Visvesvaraya Technological University

Belagavi

A Mini Project Report on

"CONVOLUTIONAL NEURAL NETWORK BASED IMAGE CLASSIFICATION FOR MEDICAL APPLICATION"

SUBMITTED BY:

JAYANTH S	**USN:1NH19EC045**
HUSSAIN PEERA	**USN:1NH19EC043**
G NIRANJAN REDDY	**USN:1NH19EC037**
K HARSHAVARDHAN	**USN:1NH19EC144**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ELECTRONICS & COMMUNICATION

2021-22

NEW HORIZON COLLEGE OF ENGINEERING

New Horizon Knowledge Park, Ring Road, Marathalli

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC

Accredited by NAAC with 'A' Grade, Accredited by NBA

NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade. Accredited by NBA

BENGALURU-560103

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the mini project work entitled **"CONVOLUTIONAL NEURAL NETWORK BASED IMAGE CLASSIFICATION FOR MEDICAL APPLICATION"** carried out by **Jayanth S (1NH19EC045), HUSSAIN PEERA (1NH19EC043), G NIRANJAN REDDY (1NH19EC037), K HARSHA VARDHAN (1NH19EC144)** bonafide students of Electronics and Communication Department, NHCE, Bengaluru in partial fulfillment for the award of Bachelor of Engineering in Electronics and Communication of the Visvesvaraya Technological University, Belagavi during the year 2021-22. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The mini-project report has been approved as it satisfies the academic requirements in respect of the mini-project work prescribed for the said degree.

Signature of the Guide	Signature of the HoD
Ms. Divya Sharma	**Dr. Sanjeev Sharma**
Sr. Assistant Professor Department of ECE NHCE, Bengaluru	Professor & HoD Department of ECE NHCE, Bengaluru

External Viva

Name of the Examiners	Signature with Date
1. _____	1. _____
2. _____	2. _____

Mini_Project_Report_NN_Divya_Sharma.docx

ORIGINALITY REPORT

23%	16%	10%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	5%
2	Submitted to Engineers Australia Student Paper	3%
3	www.hindawi.com Internet Source	3%
4	medium.com Internet Source	1%
5	Submitted to Manipal University Student Paper	1%
6	www.brilliantread.com Internet Source	1%
7	Submitted to Coventry University Student Paper	1%
8	Submitted to International Islamic University Malaysia Student Paper	1%
9	www.ijert.org Internet Source	1%

ABSTRACT

Keywords: Convolutional Neural Network (CNN), Modelling, Keras API, Google Colab, Kaggle, Google Drive, Open Source, Covid-19 detector, Chest X-Ray

In this mini-project, we work on designing a COVID-19 detection using the Convolutional Neural Network model with support of Open-Source software such as Keras, Python, Google Colab, Google Drive, Kaggle, and Visual Studio. All these Open-Source applications and software allow us to aggregate, design, create, train, visualize, and analyze bulk load of data on the cloud after programing a Deep neural network without a need for high-end processing hardware. The historical data that we will be using is Covid Chest X-Rays.

Here, Python is an interpreted high-level general-purpose programming language and Keras is an open-source software library that provides a Python interface for artificial neural networks. It is an API designed to reduce the complicity in the programming of a neural network program. Kaggle is an online community of data scientists and machine learning practitioners, from where we obtained the dataset of Covid-19 positive and Normal lung scan images for analysis. Google Colab is a free Jupyter notebook environment that runs entirely in the cloud, allows anyone with internet access and a Google account to write and execute, arbitrary Python code to complex Machine Learning and Deep Learning algorithms through the browser. It is well suited for machine learning, data analysis students as it is free. Colad is the platform where we will be modeling out CNN. We collect the data from the Kaggle dataset process the data as per our requirement using Visual Studio and save it in Google Drive which will be mounted onto the Google Colabe notebook in which we will be modeling our CNN. Once modeled we can train the model using the data that has been saved in Google Drive to get the Weights of the CNN model. This Weights can be used to test, analyze the accuracy, visualize and predict the condition of a lung using chest X-Rays at certain accuracy. This will help in identifying the problems of the patients at a faster rate, thus giving an appropriate treatment at an early stage itself to saving one life.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement helped us succeed.

We thank **Dr. Mohan Manghnani**, Chairman of **New Horizon Educational Institution**, for providing the necessary infrastructure and creating a good environment.

We also record here the constant encouragement and facilities extended to us by **Dr.Manjunatha**, Principal, NHCE, and **Dr. Sanjeev Sharma**, Head of the Department of Electronics and Communication Engineering. We extend sincere gratitude to them.

We sincerely acknowledge the encouragement, timely help, and guidance to us by our beloved guide **Ms. Divya Sharma** to complete the project within the stipulated time successfully.

Finally, a note of thanks to the teaching and non-teaching staff of the electronics and communication department for their co-operation extended to us, who helped us directly or indirectly in this successful completion of the mini-project.

JAYANTH S (1NH19EC045)
HUSSAIN PEERA (1NH19EC043)
G NIRANJAN REDDY (1NH19EC037)
K HARSHAVARDHAN (1NH19EC144)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Syndrome coronavirus 2 (SARS-CoV-2) or shortly COVID-19 are family of viruses that can cause respiratory illness in humans was first discovered in 2019 in China Wuhan province. At present 2022 it is an ongoing global pandemic that has caused more than 300 million cases and 5.47 million deaths, 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 placing 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, but still 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.

In short, X-Ray is a type of radiation called electromagnetic waves, which are used in creating X-Ray imaging to create a picture of the inside of our body, in the above-mentioned case lung scan. A limited quantity of radioactive matter known as Tracer conveys gamma beams when the beams are gotten by the scanner to snap a photo of the lungs.

The X-Ray imaging strategy accompanies incredible benefits which incorporate its minimal expense, simple accessibility of X-Ray offices, less time utilization, and so forth Hence, X-beam imaging might be viewed as a superior possibility for the mass, simple, and fast analysis technique for a pandemic like COVID-19 thinking about the current worldwide medical services emergency, as the significant disease brought about by this infection will in general stay consistent.

Thus, in this project, we intended to demonstrate how we can apply neural networks i.e., in particular, CNN classification to differentiate a healthy human lung from one that is infected with Covid-19 utilizing these X-Ray images. The proposed model is created to give accurate diagnostics for binary classification (COVID Positive versus Negative) predications on the essentials of X-Ray images. Our model created a normal grouping exactness of 96.153% for binary classes.

This kind of system can be extensively modified for different end-user applications. For example, it could be used for Interstitial lung disease (ILD) detection, etc... In various places

- Ocular Disease Recognition using CNN
- Heart rate analyzing and diagnosis using CNN

- Health monitoring using Smart device data analysis
- Health care Automated monitoring system…

Why use CNN in the project?

CNN is used in our project because it gives high accuracy when compared to other Neural Networks, also as CNN is mainly used for image classification, it can be used in medical and various other fields for the application of computer vision. With the help of a data set, we can give the necessary information to train a model to identify and classify the given data by self-learning what is what.

Relation between CNN, AI, and Deep Learning:

Al in general is getting a computer to imitate human behavior in some given way, coming to Machine Learning is a subset of AI which consists of methods or techniques that enable computers to figure out things from the given data set and give it to Al for processing the information. Deep Learning meanwhile is a subset of Machine Learning which enables computers to solve complex problems through the process of pattern recognition.

In CNN, which has a powerful image processing capability, Al uses deep learning to perform both generative and descriptive tasks often use as machine vision which comes under Image and Video recognition along with language processing stitch. These all are the application and process where the information used in the medical field for better and high accuracy purpose and help in living a better life.

Chapter 2

LITERATURE SURVEY

Table No: 2 Literature Survey

No.	Article Title	Findings	Outcome
1	**Deep-chest: multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases**	Reference to architecture and dataset references.	Understood how the architecture should be and got references to datasets
2	**An Efficient CNN Model for COVID-19 Disease Detection Based on X-Ray Image Classification (https://www.hindawi.com/journals/complexity/2021/6621607/)**	Architecture and Dataset references	The basic architecture and essential parameter references
3	**Coursera** **Neural Networks and Deep Learning**	Theory of ANN and CNN	Understood the theoretical concepts of ANN and CNN in detail.
4	**An Efficient CNN Model for COVID-19 Disease Detection Based on X-Ray Image Classification** (https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)	Understanding Convolutional Neural network	Learning theoretical concepts of CNN with e.g.
5	**Kaggle Dataset.** (https://www.kaggle.com/search?q=covid+19+image+dataset)	Datasets of Covid-19 and Normal lungs	We got an image dataset of Covid-19 and Normal
6	Keras (https://keras.io/)	Working and inbuilt function sheet of Keras API	Understanding how Keras work and how to program using it

7	Matplotlib (https://matplotlib.org/)	Used to get function sheet of matplotlib lib	Application of matplotlib to data
8	OS (https://www.geeksforgeeks.org/os-module-python-examples/)	How to interact with system OS using python	Understood how to handle files using Python and OS
9	Numpy (https://numpy.org/)	Data handling functions lib	Handling data using inbuilt functions
10	scikit-learn (https://scikit-learn.org/stable/)	Features various classification, regression, and clustering algorithms lib	Handling classification and accuracy auto-setup lib
11	Visual Studio (https://visualstudio.microsoft.com/)	Visual studio software	Used to process data as per the requirement
12	Google Colab (https://colab.research.google.com/?utm_source=scs-index)	Website for modeling the CNN	Cloud-based Hardware GPU for data processing and analysis
13	Application of Deep Learning in Medical field (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6945006/)	Application of Deep learning in medical image analysis	Advantage and use of applying Deep learning in medical image analysis

Chapter 3

3.1 Proposed Methodology

Identifying and giving the right diagnosis at a low cost to sickly people at the earliest is one of the biggest problems faced by the healthcare sector, due to the increasing number of problems such as Covid-19, pneumonia, and many more in the field of lung disorders alone. Analysis of the medical reports and finding a proper diagnosis to the patients it-self-consumes a lot of time, which can be crucial.

