COVID-19 DETECTION FROM CHEST X-RAY IMAGES USING CNN

Report Submitted to

National Institute of Technology Manipur

In partial fullfillment for the award of the degree

of

Bachelor of Technology
in Computer Science & Engineering

by

Sheikh Dilwar Komol
18103042

Under The Supervision Of
Mr. Sanabam Bineshwor Singh

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR

IMPHAL 795004

May – 2022

COVID-19 DETECTION FROM CHEST X-RAY IMAGES USING CNN

Report Submitted to

National Institute of Technology Manipur

In partial fullfillment for the award of the degree

of

Bachelor of Technology
in Computer Science & Engineering

by

Sheikh Dilwar Komol
18103042

Under The Supervision Of
Mr. Sanabam Bineshwor Singh

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR

IMPHAL 795004

May – 2022

DECLARATION

I, the undersigned solemnly declare that the project report entitled **“COVID-19 Detection From Chest X-ray Images Using CNN”** is based on my own work carried out during the course of my study under the supervision of **Mr. Sanabam Bineshwor Singh, Lecturer, Department of Computer Science and Engineering NIT Manipur**

I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that:-
I. The work contained in the report is original and has been done by me under the general supervision of my supervisor **Sir Sanabam Bineshwor Singh**.
II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this Institution or any other Institution.
III.I have followed the guidelines provided by the university in writing the report.
IV. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them in the text of the report and giving their details in the references.

Signature of the student
Sheikh Dilwar Komol
(18103042)

राष्ट्रीय प्रौद्योगिकी संस्थान,मणिपुर
NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR
Langol, Imphal – 795 004, Ph. (0385)2445812
E-mail:director@nitmanipur.ac.in, Website : www.nitmanipur.ac.in
(An Autonomous Institute under MHRD, Govt. of India)

Ref. No.NITM.3/98/ACAD/CSEDEPT/2022/B.Tech-16 **Dated:________**

Certificate

This is to certify that the Dissertation ***Report*** *entitled,* ***"COVID-19 detection from Chest X-ray images using CNN "*** *submitted by Mr. Sheikh Dilwar Komol to National Institute of Technology Manipur, India, is a record of bonafide Project work carried out by him under the supervision of* ***Mr. Sanabam Bineshwor Singh, Lecturer of NIT Manipu****r under the department of* ***Computer Science & Engineering****, NIT Manipur and is worthy of consideration for the award of the degree of Bachelor of Technology in* ***Computer Science & Engineering Department*** *of the Institute.*

Dr. Khundrakpam Johnson Singh
Head of Department
CSE Department

Mr. Sanabam Bineshwor Singh
Supervisor
CSE Department

ACKNOWLEDGEMENT

The satisfaction and euphoria that has accompanied successful completion of my project work would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned my effort with success.

It is my great pleasure to express my sincere gratitude and respect to my guide **Mr *Sanabam Bineshwor Singh* , Lecturer, Department of Computer Science and Engineering NIT Manipur**, for his valuable suggestions and expert guidance in completing this project work.

I am indebted to **Dr. Khundrakpam Johnson Singh**, Head of Department, Computer Science and Engineering and to my beloved Director, **Prof. (Dr.) Goutam Sutradhar** for providing me the required facilities at the institute.

I would also like to thank all the faculty and staff of Computer Science and Engineering Department, & well-wishers for their valuable help rendered during the course of my project.

ABSTRACT

Covid'19 global pandemic affects health care and lifestyle worldwide, and its early detection is critical to control cases' spreading and mortality. The actual leader diagnosis test is the Reverse transcription Polymerase chain reaction (RT-PCR), result times and cost of these tests are high, so other fast and accessible diagnostic tools are needed. Inspired by recent research that correlates the presence of COVID-19 to findings in Chest X-ray images, this paper's approach uses existing deep learning models to process these images and classify them as positive or negative for COVID-19. The proposed system aims to achieve a detection accuracy of Covid'19 around 97% .

LIST OF FIGURES AND TABLES

CONTENT

CHAPTER 1
INTRODUCTION

This chapter provides a brief introduction to Covid-19, CNN, DL, Objective of the project and Literature Reviews.

1.1. Introduction

Coronavirus illness is a disease that comes from Severe Acute Respiratory Syndrome (SARS) and Middle East Respiratory Syndrome (MERS). A novel coronavirus, COVID-19, is the infection caused by SARS-CoV-2. In December 2019, the first COVID-19 cases were reported in Wuhan city, Hubei province, China.

Reverse transcription Polymerase chain reaction (RT-PCR),RAT (Rapid Antigent Test) are now the main methods for COVID19 detection.

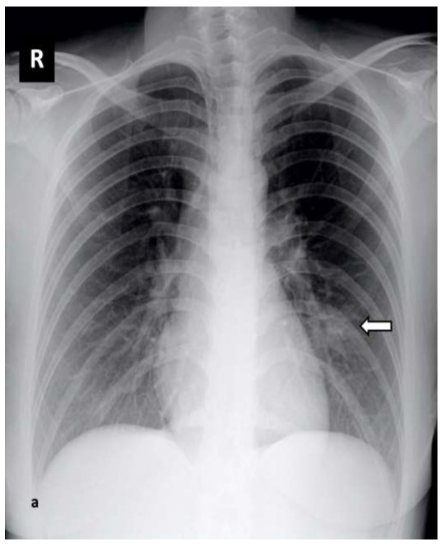
Recently, medical images such as chest X-ray and computed tomography (CT) scan images have been used to determine COVID-19 positive cases . CT scan imaging is a costly procedure while chest X-ray is cheaper, faster and widely-used way to generate 2D images of the patients that can potentially be used for COVID-19 patients as well.

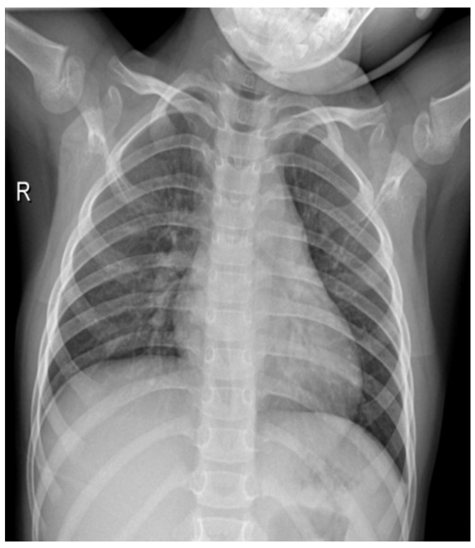
This paper presents a new approach using existing Deep Learning models. It focuses on enhancing the preprocessing stage to obtain accurate and reliable results classifying COVID-19 from Chest X-ray images. The preprocessing step involves a network to filter the images based on the projection it is (lateral or frontal), some common operations such as normalization, standardization, and resizing to reduce data variability, which may hurt the performance of the classification models, and a segmentation model to extract the lung region which contains the relevant information, and discard the information of the surroundings that can produce misleading results . Following the preprocessing stage comes the classification model. Finally, the visualization of heatmaps for different images provides helpful information about the regions of the images that contribute to the prediction of the network, which in ideal conditions should focus on the appearance of the lungs, backing the importance of lung segmentation in the preprocessing stage.

