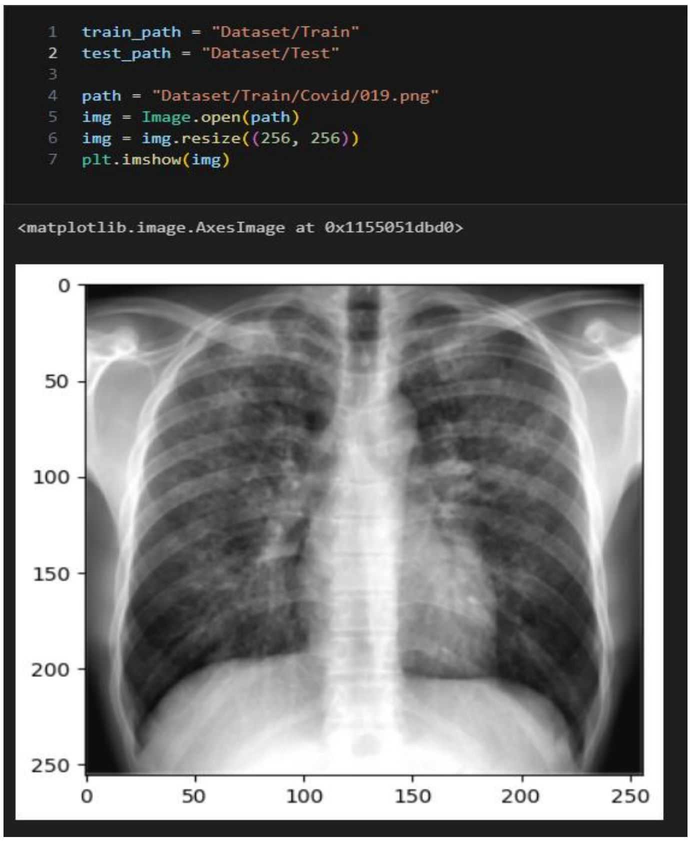
Image data generators were used to preprocess the photos by rescaling and performing data augmentation techniques such as shear, zoom, and horizontal flip. The model was trained on the training dataset for a set number of epochs before being evaluated on the testing dataset. Accuracy and loss plots were used to monitor and visualize the training progress.

The trained model was retained for later usage, and the evaluation outcomes were examined using metrics such as accuracy, loss, and a confusion matrix. A sample image was also sent through the model for prediction, demonstrating how the trained model may be used to identify fresh photos.

This code implementation demonstrates a practical application of deep learning in the medical arena, where CNN models can help detect Covid-19 early in X-ray pictures. Researchers and practitioners can obtain insights into the main steps involved in constructing and training an effective. This paper is a helpful resource for understanding the technical elements of COVID-19 identification and lays the groundwork for future research in this field.

3.1 DATASETS

We obtained the dataset from Kaggle and incorporated it into our project. Figure 3.1 depicts the loading and visualization of an image from a dataset. The paths to the training and testing datasets are defined first. Then, a specific image file from the training dataset is chosen by specifying its path. The code opens the image and resizes it to 256 × 256 pixels using the Python Imaging Library (PIL). The resized image is then presented using the Matplotlib package. This code snippet can help you understand how photos are loaded and preprocessed in machine learning jobs.



```python
train_path = "Dataset/Train"
test_path = "Dataset/Test"

path = "Dataset/Train/Covid/019.png"
img = Image.open(path)
img = img.resize((256, 256))
plt.imshow(img)
```

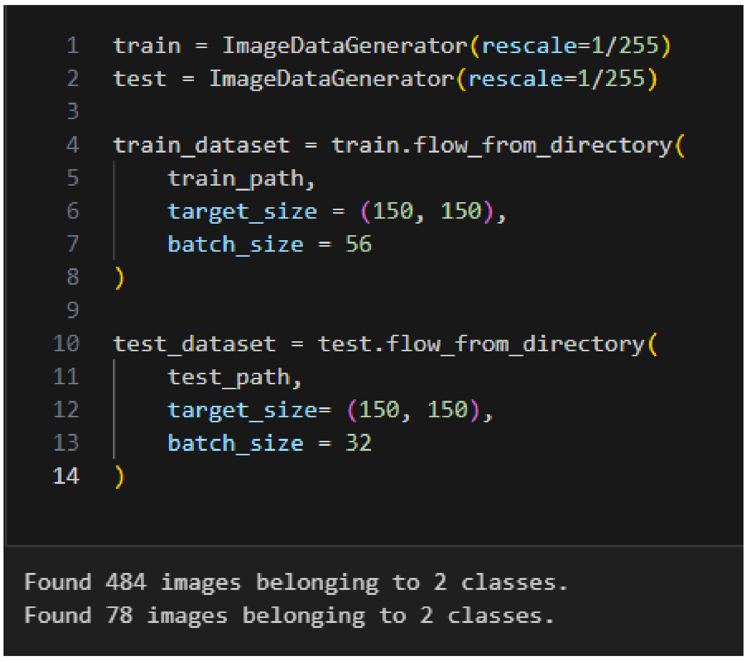
```
<matplotlib.image.AxesImage at 0x1155051dbd0>
```