To solve this kind of issue while diagnosing COVID-19 concerned patients, we have proposed to use Deep Learning as a solution. Trained a Convolutional Neural Network (CNN), similar to the one referred to in fig 3.1, to identify the problem with high accuracy using chest X-ray images of sickly, we can reduce the time required for report analysis by a huge extent. This will help in identifying the problems of the patients at a faster rate, thus giving an appropriate treatment at an early stage itself to saving one life.

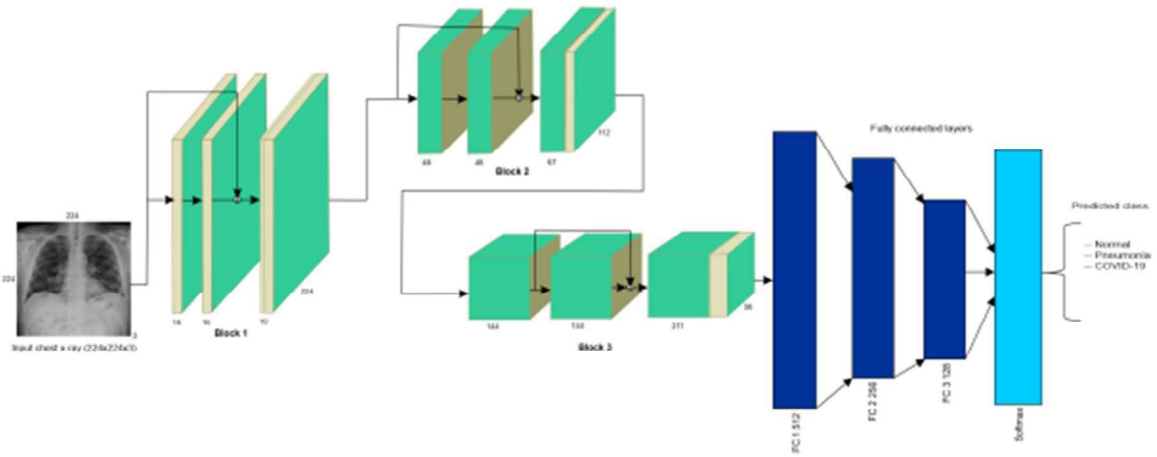

Fig 3.1 Pictorial Representation of CNN Model Being Designed

The project aims to build an n-layer Convolutional neural network, which will be trained using the training dataset and get the weights of the network, which can be used on test-dataset for analysis of the CNN model performance and optimize it for better accuracy. Once done we should be able to use that network of medical applications to identify an unhealthy lung using **Binary Classification**.

Neural Network:

A neural network is a network of neurons (or) in the modern sense, an artificial neural network that is composed of artificial neurons nodes. It is similar to that of the biological neural network is composed of a group of chemically connected to a functioning associated neuron. A single neuron may be connected to many other neurons and form a network. The connection in a neuron is modeled in ANN as weights between nodes.

Artificial Neural Network:

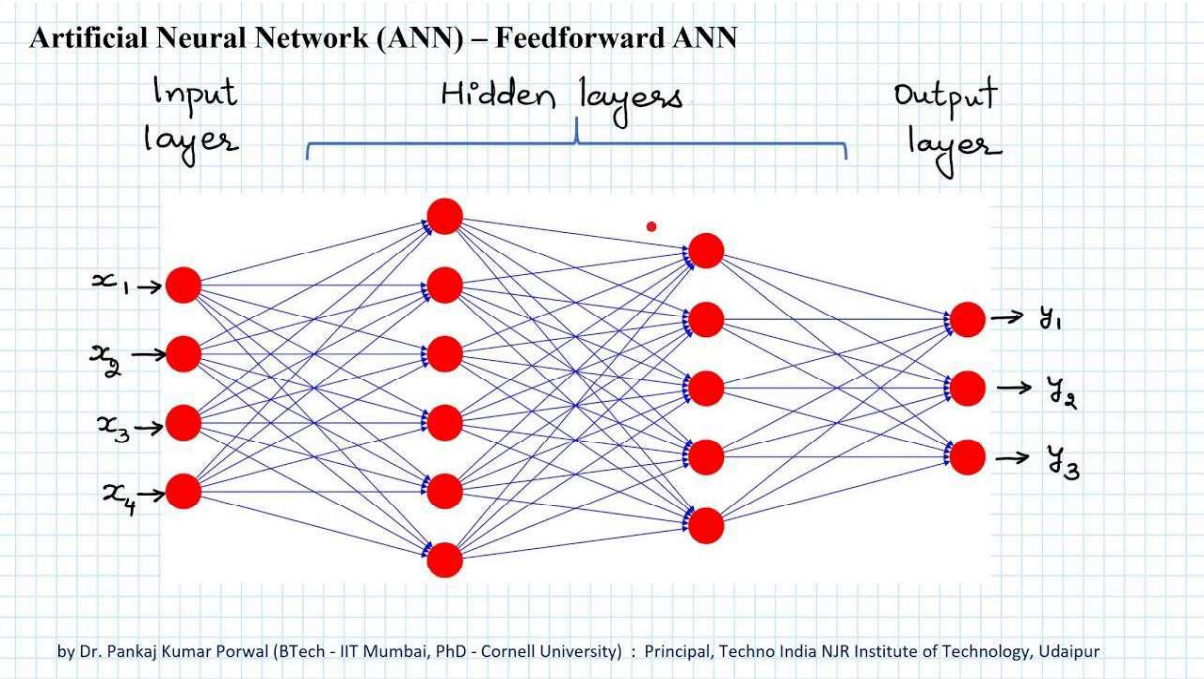

Fig 3.2 General Artificial Neural Network

As we can see the general representation of the neural network in figure 3.2, where if X is the input that we are given and y is the output that we get out, then all the other things in between these 2 things which are helping to determine the Y using X are called Hidden layers. All of these networks together form Artificial Neural Network.

ANNS are Composed of numerous hubs which mimic natural neurons of the human cerebrum. The neurons are connected by links and interact with each other. The nodes can take input information and perform a simple program. The result is passed to other neurons and the output of each node is activation (or) node valves.

Convolutional Neural Network:

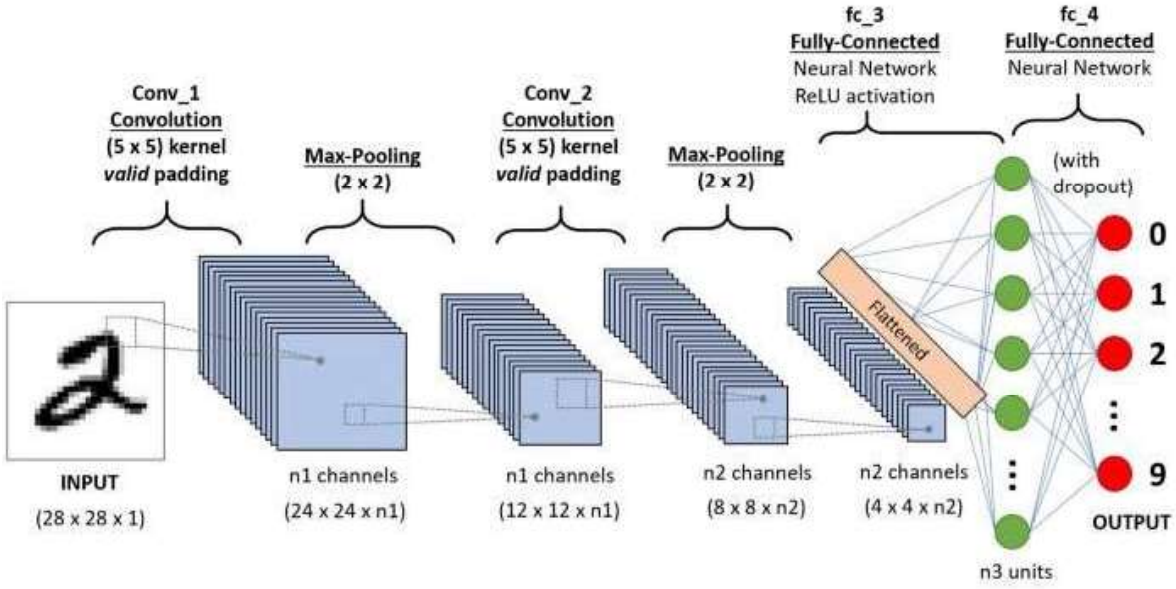

Fig 3.3 General Convolutional Neural Network

A CNN is a sort of ANN which is utilized in image processing and image recognition that is explicitly utilized for handling pixel information. CNN is a powerful image processing tool that can perform both generative and descriptive tasks which are often used by machine learning for image and video analysis.