In this study ,a convolutional neural network (CNN) is proposed to investigate chest X-ray images and identify COVID-19 patients in early stage more precisely with higher specificity, that may aid public health systems to reduce the local community transmission rate.

This project is focussing to deal with classifying 2 types of **chest x-ray images** namely, **covid'19 positive and normal**

.





(a) covid'19 +ve image (b)normal image

Fig 1:- chest x-ray images (a) covid'19 +ve (b) normal

Using Deep Learning, an image classifier is build based on a convolutional neural network to classify these 2 different types of chest X-ray images dataset collected from Kaggle and github repository that are publicly available.

1.2 CNN(Convolution Neural Network):

CNN is a type of ANN(Artificial Neaural Network) used in image recognition and processing that is specifically designed to process pixel data. CNN uses a system much like multilayer perceptron designed for reduced processing requirements such as input layer, an output layer and a hidden layer- (includes multiple convolution layers, pooling layers, fully connected layers and normalization layers.

1.3 DL(Deep learning):

Deep learning models introduce an extremely sophisticated approach to machine learning and are set to tackle these challenges because they've been specifically modeled after the human brain. Complex, multi-layered “deep neural networks” are built to allow data to be passed between nodes (like neurons) in highly connected ways. The result is a non-linear transformation of the data that is increasingly abstract

1.4 OBJECTIVE:

The goal is to propose a deep learning model using **CNN** to detect **COVID-19 positive cases** more precisely utilizing 2 types of **chest X-ray images - covid’19 positive and normal .**

1.5 LITERATURE REVIEW:

Wang et al. (2020) generated a large benchmark dataset with 13,975 chest X-ray images called COVIDx and investigated them using deep learning model that showed 93.30% accuracy.

Abbas et al. (2020) proposed DeTraC deep CNN model that gave solution by transferring knowledge from generic object recognition to domain-specific tasks. Their algorithm showed 95.55% accuracy (specificity of 91.87%, and a precision of 93.36%).

Apostolopoulos & Mpesiana (2020) implemented transfer learning using CNNs into a small medical image dataset (1427 X-ray images), which provided highest 96.78% accuracy, 98.66% sensitivity, and 96.46% specificity respectively.

Karar et al. (2020) proposed a deep CNN architecture called COVIDX-Net that investigated 50 chest X-ray images with 25 COVID-19 cases and provided 90% accuracy and 91% F-score.

Minaee et al. (2020) proposed a deep learning framework based on 5000 images named COVID-Xray-5k where they applied ResNet18, ResNet50, SqueezeNet and Densenet-121 into them and produced sensitivity 97.5% and specificity 90% on average.

Heidari et al. (2020) used transfer learning based VGG16 model into chest x-ray images which showed 94.5% accuracy, 98.4% sensitivity and 98% specificity. Again,

Khan et al. (2020) represented a deep neural network based on Xception named CoroNet that provided 89.6% accuracy for four class and 95% accuracy for three class images

This Project is focusing mainly upon classifying 2 types of chest X-ray images which are

i)Covid'19 chest X-ray images dataset for positive cases.

ii) chest images dataset for negative/normal cases.

1.6 Motivation

RT-PCR testing is the medical standard to identify COVID-19. However, might take hours to receive results. With the growing number of cases walking into hospitals, alternatively, chest X-ray is used as initial element to review the clinical situation of a patient. If the X-ray shows any pathological findings, patients are admitted for further diagnosis. If the X-Ray is normal, patients are requested to go home and wait for PCR test results..
Early sypmtoms detection with chest X-ray images will reduce mass community spread of this Covid-19 diseases.

1.7 Advantages and Disadvantages

Advantagases- RT-PCR – Costly and Time Consuming.
Disadvantages - Less Dataset available so limitation in showing high Accuracy results . CNN has great prospects in detecting COVID-19 with very limited time, resources, and costs. Though the proposed model shows promising results, it is not clinically tested till now**.**

CHAPTER 2

Theoretical Study

2.1 Proposed Algorithm

The working methodology has been used to detect COVID-19 patients from the publicly available datasets. Figure 2.1 illustrates these steps that can be split into different sections such as data collection, pre-processing, classification with different layers such as input layers, ReLU layer,Max pooling layer, Dropoout layers. This approach is described briefly as follows :