Figure 3.1: Path Setup and Displaying an Image from Dataset

3.2 IMAGE DATA PREPROCESSING AND DATA GENERATION

Figures 3.2 demonstrate how we utilize Keras' **ImageDataGenerator** class to preprocess and enrich picture data. It generates two data generators, train and test, which rescale the picture pixel values to the range [0, 1].

The **flow_from_directory** technique is then used to generate a training dataset generator (train dataset) and a test dataset generator (test dataset). These generators retrieve photographs from given directories, resize them to 150x150 pixels, and group them into batches. The batch size for the training dataset generator is 56, whereas the batch size for the test dataset generator is 32.



```
train = ImageDataGenerator(rescale=1/255)
test = ImageDataGenerator(rescale=1/255)

train_dataset = train.flow_from_directory(
    train_path,
    target_size = (150, 150),
    batch_size = 56
)

test_dataset = test.flow_from_directory(
    test_path,
    target_size= (150, 150),
    batch_size = 32
)
```

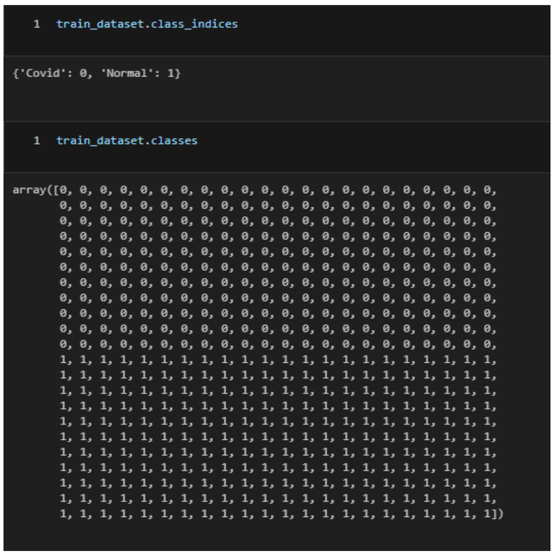
```
Found 484 images belonging to 2 classes.
Found 78 images belonging to 2 classes.
```

Figure 3.2: Data Preprocessing and Data Generation

3.3 DATASET’S LABELS AND CLASS INDICES

Figure 3.3 depicts our dataset's labels and class indices. The **train_dataset.class_indices** dictionary maps class names to numerical indices, allowing us to associate classes with their numerical labels. On the other hand,