In CNN we first process the data to lower order through various mathematical tools, then the data is flattened and given to a classical ANN network for further analysis.

Block Diagram of Proposed System:

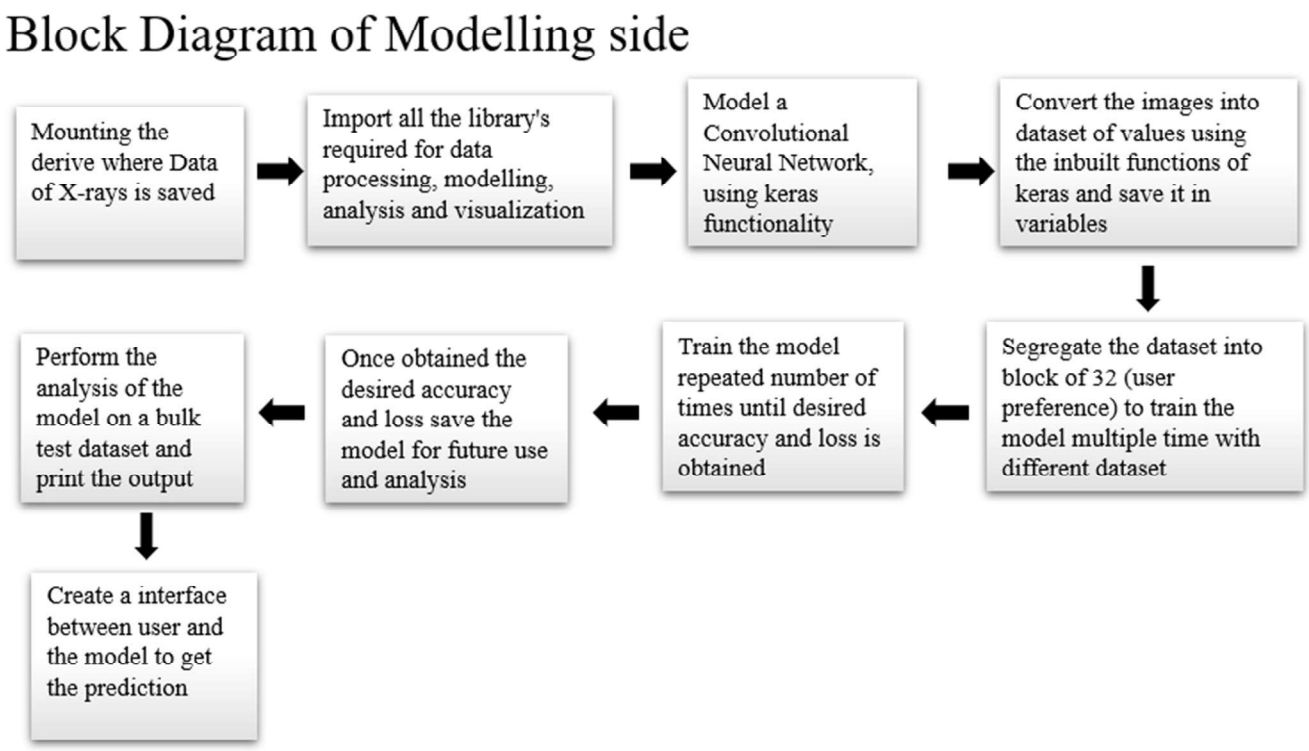

Fig 3.4 Block Diagram

Importing Libraries:

The process of creating a link between the libraries we are going to use in the program while programming such as Keras, matplotlib, etc.

Imported Libraries:

1. Numpy
2. Matplotlib
3. OS
4. Keras
    - Layer → Conv2D, MaxPool2D, Dropout, Flatten, Dense
    - Image → image, ImageDataGenerator
5. Sklearn
    - Metrics → classification_report, Confusion_matrix
6. Seaborn
7. TensorFlow

Mounting the Drive to the Program:

Creating a connection between the drive-in in which we have saved our image dataset and the program in which we are creating our CNN model.

Data Gathering and Segmentation:

Data gathering is one of the most important aspects of creating a model, that is because the quality and quantity of the dataset impact the output of the model to a great extent. Balancing Dataset is important to work on the presentation of the proposed CNN models in the detection of COVID-19, we have gathered a total of 489 lung X-Ray images of two classes, 282 images of COVID-19 positive lung X-Ray images, and 207 images of Normal lung X-Ray images. These 489 images are divided into 2 sets, Training (242 positive, 242 normal) images and Test (39 positive, 39 normal) image dataset. These connected additional X-Ray pictures were downloaded from Kaggle. The process of dividing the available dataset into groups such that it can be used to train the model with different datasets for every epoch data segmentation.

Table No: 3.1 Data Distribution Table:

	Covid-19 Positive X-Ray Image	**Normal X-Ray Image**
Training set	242	242
Test set	39	39

Processing the Data:

1. *Image to data conversion*:

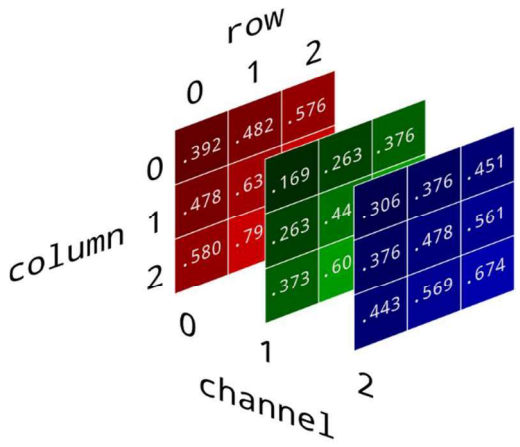

Fig 3.5 Image Conversion from RGB to RGB Matrix

The process of converting the image (RGB image) into a matrix of shape (width, height, depth) dataset with a respective pixel value between 0 - 255 can be seen in figure 3.6

2. *Convolutional (Conv2D):*

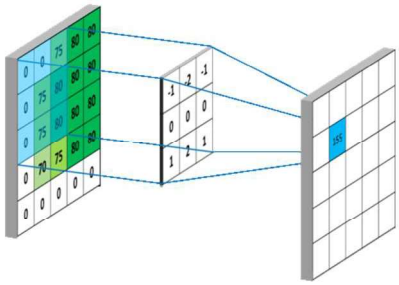

Fig 3.6 Convolution (Conv2D) Pictorial Representation

The process of application of a filter onto an input result in activation, which is simply known as Convolution is shown in figure 3.7. Rehashed utilization of the same filter to a piece of information, results in a map of activations called a feature map, showing the areas and strength of a recognized element in input, like a picture.

3. *Max-Pooling:*

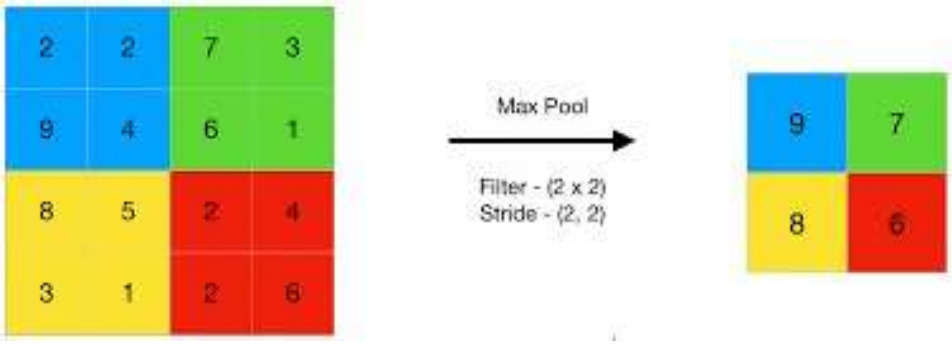

Fig 3.7 Max-Pooling Pictorial Representation

Max pooling is a pooling activity that chooses the greatest component from the area of the feature map covered by the channel. In this way, the result after the maximum pooling layer would be a feature map containing the most noticeable elements of the previous feature map. One such example is shown in figure 3.8.

4. *Flatten Layer:*

The process of converting a matrix of order M*N into (M, N*1) that is into a column matrix.