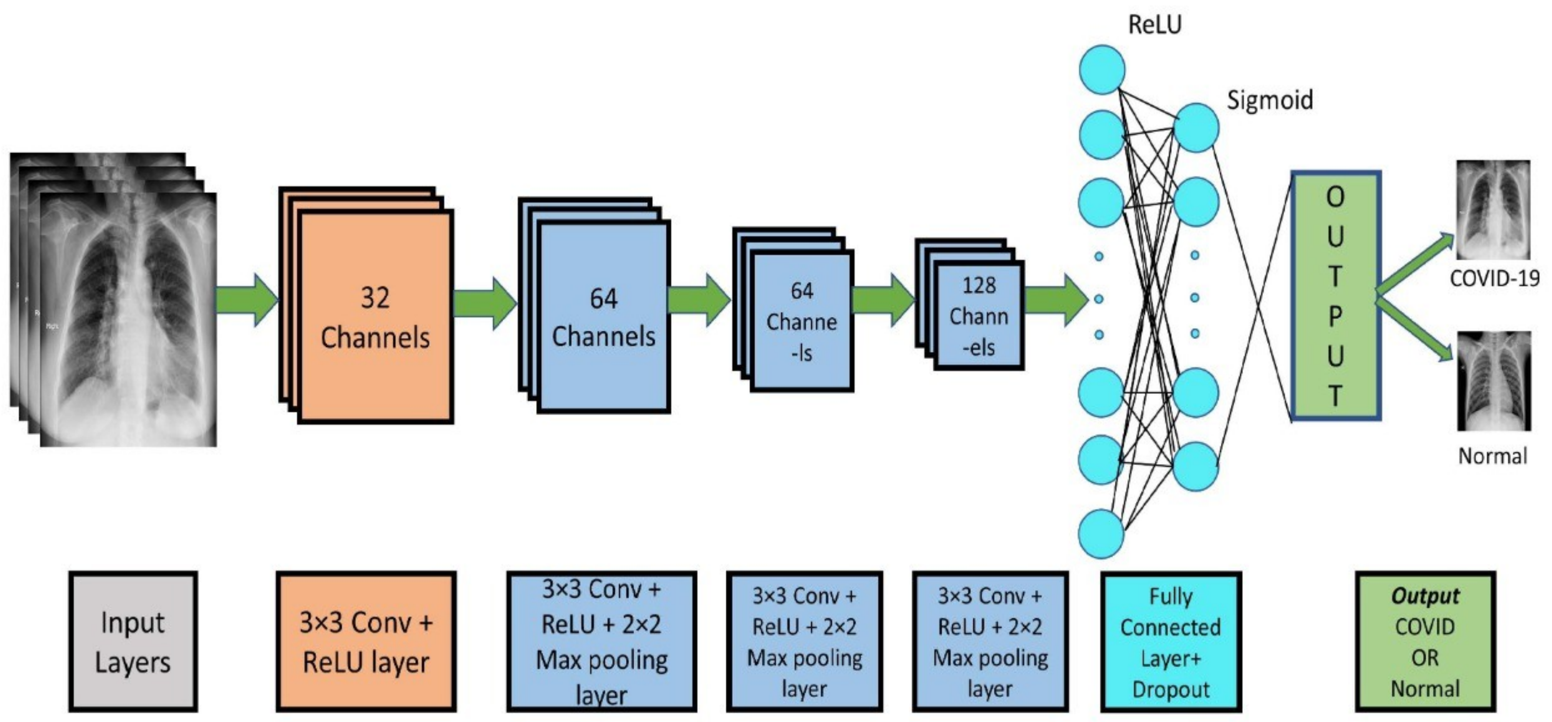
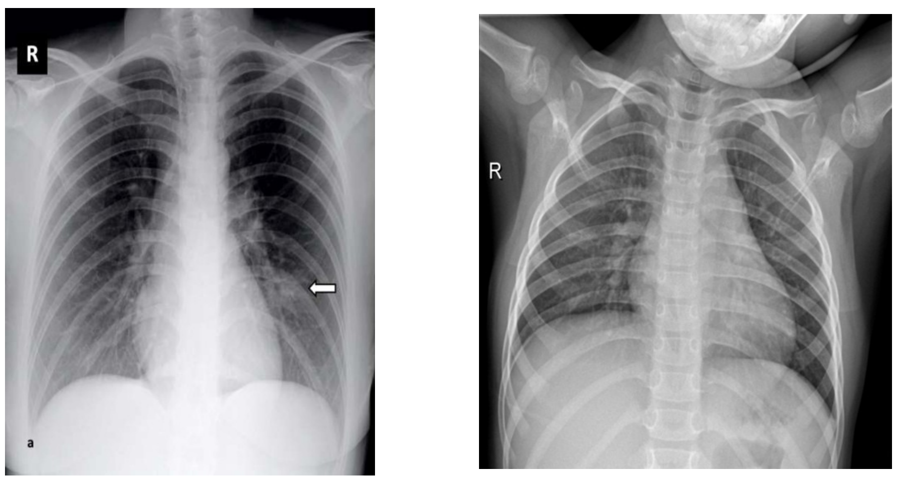

ReLU
Sigmoid
32 Channels
64 Channels
64 Channe -ls
128 Chann -els
OUTPUT
COVID-19
Normal
Input Layers
3×3 Conv + ReLU layer
3×3 Conv + ReLU + 2×2 Max pooling layer
3×3 Conv + ReLU + 2×2 Max pooling layer
3×3 Conv + ReLU + 2×2 Max pooling layer
Fully Connected Layer+ Dropout
Output COVID OR Normal

Figure 2.1: A schematic diagram of overall workflow

2.1.1 Data Collection:

Firstly the datasets were extracted (downloaded from kaggle and github repository).

Figure 2.1 illustrates the two types of chest X-ray images.



covid’19 +ve image normal image

Figure 2.2: Chest X-ray images of (a) COVID-19 and (b) normal

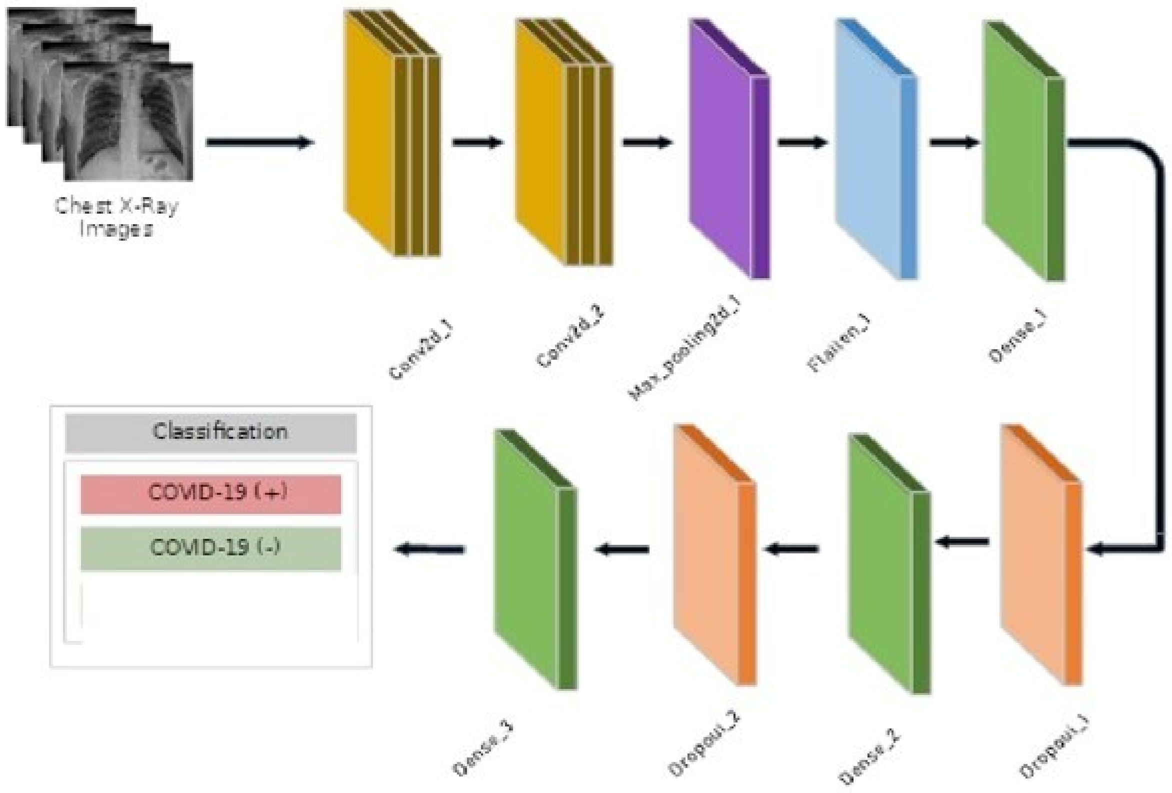


Fig 2.3:- Proposed method of CNN Layers.