train_dataset.classes returns an array of the numerical labels assigned to each image in the training dataset, allowing for label inspection and investigation during training.



```
train_dataset.class_indices
```

```
{'Covid': 0, 'Normal': 1}
```

```
train_dataset.classes
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Figure 3.3: Training Dataset Labels and Class Indices

3.4 MODEL IMPLEMENTATION

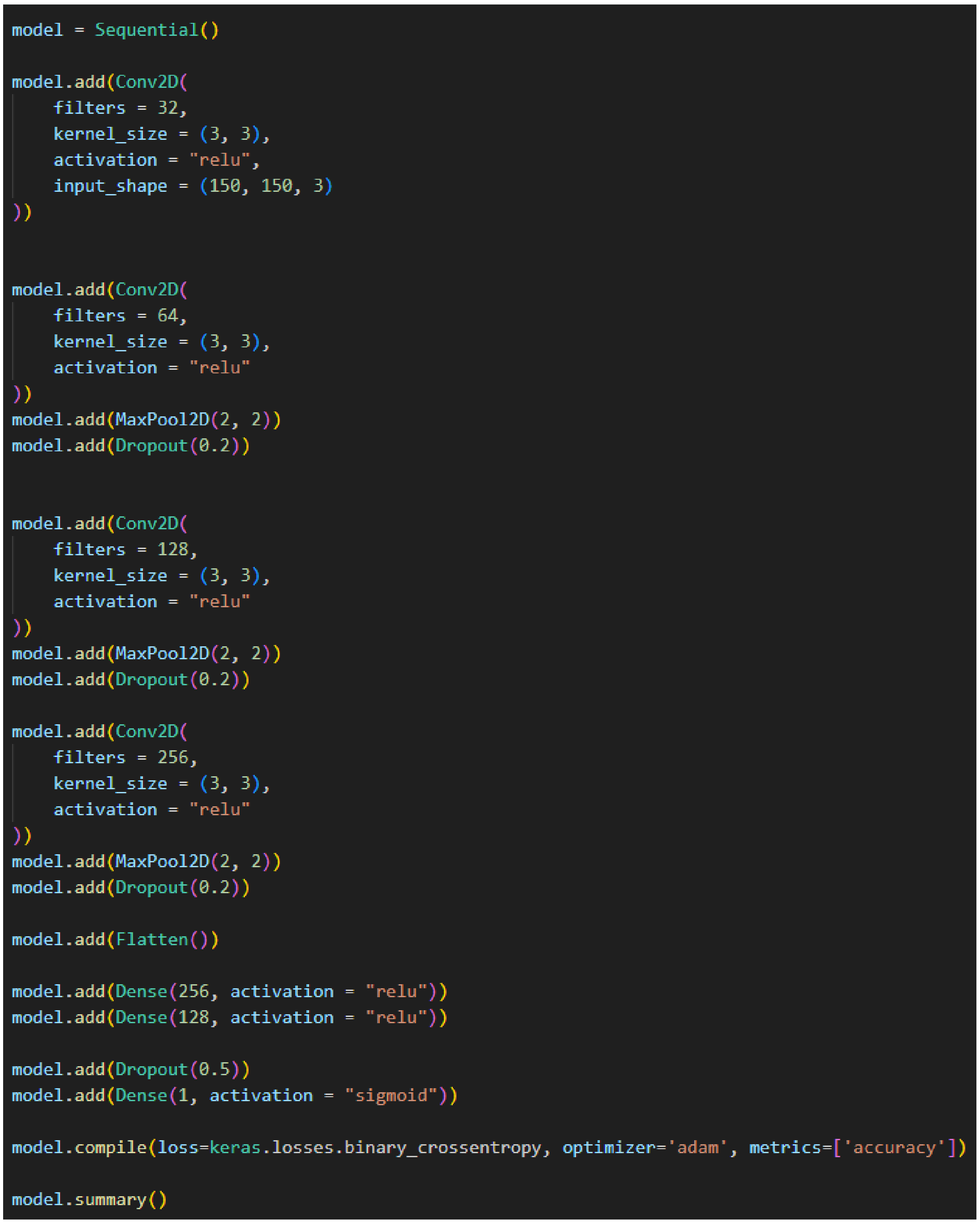
Figure 3.4 depicts the Keras library implementation of our CNN model. The model is built with the sequential class, which supports a linear stack of layers.

The model architecture is made up of several layers. The first layer is a Conv2D layer that convolutional the input images in 2D. It features 32 filters, a 3x3 kernel, and employs the ReLU activation function. The layer's input shape is set to (150, 150, 3), which represents the dimensions of the input photos.

Following Conv2D layers with increasing numbers of filters (64, 128, and 256), MaxPooling2D layers are added to minimize the spatial dimensions of the feature maps. To prevent overfitting, dropout layers are added after each

Conv2D layer.

A flatten layer is inserted after the convolutional layers to turn the 2D feature maps into a 1D vector. This gets the data ready for the fully connected layers.



```
model = Sequential()

model.add(Conv2D(
    filters = 32,
    kernel_size = (3, 3),
    activation = "relu",
    input_shape = (150, 150, 3)
))


model.add(Conv2D(
    filters = 64,
    kernel_size = (3, 3),
    activation = "relu"
))
model.add(MaxPool2D(2, 2))
model.add(Dropout(0.2))


model.add(Conv2D(
    filters = 128,
    kernel_size = (3, 3),
    activation = "relu"
))
model.add(MaxPool2D(2, 2))
model.add(Dropout(0.2))

model.add(Conv2D(
    filters = 256,
    kernel_size = (3, 3),
    activation = "relu"
))
model.add(MaxPool2D(2, 2))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(256, activation = "relu"))
model.add(Dense(128, activation = "relu"))

model.add(Dropout(0.5))
model.add(Dense(1, activation = "sigmoid"))

model.compile(loss=keras.losses.binary_crossentropy, optimizer='adam', metrics=['accuracy'])