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

5. *ReLU activation functions:*

The rectified linear activation function or ReLU for short is a linear function that will output the zero if the input is less than zero, and the same value as input is greater than zero. The graphical of ReLU is given the figure 3.9

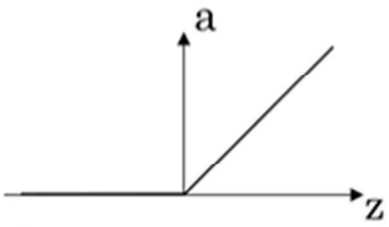

Fig 3.8 ReLU Function Graph

ReLU: $\boxed{g(z) = \max(0,\ Z)}$

Derivative:

$$\frac{\mathrm{dg(z)}}{\mathrm{dz}} = \begin{cases} 0; if\ z < 0 \\ 1; if\ z \geq 0 \end{cases} \rightarrow \text{Equ 3.1}$$

6. *Sigmoid activation function:*

The reason we use the sigmoid function is that it exists between (0 to 1). Thus, it is especially used for Neural Networks where we have to predict the probability as a binary output. Since the probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. As our output is a yes or no type of classification that is binary classification in the last stage, we make it a sigmoid function in the last layer of CNN. The Graph and function of the Sigmoid activation function is given in figure 3.10

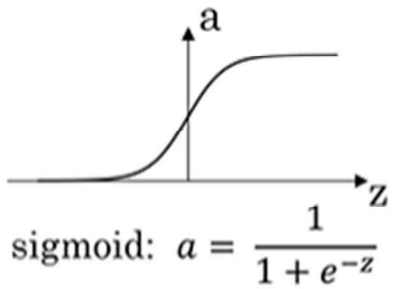

Fig 3.9 Sigmoid Function Graph

7. *Adam Optimizer:*

$$For\ each\ Parameter\ w^j$$
$$(j\ subscript\ dropped\ for\ clarity)$$
$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*
g_t : *Gradient at time t along* ω^j
ν_t : *Exponential Average of gradients along* ω_j
s_t : *Exponential Average of squares of gradients along* ω_j
β_1, β_2 : *Hyperparameters*

Fig 3.10 Adam Optimizer Formula

Adam optimizer is a mix of two slope plunge improvements. This algorithm is utilized to speed up the gradient descent calculation by thinking about the consideration of the exponentially weighted average. Utilizing midpoints causes the calculation to combine towards the minima at a quicker pace.

Parameter Setup:

Varies parameters have to be set up while modeling CNN before training it such as input size is set to (150, 150,3) i.e., any image input will be resized to this configuration, the parameter of Initial, Exponential Average, the Learning rate of gradients, Exponential average of squares of gradients, Hyperparameters are auto-configure 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'])
```

Fig 3.11 Code implication of Adam optimizer

Model analysis:

The process of analyzing the accuracy and the reliance of the model using the test dataset for a better understanding of the trained model and visualization of the output model is known as model analysis.

Creating an Interface:

Creating a method where the use a give his image to get the predication whether it is Covid-19 positive or Negative, and the predication is displayed on to the screen along with the image.

3.2 Proposed CNN Model

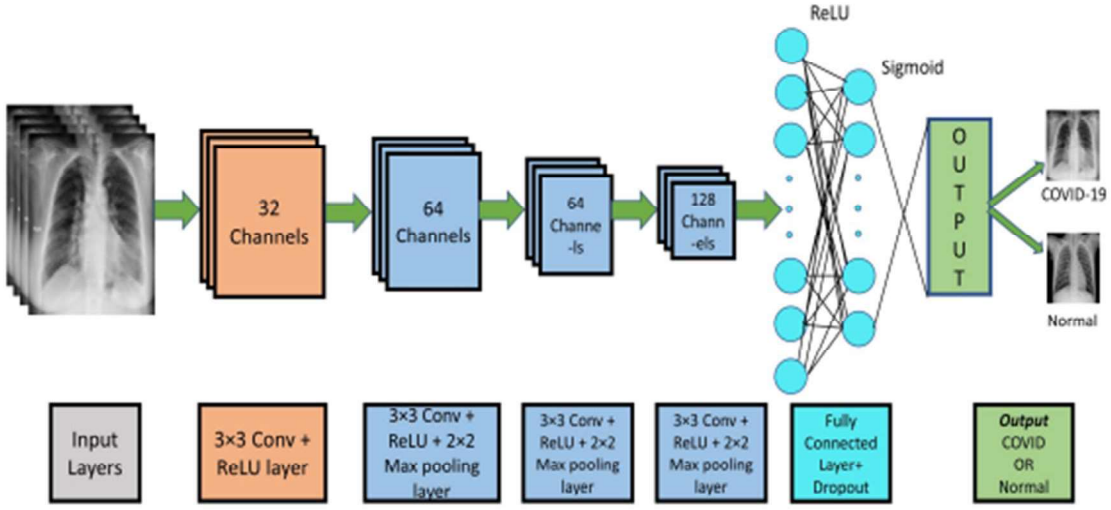

Fig 3.12 Pictorial Representation of CNN model

The Process of Creating a Model of Convolutional Neural network by setting the flow of the model as shown in the figure and setting the essential parameters to train the model is known as modeling CNN.

The last CNN model comprises 16 layers out of which 4 are Convolutional, 4 Max_Pooling layers, 4 Dropout layers, 56 bunch standardization layer for the preparation set and 1 flatten layer, and 2 fully connected layers, 6 actuation layers (5 ReLU layers and last one sigmoid layer); input picture state of CNN model is (150, 150, 3). All the Convolutional layers are handled with a 3*3 size filter or kernels, the number of filters been utilized in each convolutional layer is 32, 64, 128, 256 individually. After each con2D layer, the Max-pooling layer with a 2*2 size Max_Pooling has been utilized. The actuation layer is being applied with the ReLU function, and the dropout layer has been utilized with a 25% dropout rate. The result of 65536 neurons of 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, 1 respectively. Since only one output node is needed to classify the data to one of 2 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 No: 3.2 *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 (MaxPooling2D)	(None, 35, 35, 128)	0
dropout_1 (Dropout)	(None, 35, 35, 128)	0
conv2d_3 (Conv2D)	(None, 33, 33, 128)	295168
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 256)	0
dropout_2 (Dropout)	(None, 16, 16, 256)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 256)	16777472
dense (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129
Total params: 17,198,913 Trainable params: 17,198,913 Non-trainable params: 0		

Flowchart of the Program:

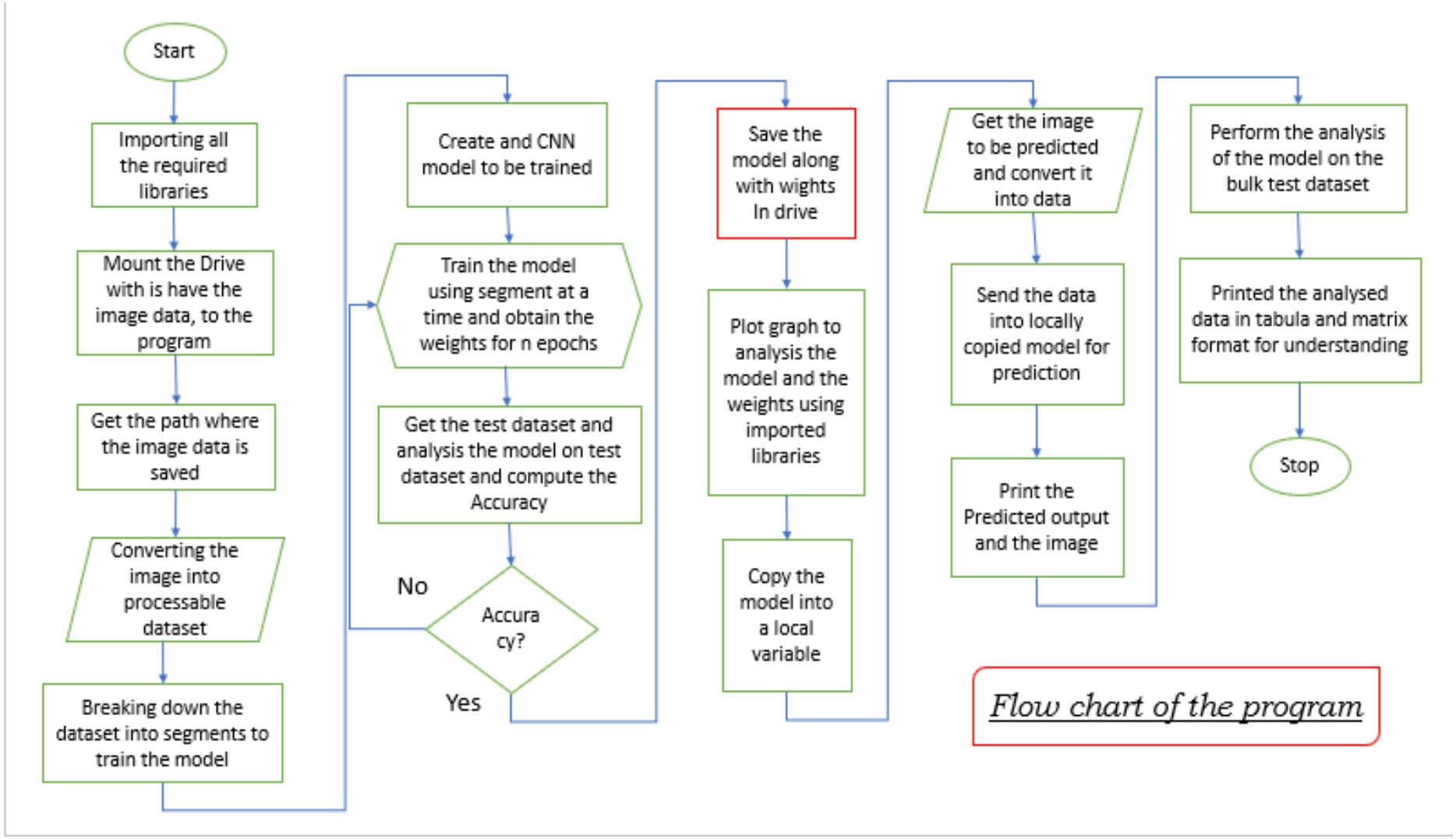

Fig 3.13 Flowchart of the Program

Flowchart of the CNN Model.

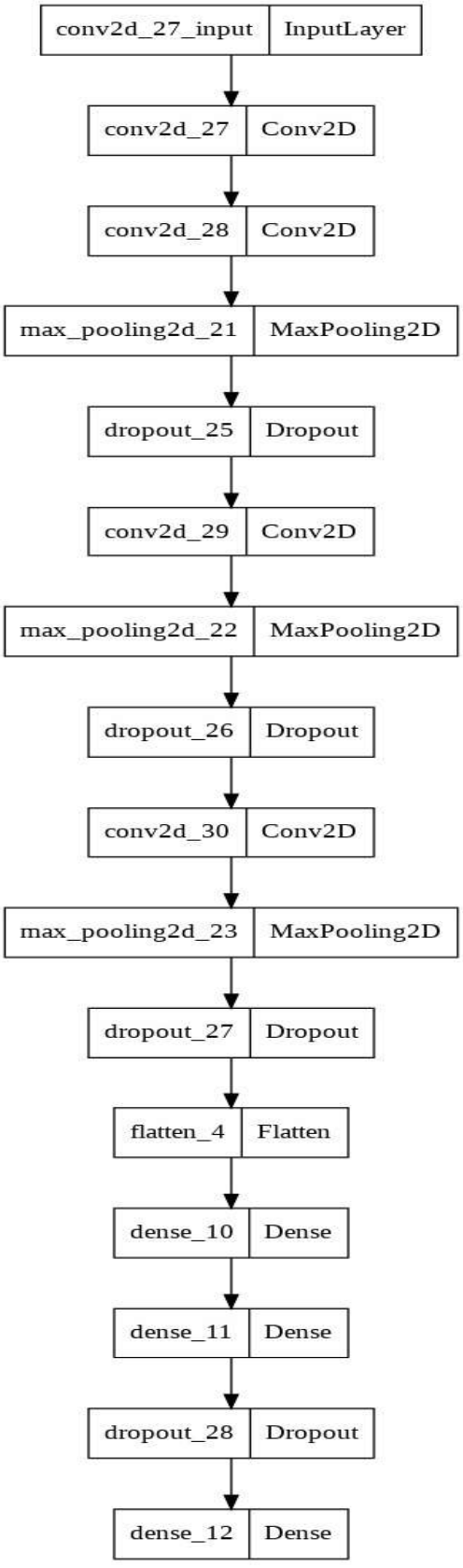

Fig 3.14 Flowchart of CNN Model

AlexNet Style Representation of Model :

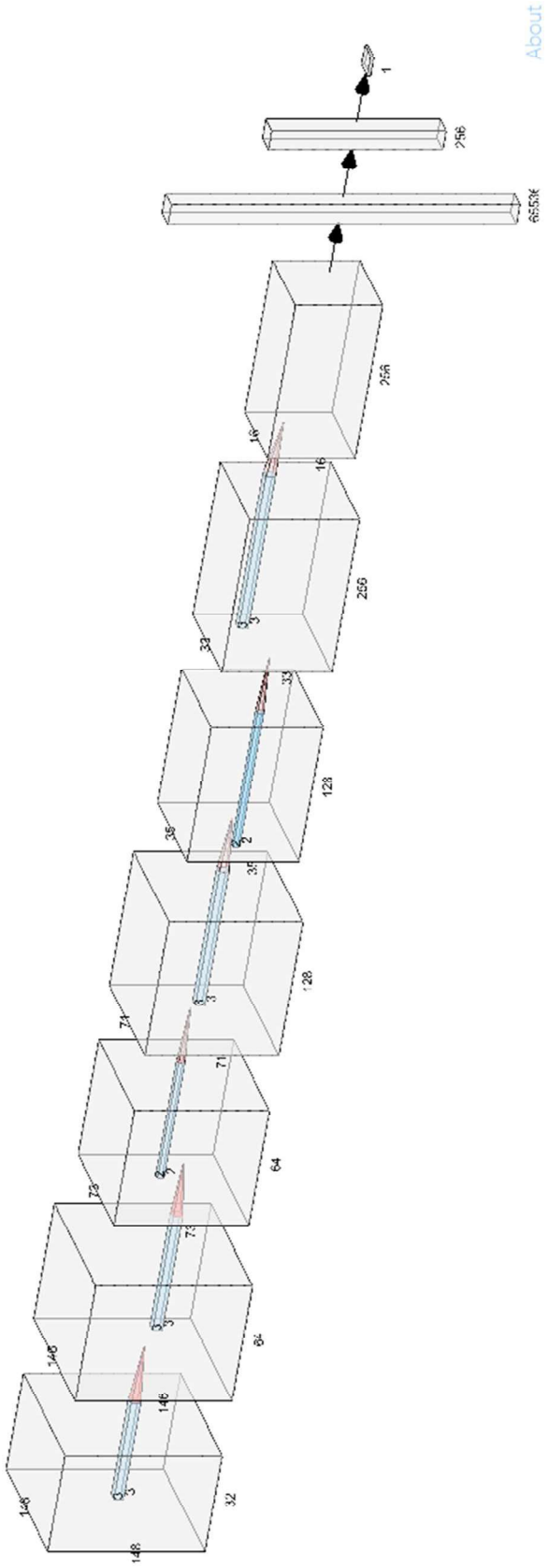

Fig 3.15 AlexNet Style Representation of Model

Chapter 4

Project Description

4.1 Software Description:

Google Colab

"Colab" for short, is a product from Google Research in a type of web application that permits anyone to compose and execute code in python, or code that upholds numerical conditions, and it is appropriate for AI, information investigation, applications. Colab is a facilitated Jupyter scratchpad administration, simply need a google record to log in to the help and we can begin dealing with it. It gives leisure time confined admittance to registering power like GPU and TPU.

It accompanies pre-introduced AI libraries like Keras, TensorFlow, PyTorch, and so forth, saved on the cloud. It utilizes the Anaconda distribution of Jupyter Notebook alongside libraries, like Pandas, NumPy, Matplotlib. We can execute our whole model in this stage (Google Colab), using the GPU furnished by the stage effortlessly.

Google Colab: https://research.google.com/colaboratory/

Keras

Keras is an open-source software Neural Network library with high-level APIs written in python, it is a simple, flexible and powerful library for implementing Deep Neural Networks. It is a high-level neural network library that is running on CNTK, TensorFlow, and Theano. Deep Learning modeling allows for prototyping as well as running seamlessly Google Colab GPU and TPU. The framework is written in easy to debug and program programming language python code.

Keras lets us design deep neural models and allows distributed training of those models on GPUs or TPUs.

Keras: https://keras.io/

NumPy

NumPy is a core library that is used for working with array and scientific computing in python, its community-driven open-source project. It has functions for working in linear algebra, Fourier transform, and matrices. It gives an elite presentation complex exhibit of articles. It assumes a part in the AI library as it has a great deal of helpful inbuilt libraries.

Numpy: https://numpy.org/

Matplotlib

Matplotlib is an Open-Source multiple-platform data visualization, plotting library used in python programming language for 2D plots of arrays. It is comprehensive for creating static, animated, and interactive visualizations in python.

Matplotlib: https://matplotlib.org/

Visual Studio

Visual Studio Code is a standalone source code editor i.e., integrated development environment (IDE) from Microsoft. It is used for different types of software developments such as websites, web pages, applications, analysis tools, etc, Features such as IntelliSense suggestions and syntax checking in the editor make it popular.

Visual Studio: https://visualstudio.microsoft.com/

Scikit-learn

Scikit-learn is the most helpful Open-Source and vigorous library based on NumPy, SciPy, and matplotlib for AI in python which gives proficient apparatuses to prescient data analysis, AI, and statistical modeling including classification, clustering, regression, and dimensionality reduction through a consistence interface in python.

Scikit-Learn: https://scikit-learn.org/stable/

Seaborn

Seaborn is a data visualization tool i.e., a library based on top of matplotlib, and is coordinated with Panda's information structures in python. It helps in the investigation and comprehension of information. One must be acquainted with NumPy and Matplotlib and pandas use seaborn.

TensorFlow

TensorFlow is an Open-Source stage for AI and ML libraries, utilizing information streams and charts to assemble models. It is predominantly utilized for Classification, Perception, Understanding, Discovering, Prediction and Creation. It permits designers to make huge Scale Neural networks with many layers with a complete, adaptable environment of apparatuses, libraries, and points of interaction.

TensorFlow: https://www.tensorflow.org/

Kaggle

Kaggle platform which allows users to find the published data sets offers a no-setup, customizable, Jupyter Notebooks environment. Users can build models in a web-based data science environment, AI engineers, information researchers, and enter contests to tackle information science challenges.

Kaggle: https://www.kaggle.com/

Chapter 5

Results and Discussion

A CNN model is designed and trained using Keras in Google Colab. The accuracy and precession are verified through testing the trained model using. A sample image of both covid positive and normal lung X-Ray image is passed and the prediction is verified. Confusion Matrix, Accuracy v/s Loss dataset graphs for both test and training data set, for better understanding and analysis of the model.

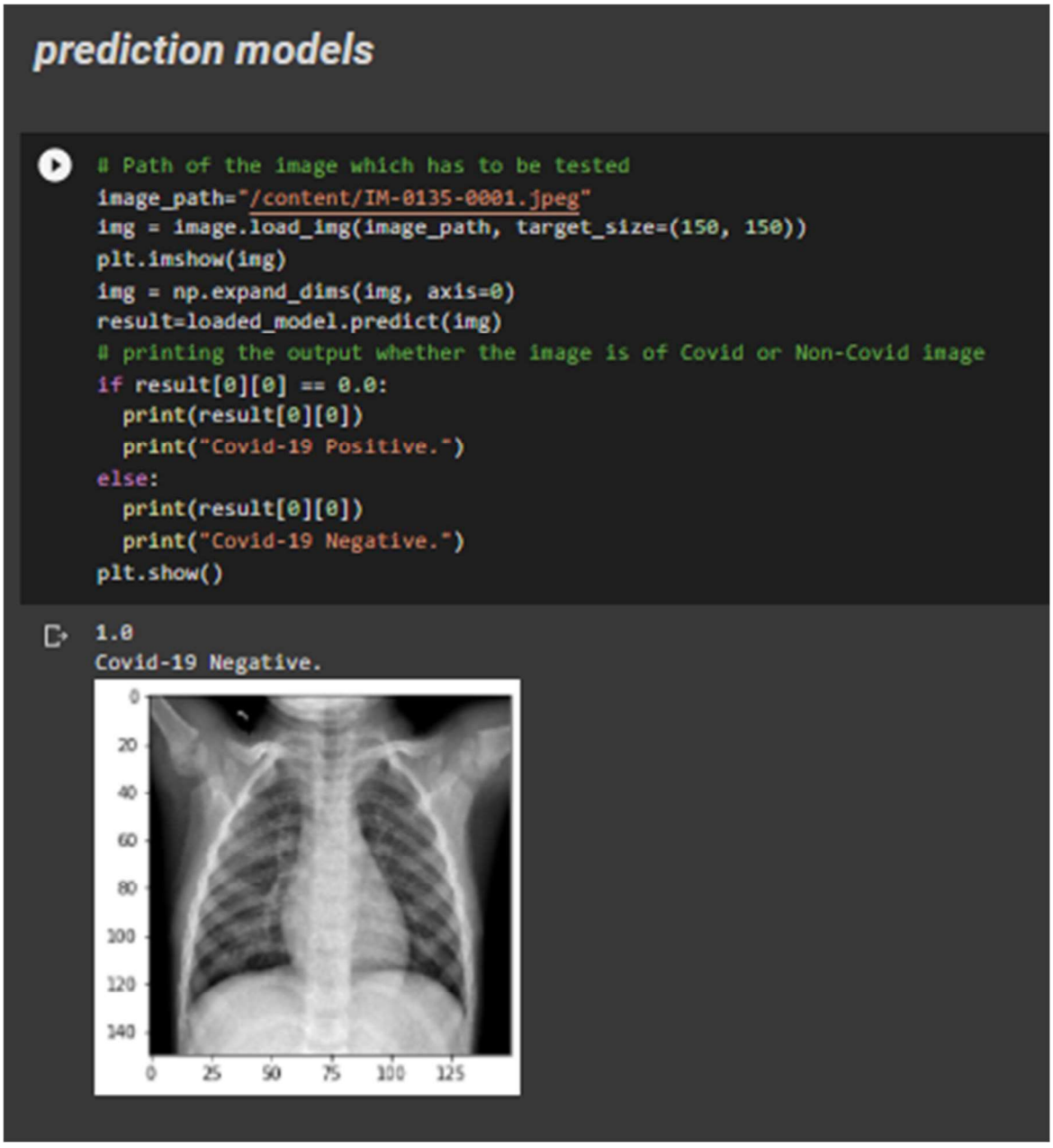

Fig 5.1 Interfacing and Predication of Image using CNN Model

The proposed CNN model is capable of classification or detecting COVID-19 X-Ray images and Normal lung X-Ray images when the trained model is supplied with the X-Ray image of the patients for predicting, which we can see in figure 5.1 where it is predicting the conduction of the given X-Ray lung image.

Varies parameters are calculated during training the model and testing, they are shown in the following report.

Data Classification:

Data Visualization