2.1.2 Data pre-processing :

The first process is to downsize the original image into same dimensions (500,550) pixels in order to decrease dimensionality, computations and help the network to show a better performance in lower time and more straightforward calculation .

In this step, training dataset are normalized into grayscale images. Then, all baseline classifiers have been implemented with transformed dataset respectively. But, pre-trained CNN models such as VGG16, ResNet50, InceptionV3 cannot support grayscale images, hence we directly employed them into primary dataset .

2.2Proposed Convolutional Neural Networks:

2.2.1 CNN Model Implementation:

Convolutional Neural Networks (CNN) is a special class of artificial neural network (ANN) that manipulates an input layer along with the sequence of hidden and output layers. It maintains a sparse connection between layers and weights that shares them with output neurons in the hidden layers. Like regular ANN, CNN contains a sequence of hidden layers, which are denoted as convolutional and polling layer .

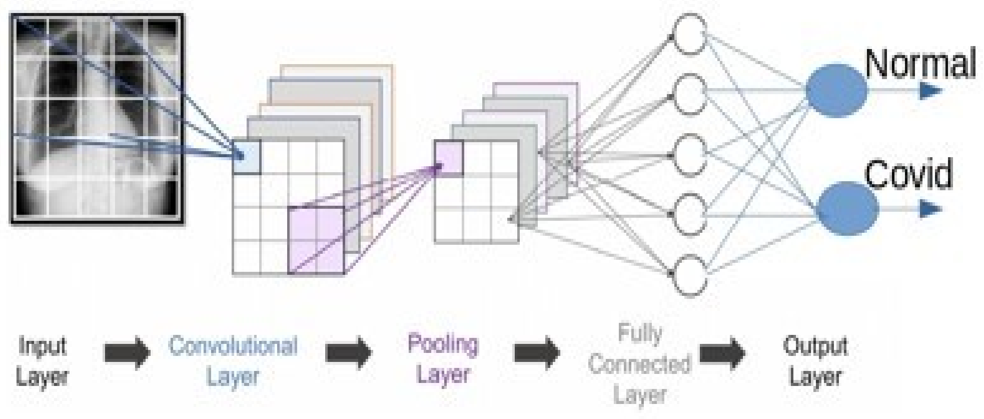


Fig2.4 : CNN Model

2.2.2 Convolutional Layer:

Convolution layer is the core structure of a CNN that manipulates convolution operation (represented by) instead of ∗ general matrix multiplication. This layer accomplishes most of the computations of CNN model. The count of filters, size of local region, stride, and padding are mentioned as hyper parameters of this layer. Convolution layers extract and learn about features using these filters. Hence, it is known as the feature. extraction layer.

2.2.3 Pooling Layer:

In CNN, the sequence of convolution layer is followed by an optional pooling or down sampling layer to lessen the volume of input images and number of parameters..

2.2.4 Padding Layer :

Padding layers in a CNN are added to an image when it is being processed by the kernel of the CNN.The padding layer works by extending the area of which a convolutional neural network processesan image. 7 The kernel is the neural networks filter which moves across the image, scanning each pixel and converting the data into a smaller, or sometimes larger, format. In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding padding to an image processed by a CNN allows for more accurate analysis of images.

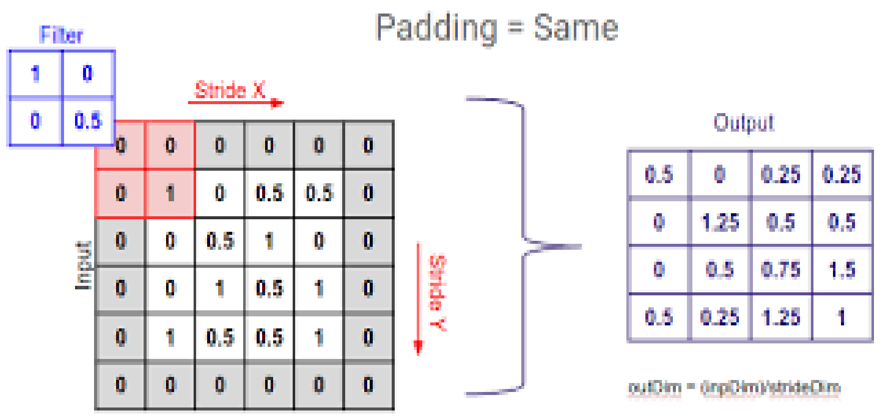


Fig 2..5: *Padding layer*

2.2.5 Flatten layer:

After implementing the pooling layer, a flatten layer has been employed to flat the entire network. It converts the entire pooled feature map matrix into a single column.

2.2.6 Dense layer:

Then, we have implemented three dense layers which are also known as a fully connected layer. In this layer, the input of previous layers is flattened from a matrix into a vector and forwarded it to this layer like a neural network .

2.2.7 Dropout layer:

When a large feed-forward neural network is investigated with a small 205 training set, it usually shows poor performance on held-out test data, and dropout is a useful procedure to mitigate this problem. In our model, we used dropout layer after each dense layer and to reduce over-fitting by preventing complex co-adaptations on the training data.

2.2.8 Fully connected Layer :

The Fully connected input layer collects the outputs of the previous layers, and convert or "flattens" them into a single vector ,each representing a probability that a certain feature belongs to a label or class.The main goal of a fully connected layer is to take the results of the previous layers in the network and use them to classify the image

2.2.9 Baseline classifiers:

Several machine learning classifiers have been used to perform comparative performance assessements, such as support vector machine (SVM), random forest (RF), k-nearest neighbor (k-NN), logistic regression(LR), gaussian na¨ıve bayes (GNB), Bernoulli na¨ıve bayes (BNB), decision tree (DT), Xgboost (XGB), multilayer perceptron (MLP), nearest centroid (NC) and perceptron.

2.2.10 Evaluation:

The performance of individual classifiers was assessed by different evaluation metrics such as accuracy, AUC, F-measure, sensitivity, and specificity, respectively.