model.summary()
```

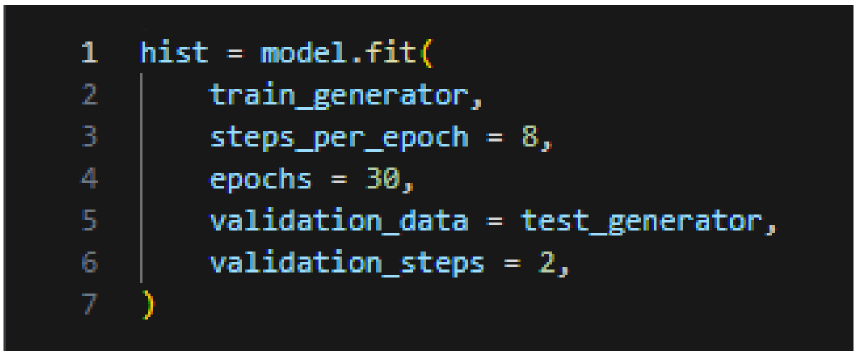
Figure 3.4: Model Creation

The final output layer is a dense layer with a single unit and sigmoid activation that is appropriate for binary classification applications. The binary_crossentropy loss function and the Adam optimizer are used to compile the model. The model's performance during training is also evaluated using the accuracy metric.

Overall, the architecture of this model follows a standard pattern in CNNs, beginning with convolutional layers to extract features from input images, then pooling layers for down sampling, and fully connected layers for classification. The use of dropout aids in the prevention of overfitting, and the selection of activation and loss functions is ideal for binary classification.

3.5 MODEL TRAINING

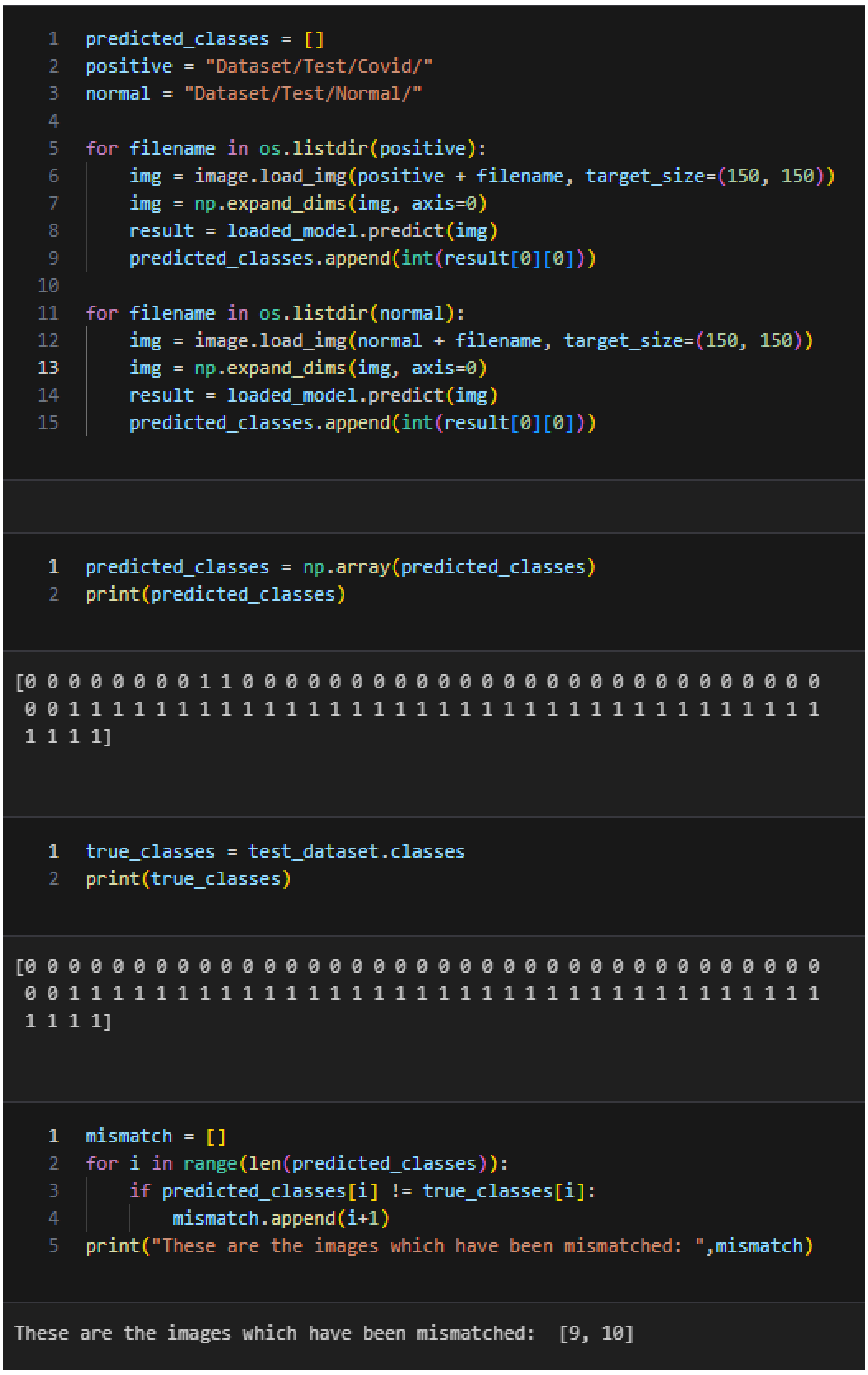
After establishing our model, we use the **model.fit()** function to begin the training process. As the training data source, the **train_generator** generates augmented and normalized image batches. The code defines the number of training steps per epoch, the total number of epochs, and the testing data source (**test_generator**) to be used to assess the model's performance. Iterating over the epochs, adjusting the model's weights based on the training data, and assessing the model based on the testing data comprise the training process. The **hist** object stores the training progress and performance metrics.