```
# dataset into "batches" to be supplied in batches
Org_train_dataset = train.flow_from_directory(Train_path,
                                              target_size = (150, 150),
                                              batch_size = 56)

Org_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.
```

Understanding the data in the dataset

```
# Displaying the different types of data Classes
Org_train_dataset.class_indices
```

```
{'Covid_img_train': 0, 'Normal_img_train': 1}
```

Fig 5.2 Data classification is done by the program

While pre-processing the dataset of around 562 X-Ray images, for testing and training the proposed CNN, the dataset was divided into 2 classes. The training dataset contains 242 COVID-19 positive X-Ray images and 242 Normal X-Ray images, making a total of 484 training images. The test dataset similarly, contained 39 COVID-19 positive and Normal images each, making a total of 78 test dataset images. We can see how the program is understanding the classification of training and test dataset given by use in figure 5.2.

```
Accuracy

# Evaluating the loss and Accuracy of the model
model.evaluate(test_generator)

9/9 [==============================] - 19s 2s/step - loss: 0.1271 - accuracy: 0.9669
[0.12714128196239471, 0.9669421315193176]
```

Fig 5.3 Models Loss and Accuracy

In figure 5.3 we see that the CNN model achieved execution with an accuracy of 96.69% with the test validation dataset, during the process of training the model and testing it with the validation dataset. We can also see that the loss of the model at end of the training has reduced to 0.127.

Confusion matrix:

Figure 5.4 shows the Confusion matrix of the model, according to which the CNN model tested for 78 images, out of which 2 have been predicted wrongly.

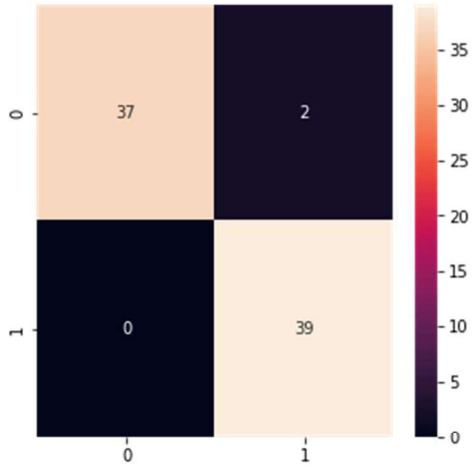

Fig 5.4 Confusion Matrix

The Y-axis represents the True classification and X-axis Predicated classification where 0 indicates Covid-19 Positive and 1 indicates Covid-19 Negative. As we can see the model was able to predict all the True Covid-19 negative samples correctly, were as coming to Covid-19 positive samples it predicted 37 cases correctly but gave 2 false predictions saying that Covid-19 positive cases as negative.

Figure 5.5 shows the Cases which have been predicted wrongly by the model. In the output, we can see the image identity numbers, which have been predicted wrongly.

Mismatch Image

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

     This are the images which have been mismatched: [2, 21]
```

Fig 5.5 Wrong Predictions (Mismatching)

Graphical Analysis:

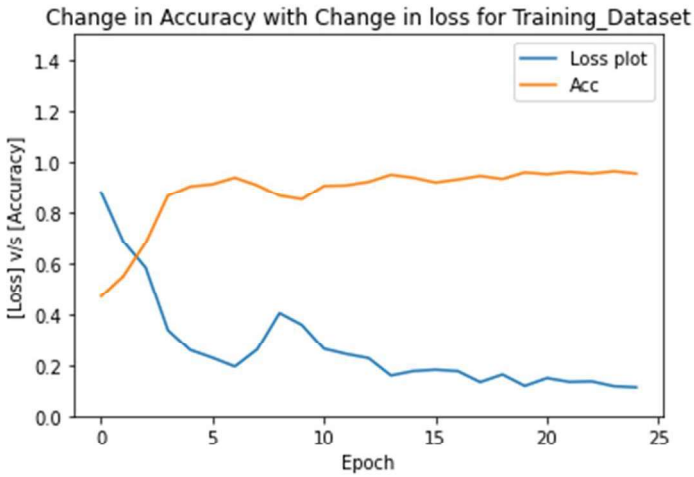

Fig 5.6 Accuracy V/S Loss Training Dataset

Figures 5.6, 5.7, 5.8, and 5.9 are Graphical Visualization of the curve of the accuracy and loss for training and test dataset and their interrelation, we can see how as the loss decreases the accuracy of the system is increasing respectively with each epoch, and also with the increase in an epoch the accuracy is also increasing respectively, after a certain epoch the accuracy tends to get saturated which can be observed in the graphs after 15 epochs.