2.2.11 ReLU:

The Rectified Linear Unit a commonly used activation function in deep learning. This function returns 0 if it receives any negative input, but for any positive value xx it returns that value back. So it can be written as f(x)=max(0,x)f(x)=max(0,x)

Graphically it can be presented as :

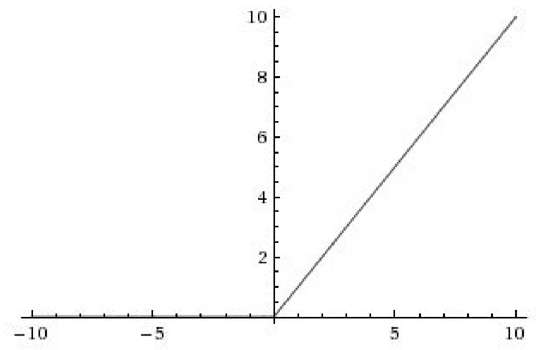


Fig 2.6: Relu function

Relu can allow our model to account for non-linearities and interactions so.It also works great in most applications, so it is widely used in many CNN models.

2.3 Training Phase:

After making a CNN model,the next phase is to train the model with some data ,usually the dataset is splited into two parts 60% for training and 40% for validation.after splitting the dataset ,it is feeded to the Convolutional Neutral Network for training.

2.4 Testing Phase:

After the training phase, the trained model is ready to be tested with unlabeled data. Initially the unlabeled dataset is preprocessed before feeding it to the CNN similar to the training set .

2.5 Hardware and Software Requirements:

1. Windows 10 Pro
2. Python 3.10
3. Processor- Intel(R) Core(TM) i5 CPU M 520 @ 2.40GHz 2.40 GHz
5. RAM_ 4.00 GB (3.86 GB usable)
6. System Type- 64-bit operating system, x64-based processor
7. Ultra Office
8. Google Colab

Cost The cost of my project is 0.00 rupees,as the data and tools needed for the creation of the models are free to download from the internet.

Implemention of this project is done in Python 3.10 and code is running on google colab , so along with python additionally the following python libraries are imported :

2.5.1 Tensorflow

Tensorflow is an open source software library for numerical computation. First the nodes of the computation graph are defined, and then inside a session the 13 actual computations take place. Tensorflow is widely used in Machine Learning, to implement the CNN.

2.5.2 Keras

Keras is a high-level neural networks library written in python that works as a wrapper to Tensorflow. It is used in cases to quickly build and test the neural network with minimal lines of code. It contains implementation of commonly used neural network elements like layers, Objective, activation functions, optimizers, and tools to make working with image and text data easier

2.5.3 Numpy:

Numpy known as numerical python is used to perform numerical operation.

2.5.4 Matplotlib:

Packages to show images or to display images or any kind of plot . It works for visualization.

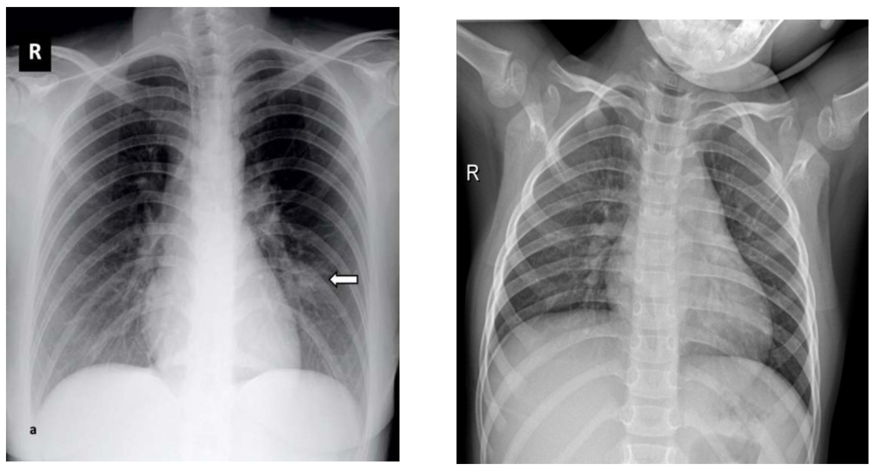
CHAPTER 3

Experimental Study

This Chapter with the experimental study which includes collection of data, preprocessing, implementation of CNN , training , testing and tools required.

3.1. Data Collection

Firstly the dataset of chest X-ray images that was downloaded from kaggle and github repository were extracted. It contains 274 normal and 298_ covid'19 positive patients chest x-ray iamges.In total , 1044 chest x-ray images are used for training and validation processes i,e 572 each for validation and training.



covid'19 +ve image normal image

Fig 3.1: Chest X-ray images of (a) COVID-19 and (b) normal

3.2 Data Preprocessing:

Downloaded dataset are resized into same size for preprocessing .These datasets are originally in the format .pnp, .jpg then using a python script converter, these .png, jpg files are converted into .jpeg files so that the model can interpret the dataset.

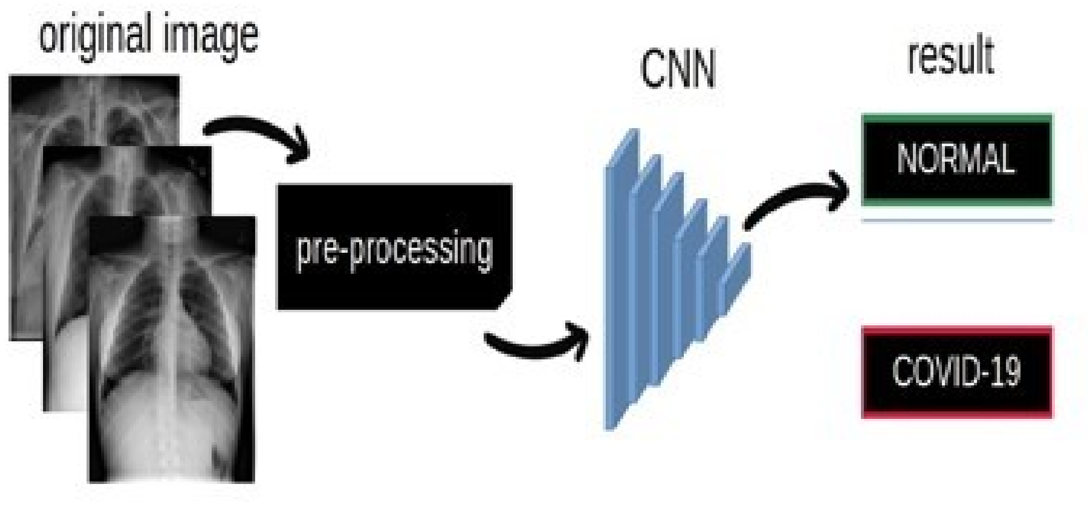
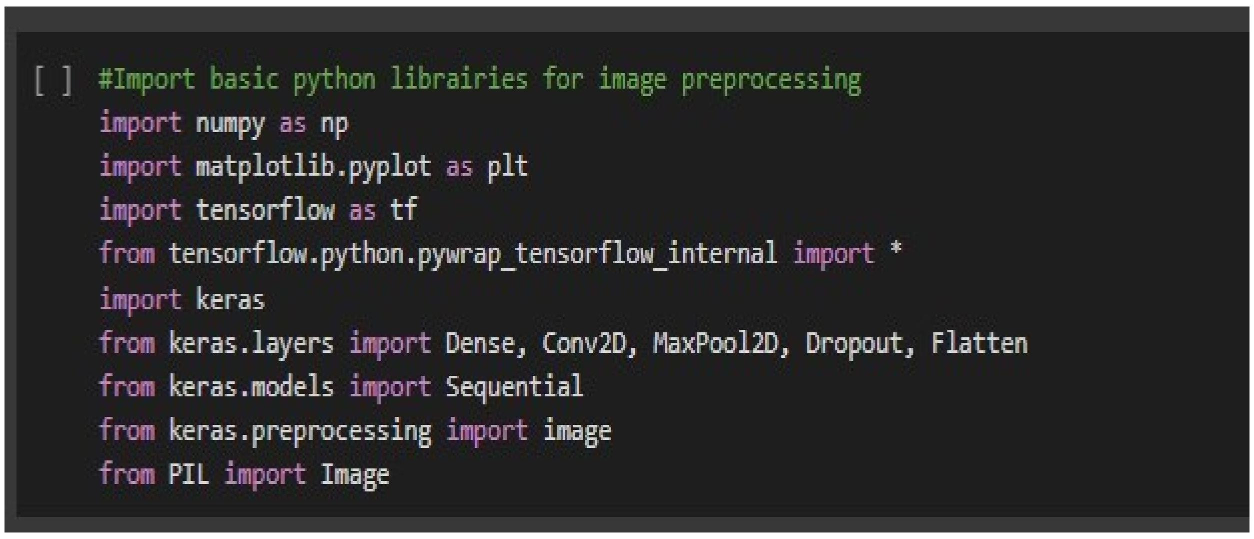