```
hist = model.fit(
    train_generator,
    steps_per_epoch = 8,
    epochs = 30,
    validation_data = test_generator,
    validation_steps = 2,
)
```

Figure 3.5: Training the Model

3.6 IMAGE CLASSIFICATION EVALUATION AND MISMATCH DETECTION



```python
predicted_classes = []
positive = "Dataset/Test/Covid/"
normal = "Dataset/Test/Normal/"

for filename in os.listdir(positive):
    img = image.load_img(positive + filename, target_size=(150, 150))
    img = np.expand_dims(img, axis=0)
    result = loaded_model.predict(img)
    predicted_classes.append(int(result[0][0]))

for filename in os.listdir(normal):
    img = image.load_img(normal + filename, target_size=(150, 150))
    img = np.expand_dims(img, axis=0)
    result = loaded_model.predict(img)
    predicted_classes.append(int(result[0][0]))
```

```python
predicted_classes = np.array(predicted_classes)
print(predicted_classes)
```

```
[0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1]
```

```python
true_classes = test_dataset.classes
print(true_classes)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1]
```

```python
mismatch = []
for i in range(len(predicted_classes)):
    if predicted_classes[i] != true_classes[i]:
        mismatch.append(i+1)
print("These are the images which have been mismatched: ",mismatch)
```

```
These are the images which have been mismatched:  [9, 10]
```

Figure 3.6: Predicting the classes of our test dataset

Following the training of our model, we attempt to predict the classes in our test dataset. It loads and processes photos from the test dataset, then runs them through the model to get the predicted classes. The predicted classes are then compared to the true classes to discover any mismatched forecasts. The mismatched image indices are printed, allowing for additional study and evaluation of the model's performance. This analysis sheds light on the image classification model's accuracy and dependability on the test dataset.

In this chapter, we discussed the implementation of our CNN model for image classification, covering datasets, data preprocessing, dataset labels and indices, model implementation, model training, image classification evaluation, and mismatch detection. The effective implementation of this solution establishes the groundwork for reliable image categorization, allowing for further evaluation and analysis of the model's performance.

CHAPTER 4

RESULTS AND DISCUSSION

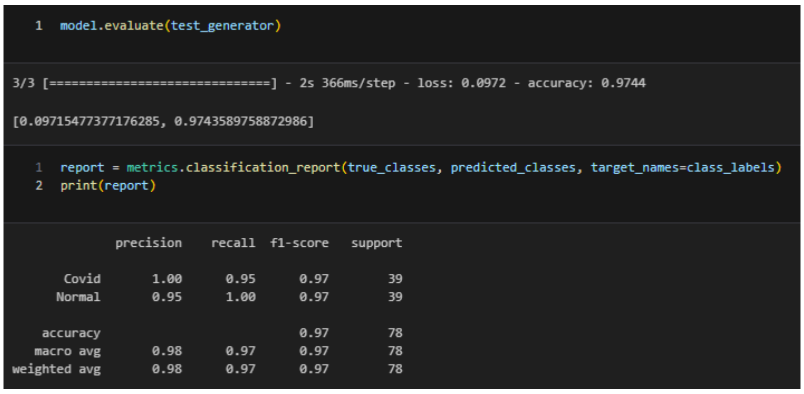
We were able to successfully develop and train a CNN model using Keras. The trained model's accuracy and precision are validated by using sample images of both COVID-positive and normal lung X-ray images.

When given the X-ray images of the patients for prediction, our CNN model is capable of categorizing or identifying COVID-19 X-ray images and normal lung X-ray images. The model accurately predicts the condition based on the supplied X-ray lung image.

4.1 PERFORMANCE EVALUATION

As shown in Figure 4.1, we used the "model.evaluate()" function with the 'test_generator' to evaluate the performance of our trained model on the test dataset. Based on the test data, we were able to quantify the loss value and model accuracy using this code. The evaluation results show how well our model works in terms of predicted accuracy and overall effectiveness in identifying test samples.

We also used the "metrics.classification_report ()" function to get a complete evaluation of the model's performance. This code produces a complete report that includes several performance indicators for each class in our classification challenge, including precision, recall, F1-score, and support. We gained vital insights into the model's accuracy and capacity to correctly classify distinct classes by comparing the true classes ('true_classes') with the predicted classes ('predicted_classes'). The report presented an in-depth analysis of the model's performance, allowing for a better understanding of its strengths and opportunities for improvement.