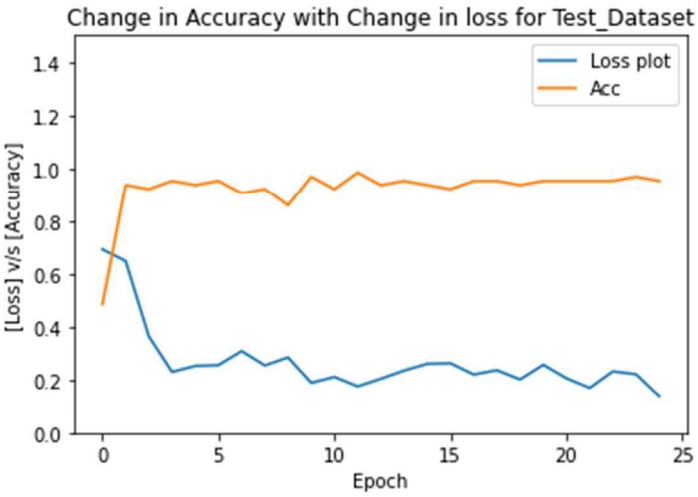

Fig 5.7 Accuracy V/S Loss Test Dataset

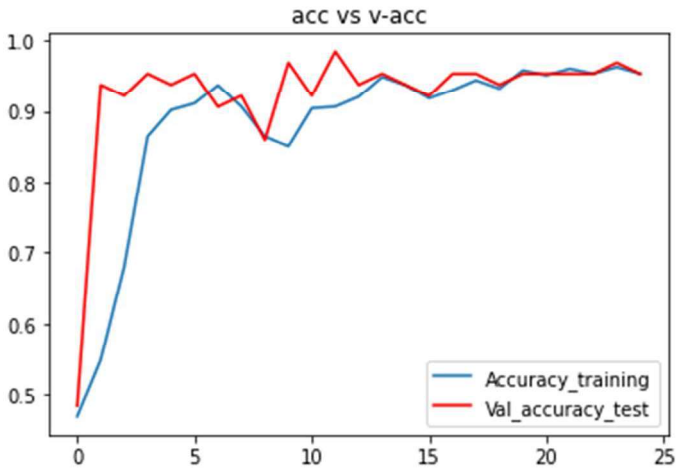

Fig 5.8 Accuracy of Training dataset V/S Test dataset

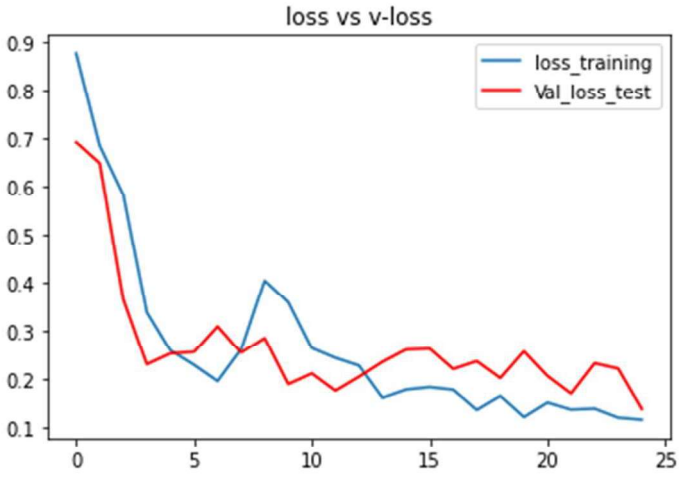

Fig 5.9 Accuracy of Test dataset V/S Test dataset

Data Sheet:

	precision	recall	f1-score	support
Covid_img_test	1.00	0.95	0.97	39
Normal_img_test	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

Fig 5.10 Data Sheet of Models Performance on Test Dataset

The above give report (figure 5.10) we can see shows the model is performing on the mass dataset of 78 pictures, out of which 39 are COVID-19 positive and 39 are negative. The model can anticipate the COVID-19 positive pictures at 100 percent accuracy yet when coming to the Normal or COVID-19 negative pictures the accuracy is just 95%. The general exactness of the model on the test dataset is around 97%.

Chapter 6

Conclusion and Future Scope

In a world overwhelmed by advanced innovation, Artificial intelligence based on top of Machine learning and Deep Learning assumes a conspicuous part in our lives. It has made an environment that joins numerous to give a brilliant and noteworthy exhibition in every task. Thus, in such a case using this type of rising technology in the medical field can save many lives. We don't need sophisticated, costly equipment to diagnose the patient.

In a pandemic like this in which we are leaving (2022), diagnosing the patient at the early stages of the problem or spreading of the virus present in one's body can be greatly helpful, and also the system that is been used should be able to adapt with ever-changing mutations from time to time. Thus, using Artificial intelligence built on top of Machine learning and Deep Learning can be used to solve these problems at a fast rate and high accuracy, thus in the process helping millions of lives in million ways, directly and as well as indirectly to save lives.

Hence, we conclude CNN models proposed, can determine significant bits of information based on data given information analysis, share the information on the cloud, and dissect it securely to give the particular output at a lightning speed, thus making a huge difference in how we tackle a problem and solve it to help millions.

This mini project is a simple demonstration of how we can implement this Deep Learning Convolutional Neural Network (CNN) using a few Open-Source libraries for analysis of lung X-Ray scans. This has a variety of applications like Ocular Disease Recognition using CNN, Heart rate analysis and diagnosis using CNN, Health monitoring using Smart device data analysis, Health care Automated monitoring system, and many more. This can be applied in other filed too where image analysis is required.

We can further implement this modeling scheme in other fields of medical application too, and also, we can improve the accuracy and loss rates. Other all the more remarkable CNN models, for example, ResNetv2 and ensemble of the multiple CNN models have not been assessed, however, they could work on the outcomes, along these lines we can use this model to do forecasts additionally and look at the model's presentation; Visualization can likewise be added to work on the understanding clarification of the CNN results since this is fundamental for the reception of a CNN based framework in our genuinely clinical application.

References

[1] "https://colab.research.google.com/?utm_source=scs-index", use this link to model, train, analysis, and visualize the datasets, Google Colab.

[2] "https://visualstudio.microsoft.com/", Refer to this link for installation and setup of Visual Studio.

[3] "**Coursera-Neural Networks and Deep Learning**", Refer the video of the to understand Deep Learning and CNN

[4] "https://www.kaggle.com/search?q=covid+19+image+dataset", the website from where we got the dataset of Covid-19 X-ray lung images and Normal X-ray lung images

[5] "**https://www.hindawi.com/journals/complexity/2021/6621607/**", Refer to the architecture and modeling.

[6] "https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53", visit the official website better understanding of CNN.

[7] "https://link.springer.com/article/10.1007/s12065-020-00540-3#:~:text=Researchers%20have%20successfully%20applied%20CNNs,%5D%2C%20blood%20cancer%2C%20anomalies%20of", visit the "Springer Link" website for the information about the application of CNN in medical image understanding

[8] "https://github.com/Jayanth9601/COVID-19_Detection_using_CNN", for the dataset and code used in this project, refer to the link given.

Appendix

Google Colab with Keras

Keywords

Here are some keywords that are used while modeling CNN using Keras.

ReLU – It is an activation function that is used after every convolution layer.

Sigmoid – It is an activation function that is used at the end of the model for Binary classification networks.

Convolution – The process of applying the filter to a matrix for activation of it.

Dense – It is a keyword used to create an ANN, this command auto-tunes the model of getting approximate weights, learning rate, forward propagation, and backward propagation.

flatten – The process of converting a Metrix into a column matrix to apply to the Dense ANN.

Accuracy – the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

Loss – Loss is the penalty for a bad prediction. That is, the loss is a number indicating how bad the model's prediction was on a single example.

Precision -- Precision is the one indicator of the machine learning model's performance, it refers to the number of true positives divided by the total number of positive predictions.