Figure 3.2 : proposed CNN Model

3.2.1 Image Preprocessing :

Google Colab is used for implementing the multiclass classifier.

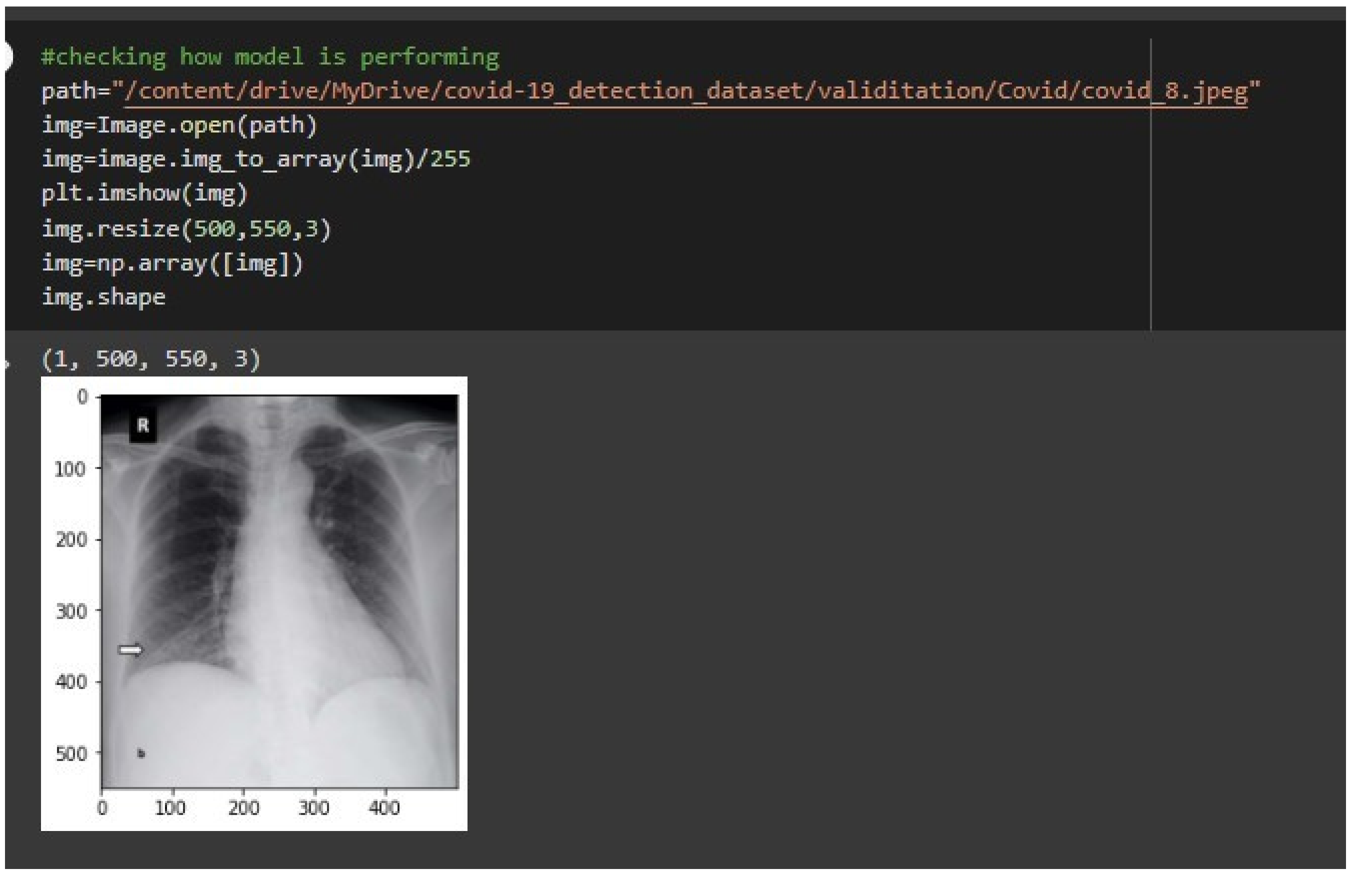
Then required python librairies are imported for image preprocessing as shown below:



```
[ ] #Import basic python librairies for image preprocessing
    import numpy as np
    import matplotlib.pyplot as plt
    import tensorflow as tf
    from tensorflow.python.pywrap_tensorflow_internal import *
    import keras
    from keras.layers import Dense, Conv2D, MaxPool2D, Dropout, Flatten
    from keras.models import Sequential
    from keras.preprocessing import image
    from PIL import Image
```

fig 3.3: codes for image preprocessing

The image dataset is converted into a taget size of (500,550,3) as shown below:-



```
#checking how model is performing
path="/content/drive/MyDrive/covid-19_detection_dataset/validitation/Covid/covid_8.jpeg"
img=Image.open(path)
img=image.img_to_array(img)/255
plt.imshow(img)
img.resize(500,550,3)
img=np.array([img])
img.shape
```

```
(1, 500, 550, 3)
```