```
model.evaluate(test_generator)
```

```
3/3 [==============================] - 2s 366ms/step - loss: 0.0972 - accuracy: 0.9744

[0.09715477377176285, 0.9743589758872986]
```

```
report = metrics.classification_report(true_classes, predicted_classes, target_names=class_labels)
print(report)
```

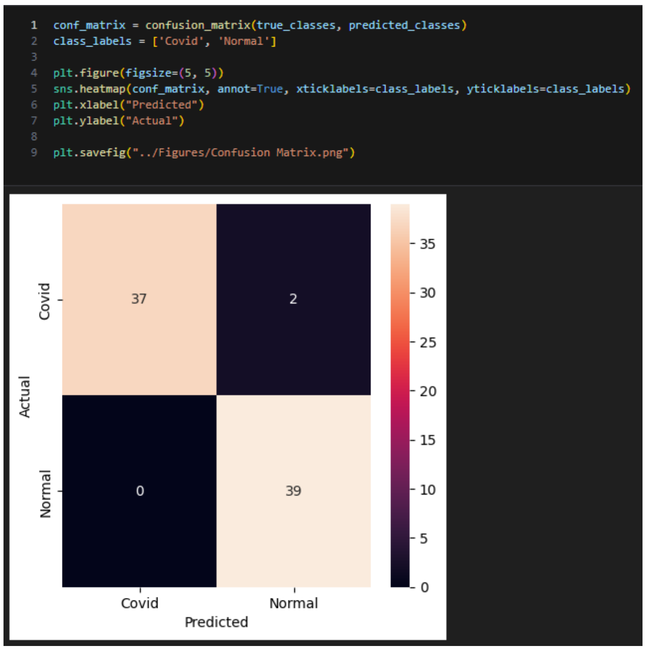
```
              precision    recall  f1-score   support

       Covid       1.00      0.95      0.97        39
      Normal       0.95      1.00      0.97        39

    accuracy                           0.97        78
   macro avg       0.98      0.97      0.97        78
weighted avg       0.98      0.97      0.97        78
```

Figure 4.1: Performance Evaluation

4.2 CONFUSION MATRIX

We used the 'confusion_matrix()' function to construct a confusion matrix to better examine the performance of our model.

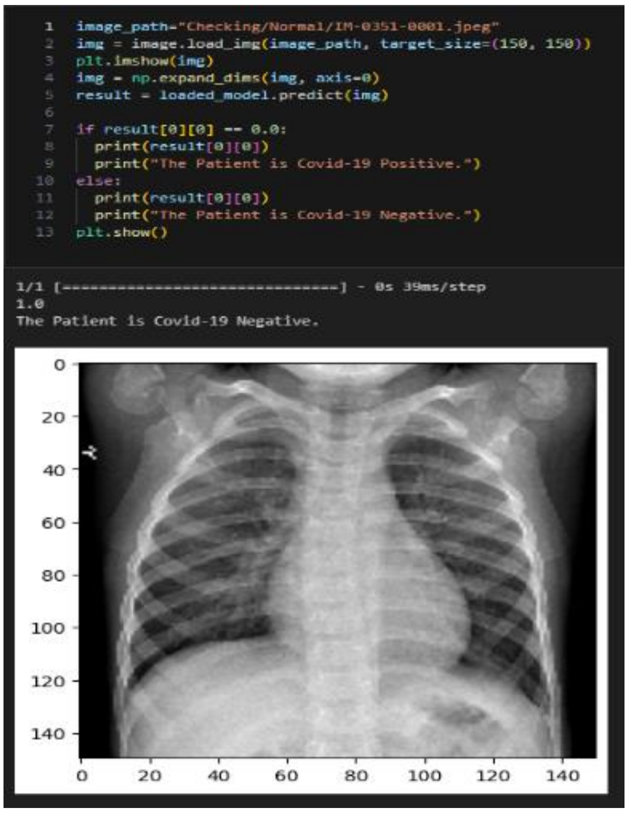
This matrix enabled us to see the distribution of predicted classes versus true classes. We then used the Seaborn package to create a heatmap plot, which offered a graphical representation of the confusion matrix with annotated values. The resulting figure showed the areas of correct and incorrect classifications, providing useful insights into the model's performance across several classes. Our confusion matrix is depicted in Figure 4.2.