Code Review:

Code of Modelling, Training, Testing, Predicting and Visualizing dataset

```
"""***Covid-19 Detection Using CNN***"""
# numpy used to Data Handling
import numpy as np
# Used for Graph Generation
import matplotlib.pyplot as plt
# Used for file handling
import os
"""library for Deeplearning keras"""
import keras
# Used to creat model in concatination (one_layer_ofter_other)
from keras.models import Sequential
# All the needed tooles to creat model
from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense
# Used conver image into dataset
from keras.preprocessing.image import image, ImageDataGenerator
import sklearn.metrics as metrics
# Used to set the parameters and hyperparameters
from sklearn.metrics import classification_report, confusion_matrix
# Python data visualization library
import seaborn as sns
# Used to save and load the developed models
import tensorflow as tf

"""Path of the folders where the trainging and test dataset are saved"""
# loading the path were the training dataset is saved
Train_path = "< Location_of_the_train_dataset >"
# loading the path were the test dataset is saved
Test_path = "< Location_of_the_test_dataset >"

"""Mounting the Drive where the Image data has been saved"""
from google.colab import drive
drive.mount('/content/drive')

"""***Image and Data visualization***"""

"""***Image Visualization***"""
# Displaying a image form our dataset
# path of the location where the image is saved
path = "< Location_of_the_image_to_be_displayed (lung image) >"
# Path, target_size = {size at which the image is displayed}
img = image.load_img(path, target_size=(256,256))
# showing the image
```

```
plt.imshow(img)

"""***Image to data Conversition***"""
# Converting RGB image into Dataset (array) and scaling them down by (1/255)
# Taining dataset
train = ImageDataGenerator(rescale=1/255)
# Test dataset
test = ImageDataGenerator(rescale=1/255)

"""***Data Visualization***"""
# Seperating our dataset into "batches" to be supplied in groups
Org_train_dataset = train.flow_from_directory(Train_path,
                                target_size = (150, 150),
                                batch_size = 32)

Org_test_dataset = test.flow_from_directory(Test_path,
                                target_size= (150, 150),
                                batch_size = 32 )

"""Understanding the data in the dataset"""
# Displaying the different types of data
Org_train_dataset.class_indices
# Visualizing the difference between the data
Org_train_dataset.classes

"""***CNN Basic Model in Keras***"""

"""Creating a CNN model"""
# Creating a model named "model" which will be sequential
model = Sequential()
# Input layer (Raw Image)
# First layer (Input layer), 1st Convolution layer with 32 filters of size 3*3, relu activation
model.add(Conv2D(32,kernel_size=(3,3), activation='relu',
            input_shape=(150,150,3)))

# Second layer of Convolution
# 2nd layer, 2st Convolution layer with 64 filters of size 3*3, relu activation
model.add(Conv2D(64,(3,3), activation='relu'))
# Pooldown layer of size 2*2
model.add(MaxPool2D(2,2))
# fitting the model
model.add(Dropout(0.25))

# Third layer of Convolution
# 3rd layer, 3rd Convolution layer with 128 filters of size 3*3, relu activation
model.add(Conv2D(128,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
```

```
model.add(Dropout(0.25))

# Fourth layer of Convolution
# 4rd layer, 4rd Convolution layer with 256 filters of size 3*3, relu activation
model.add(Conv2D(256,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))


"""Data Flattening"""
#flattening the data into Column matrix
# Converting the matrix into column matrix
model.add(Flatten())
# Dense layer with relu activation
model.add(Dense(256,activation='relu'))
model.add(Dense(128,activation='relu'))
# fitting the model
model.add(Dropout(0.5))
# Final output layer with sigmoid activation "Binary Classification"
model.add(Dense(1,activation='sigmoid'))

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

# Getting the Summary of the designed model with parameter of respective layers
model.summary()

"""Flowchart"""
# Getting the Flow chart of the designed model
from keras.utils.vis_utils import plot_model
plot_model(model)

"""***Training the Model***"""

#Train from scratch
train_datagen = image.ImageDataGenerator(
   rescale = 1./255,
   shear_range = 0.2,
   zoom_range = 0.2,
   horizontal_flip = True,
)

test_dataset = image.ImageDataGenerator(rescale=1./255)

# Conveting RGB images into dataset in batch of 56 images at a time
```

```
train_generator = train_datagen.flow_from_directory(
   Train_path,
   target_size=(150,150),
   batch_size=56,
   class_mode='binary',
)

# Conveting RGB images into dataset in batch of 32 images at a time
test_generator = test_dataset.flow_from_directory(
   Test_path,
   target_size=(150,150),
   batch_size=32,
   class_mode='binary'
)

hist = model.fit(
   train_generator,
   # Total number of steps (batches of samples) to yield from generator
   steps_per_epoch=8,
   # an instant in time chosen as the origin of a particular calendar era
   epochs=15,
   # test the test datset
   validation_data = test_generator,
   validation_steps=2,
)

"""**Accuracy**"""
# Evaluating the loss and Accuracy of the model
model.evaluate(test_generator)

"""***Saving the generated model***"""
# Creating a folder to save the model generated
model.save('< Location_where_the_train_model_to_be_save_in_".h5"_formate >')
print("model Save with name <model.h5>.")

"""***Graphical Representation of Ouput***"""
# Used to Generat the graph
h = hist.history
# Checking different parametes that we obtained after training the model
h.keys()

"""Accuracy Graph With respect to no of iterations"""
# Grpah of the Accuracy of Training dataset v/s Test dataset
plt.plot(h['accuracy'], label="Accuracy_training")
plt.plot(h['val_accuracy'] , c = "red", label="Val_accuracy_test")
plt.title("acc vs v-acc")
plt.legend()
```

```
plt.show()

"""Loss Graph With respect to no of iterations"""
# Grpah of the Loss of Training dataset v/s Test dataset
plt.plot(h['loss'],label="loss_training")
plt.plot(h['val_loss'] , c = "red",label="Val_loss_test")
plt.title("loss vs v-loss")
plt.legend()
plt.show()

"""***Accuracy V/S Loss Graph***"""
#Graph of Accuracy v/s loss of Training dataset
epochs=15
plt.figure()
plt.title("Change in Accuracy with Change in loss for Training_Dataset")
plt.xlabel('Epoch')
plt.ylabel('[Loss] v/s [Accuracy]')
plt.plot([i for i in range(epochs)], h['loss'], label='Loss plot')
plt.plot([i for i in range(epochs)], h['accuracy'], label = 'Acc')
plt.legend()
plt.ylim([0,1.5])

#Graph of Accuracy v/s loss of Test dataset
plt.figure()
plt.title("Change in Accuracy with Change in loss for Test_Dataset")
plt.xlabel('Epoch')
plt.ylabel('[Loss] v/s [Accuracy]')
plt.plot([i for i in range(15)], h['val_loss'], label='Loss plot')
plt.plot([i for i in range(15)], h['val_accuracy'], label = 'Acc')
plt.legend()
plt.ylim([0,1.5])

"""Output checking"""

"""**Loading the saved Model**"""
# Loading the saved model from memory
loaded_model =
tf.keras.models.load_model('/content/saved_model_Moredata/covid.h5')
#printing the size of the input layer of the model
loaded_model.layers[0].input_shape

"""***prediction models***"""
# Path of the image which has to be tested
image_path = "< Location_of_the_Image_which_has_to_pridicted >"
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)
# printing the output whether the image is of Covid or Non-Covid image
if result[0][0] == 0.0:
  print("Covid-19 Positive.")
else:
  print(result[0][0])
  print("Covid-19 Negative.")
plt.show()

"""***Model Visulization of Test Dataset***"""
# Predicated output values from the image
predicted_classes = []
positive = "< Location_of_the_test_COVID_positive_dataset >"
normal = "< Location_of_the_test_COVID_Negative_dataset >"
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]))

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

# Orginal output values
true_classes = Org_test_dataset.classes
true_classes

"""***Mismatch Image***"""
# The mismatched images
mismatch = []
for i in range(len(predicted_classes)):
  if predicted_classes[i] != true_classes[i]:
    mismatch.append(i+1)
print("This are the images which have been mismatched:",mismatch)

"""***Report***"""
class_labels = list(Org_test_dataset.class_indices.keys())

# Report of the tested dataset analysis
report = metrics.classification_report(true_classes, predicted_classes,
target_names=class_labels)
print(report)
```

```
"""***Confusion Matrix***"""
conf_matrix = confusion_matrix(true_classes, predicted_classes)
plt.figure(figsize = (5,5))
sns.heatmap(conf_matrix, annot=True)

"""***Thank You***"""
```

Code for Data segregation

```
# Used to Read data from CSV files
import pandas as pd
# Used to handle files and DIR
import os
# Used to copy the Images
import shutil

# Create the data for positive samples
# Path where Metadata CSV file is saved
File_path = "< Location_of_CSV_file >"
# Path were the Images are saved
Images_path = "< Location_where_the_images_are_saved >"

# Reading the Matadata CSV file for Processing
df = pd.read_csv(File_path)
# Printing number of Rows and colums in the metadata table
print(df.shape)

# WE CAN CHECK THE CSV FILE USING THE COMMAND
# "df.head()"
 # Path (folder) where the Images will be saved
Target_DIR = "< Location_where_the_image_data_should_be_saved >"

# Checking if the file were the images are to be copyed exists
if not os.path.exists(Target_DIR):
    # If doesn't exist Creating one
    os.mkdir(Target_DIR)
    print("Covid folder created.")

co = 0

for (i,row) in df.iterrows():
    # For matadata selecting only covide PA images
    if row["finding"] == "COVID-19" and row["view"] == "PA":
```

```
        # Copying filename
        filename = row["filename"]
        # Copying the Path were Image is saved
        image_path = os.path.join(Images_path,filename)
        # Making the Path were the Image has to be Copyed
        image_copy_path = os.path.join(Target_DIR,filename)
        # Copying the Image to the Desired location (path_in_previous_step)
        shutil.copy2(image_path,image_copy_path)
        # Counting the number of Images being copyed
        co += 1

print("Copied",co," to DIR.")
```