fig 3.4: preprocessed image

3.3 Implementation of CNN Model:

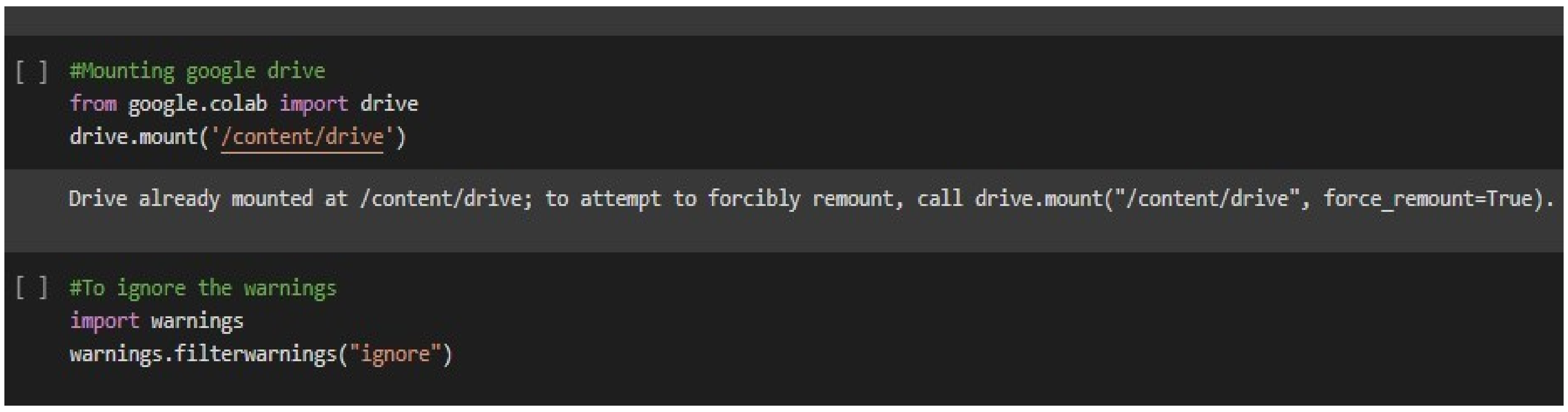
A Convolution Neural Network has mainly input layer, convolution layer, pooling layer, Rectified Linear Unit (ReLU) layer and fully connected layer. In convolution layer, the given input image is separated into various regions.

In ReLU layer element wise activation function is carried out. The Pooling layer is an optional layer. However the pooling layer's role is mainly for down sampling. In the final layer also called Fully connected layer is used to generate the class or label score value based on the probability in between 0 to 1.

The CNN based covid-19 chest x-ray iamges classification is divided into two phases such as training and testing phases. The number of images is divided into different category by using labels for 0 and 1 for covid and normal respectively . In the training phase, preprocessing,and classification with Loss function is performed to make a prediction model. Initially, training image set is labeled.and in preprocessing ,image resizing is applied to change size of the image.

Finally, the CNN is used for covid-19 chest x-ray iamges classification. The loss function is calculated by using binary cross entropy. The raw image pixel is mapping with class scores by using a score function. The quality of particular set of parameters is measured by loss function. It is based on how well the induced scores approved with the ground truth labels in the training data. The loss function calculation is very important to improve the accuracy. If the loss function is high, when the accuracy is low. Similarly, the accuracy is high, when the loss function is low.

Before implementing CNN model,google drive is mounted as implementing is to be done on google colab and then warnings librairy is also imported to ignore warnings as shown below :

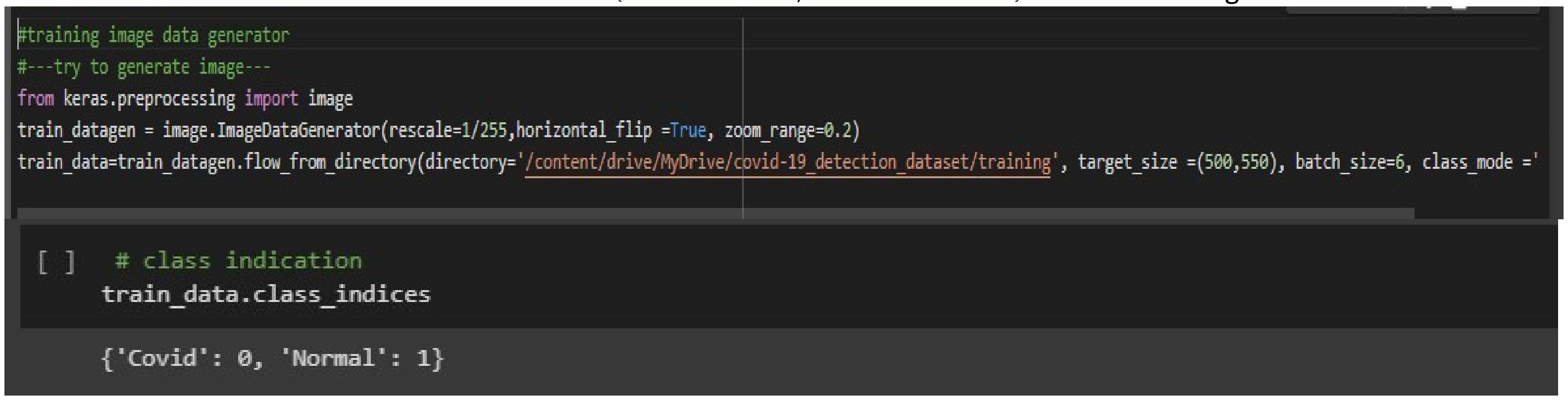


```
[ ]  #Mounting google drive
     from google.colab import drive
     drive.mount('/content/drive')

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[ ]  #To ignore the warnings
     import warnings
     warnings.filterwarnings("ignore")
```

fig 3.5 : codes for mounting google drive and importing warning laibrary

Next after impoting required python liabrairies ,image data are generated from keras.preprocessing import image for training.