```python
conf_matrix = confusion_matrix(true_classes, predicted_classes)
class_labels = ['Covid', 'Normal']

plt.figure(figsize=(5, 5))
sns.heatmap(conf_matrix, annot=True, xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")

plt.savefig("../Figures/Confusion Matrix.png")
```

Figure 4.2: Confusion Matrix

4.3 IMAGE TESTING



```python
image_path="Checking/Normal/IM-0351-0001.jpeg"
img = image.load_img(image_path, target_size=(150, 150))
plt.imshow(img)
img = np.expand_dims(img, axis=0)
result = loaded_model.predict(img)

if result[0][0] == 0.0:
  print(result[0][0])
  print("The Patient is Covid-19 Positive.")
else:
  print(result[0][0])
  print("The Patient is Covid-19 Negative.")
plt.show()
```

```
1/1 [==============================] - 0s 39ms/step
1.0
The Patient is Covid-19 Negative.
```

Figure 4.3: Prediction Model predicting an Image

Finally, we developed a prediction model based on our previously trained model and attempted to predict if the image was COVID-19 positive or negative.

The photos utilized in the prediction model were not used during our model's training and testing. We load the image, resize it to the desired dimensions, and then make predictions with the model that has already been loaded. The "result" output is the model's forecast for the given image.

To evaluate the precision of our model, we used various photographs, one of which is displayed in Figure 4.3.

4.4 GRAPHICAL ANALYSIS

During the model's development, we produced graphs for training and testing accuracy, as well as training and testing loss over the epochs. It aids us in visualizing the performance of our model during the training phase.

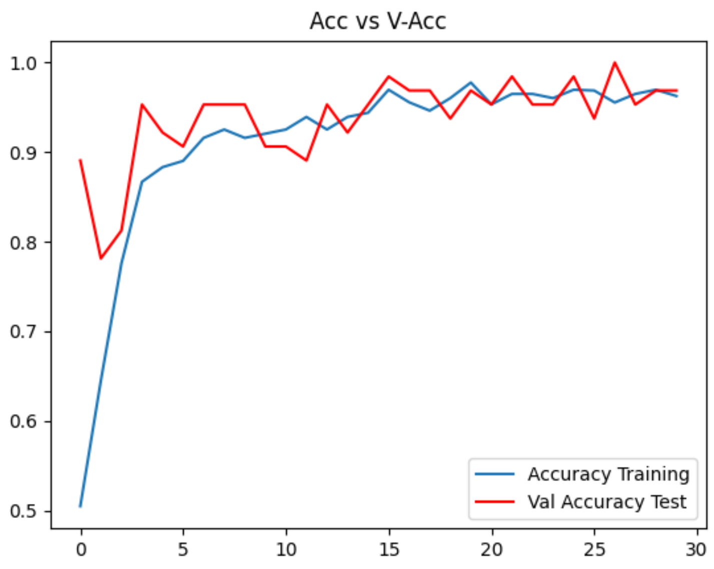


Figure 4.4: Line Graph of Training Accuracy vs Testing Accuracy

As illustrated in Figure 4.4, the first figure contrasts training accuracy (blue line) with testing accuracy (red line) over epochs. It provides an overview of how effectively the model is learning from training data and generalizing to previously unknown testing data.

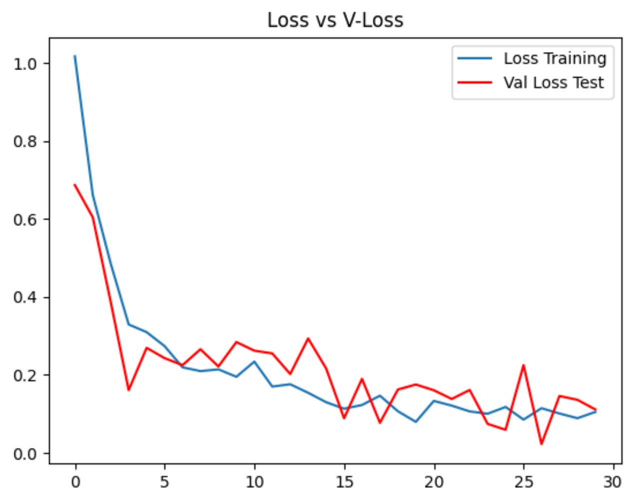


Figure 4.5: Line Graph of Training Loss vs Testing Loss

As shown in Figure 4.5, the second figure contrasts the training loss (blue line) and testing loss (red line) over the epochs. It depicts the amount of error encountered by the model during training and testing. Lower values denote superior performance.

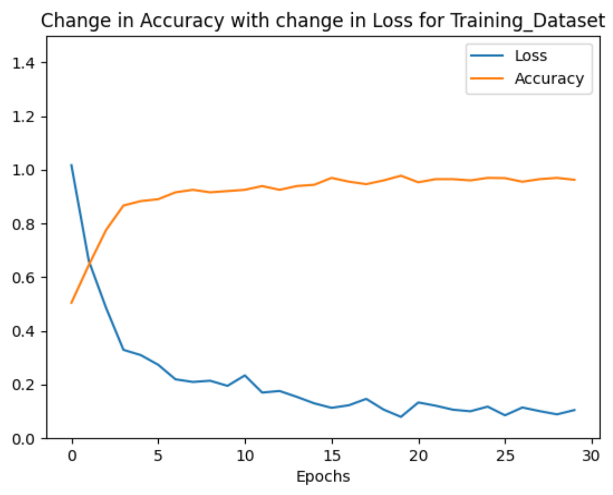


Figure 4.6: Line Graph for Change in Accuracy with change in Loss for Training_Dataset

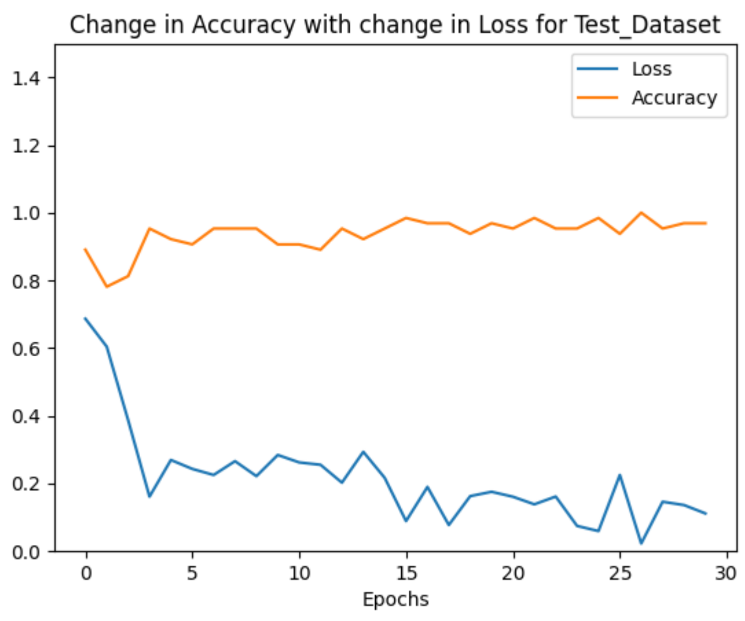


Figure 4.7: Line Graph for Change in Accuracy with change in Loss for Testing_Dataset

As demonstrated in Figures 4.6 and 4.7, the third and fourth plots indicate the relationship between accuracy (orange) and loss (blue) for both the training and testing datasets over the epochs. It aids in analyzing how accuracy changes in relation to loss during the training process.

CHAPTER 5

CONCLUSION AND FUTURE DECISION

In conclusion, we constructed a convolutional neural network (CNN) model that was used to build a COVID-19 detection system utilizing X-ray datasets, and the results were promising. We built the deep learning model with a CNN, which provides adequate image classification. Many tests were run on each dataset to demonstrate the performance of the proposed model. In the first experiment, we trained a model using a CNN with different layers on the first dataset, and then we used the same model constructor to train the remaining datasets. To obtain a robust model, we changed the model's hyperparameters for each dataset. In the second experiment, in the absence of the proposed model, we applied different machine learning techniques to each dataset. This highlighted the value and importance of the proposed model. Despite the scarcity of examples in the dataset, our model exhibited good COVID-19 classification accuracy. Finally, the classification rates of our strategy were compared to earlier studies, and the created approach performed the best.

Several aspects can be investigated considering future decisions of this project. To increase the model's generalization skills, the dataset can be enlarged by integrating more diverse and representative samples. Furthermore, by fine-tuning hyperparameters or applying sophisticated approaches such as transfer learning, the model architecture can be further optimized. Experimenting with different pre-processing approaches or enriching the dataset can also help the model perform better. Furthermore, the model's implementation in a real-world scenario might be evaluated, assuring its integration into clinical procedures, and assessing its impact on medical diagnosis. Continuous monitoring, updates, and changes to the model can be pursued to respond to changing medical settings and improve its detection of COVID-19.

REFERENCES

[1] Daksh Trehan, “Detecting COVID-19 using Deep Learning”, Towards Data Science, 2020.

[2] Yousef Alhwaiti, Muhammad Hameed Siddiqi, Madallah Alruwaili, Ibrahim Alrashdi, Saad Alanazi, and Muhammad Hasan Jamal, “Diagnosis of COVID-19 using a Deep Learning Model in Various Radiology Domains”, Hindawi, 2021.

[3] Aijaz Ahmad Reshi, Furqan Rustam, Arif Mehmood, Abdulaziz Alhossan, Ziyad Alrabiah, Ajaz Ahmad, Hessa Alsuwailem, and Gyu Sang Choi, “An Efficient CNN Model for COVID-19 Disease Detection Based on X-Ray Image Classification”, Hindawi, 2021

[4] Dandi Yang, Cristhian Martinez, Lara Visuna, Hardev Khandhar, Chintan Bhatt & Jesus Carretero, “Detection and Analysis of COVID-19 in medical images using deep learning techniques”, Nature, 2021

[5] Fatima Aurora Saiz, and Iñigo Barandiaran, “COVID-19 Detection in Chest X-ray Images using a Deep Learning Approach”, ResearchGate, 2020