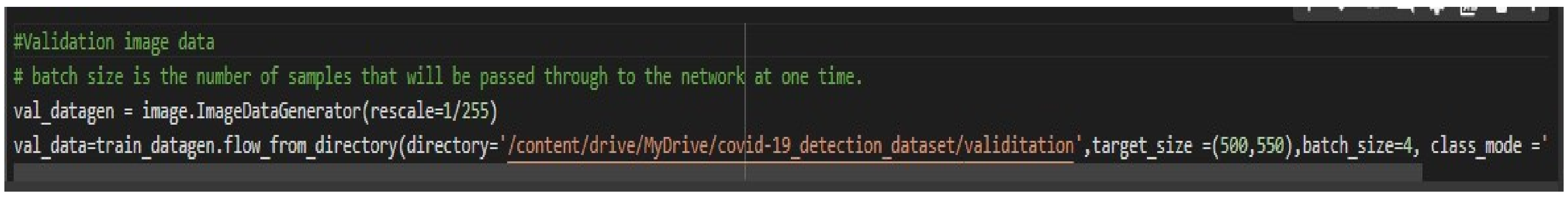
Then trained data are classified as {'Covid': 0, 'Normal': 1} shown in figure.



```
#training image data generator
#---try to generate image---
from keras.preprocessing import image
train_datagen = image.ImageDataGenerator(rescale=1/255,horizontal_flip =True, zoom_range=0.2)
train_data=train_datagen.flow_from_directory(directory='/content/drive/MyDrive/covid-19_detection_dataset/training', target_size =(500,550), batch_size=6, class_mode ='

[ ]  # class indication
    train_data.class_indices

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

Fig 3.6: codes for image data generator

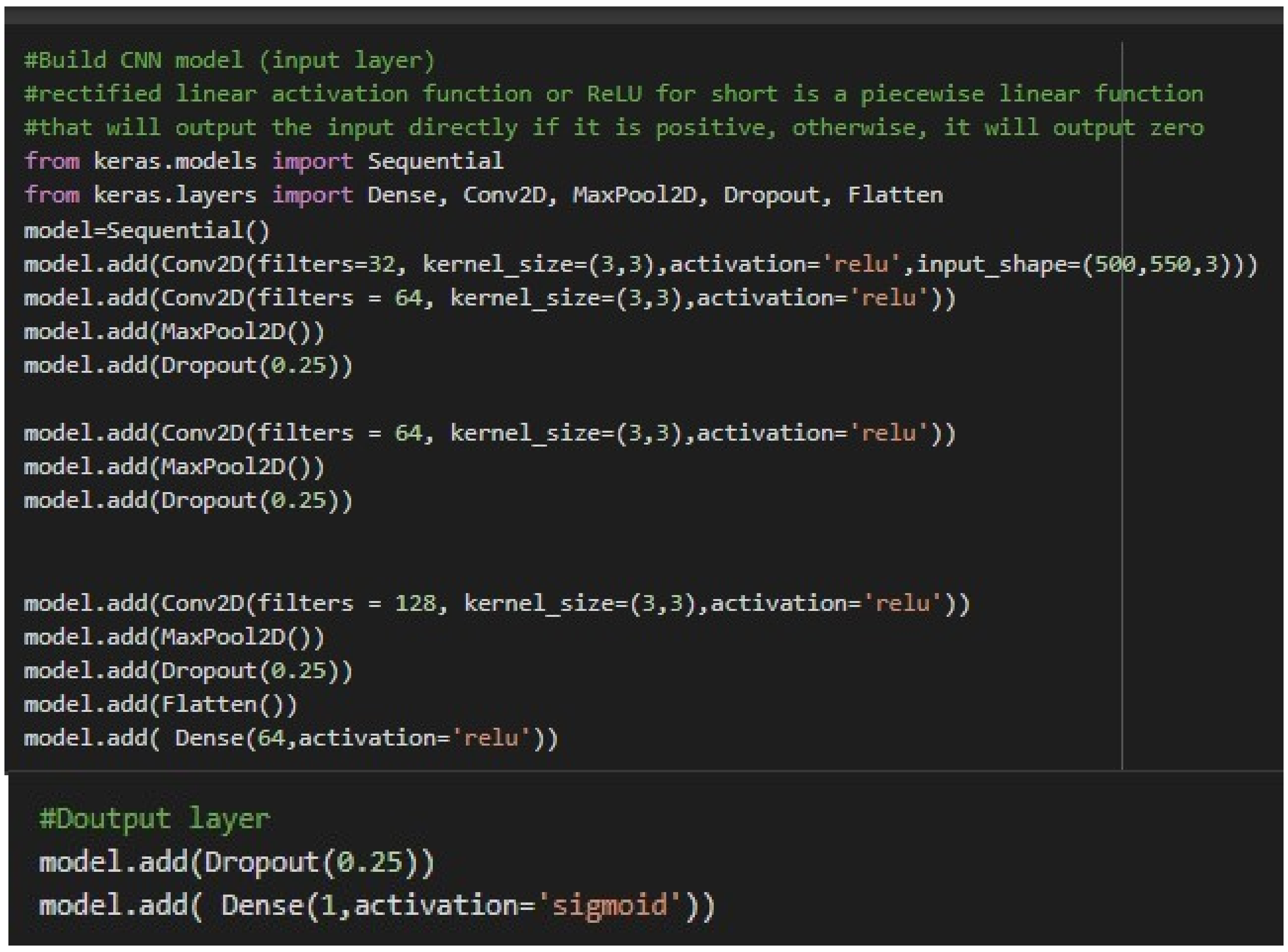
Again, validation is done for image data.



```
#Validation image data
# batch size is the number of samples that will be passed through to the network at one time.
val_datagen = image.ImageDataGenerator(rescale=1/255)
val_data=train_datagen.flow_from_directory(directory='/content/drive/MyDrive/covid-19_detection_dataset/validitation',target_size =(500,550),batch_size=4, class_mode ='
```

fig 3.7: codes for validation

Finally, CNN model is build (input layer and output layer) as shown below.



```
#Build CNN model (input layer)
#rectified linear activation function or ReLU for short is a piecewise linear function
#that will output the input directly if it is positive, otherwise, it will output zero
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Dropout, Flatten
model=Sequential()
model.add(Conv2D(filters=32, kernel_size=(3,3),activation='relu',input_shape=(500,550,3)))
model.add(Conv2D(filters = 64, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.25))


model.add(Conv2D(filters = 128, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.25))
model.add(Flatten())
model.add( Dense(64,activation='relu'))
```

```
#Doutput layer
model.add(Dropout(0.25))
model.add( Dense(1,activation='sigmoid'))
```

Fig 3.8: codes for building CNN model(input layer and output layer)

Conv2D:

Conv2D is a 2D Convolution Layer, this layer creates a convolution. kernel that is wind with layers input which helps produce a tensor of outputs.

Kernel_size:

An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.

Filters:

Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).

Activation:

Activation function to use. Activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.

MaxPooling2D:

Downsamples the input representation by taking the maximum value over the window defined by pool_size for each dimension along the features axis.

Pool size:

Integer or tuple of 2 integers, window size over which to take the maximum.(2, 2) will take the max value over a 2x2 pooling window. If only one integer is specified, the same window length will be used for both dimensions.

Flatten:

Flatten is used to flatten the input.which flattens to one dimensional vector.

15 The Features of the model are as −

First layer, Conv2D consists of 16 filters and 'relu' activation function with kernel size, (3,3).

Second layer, MaxPooling has pool size of (2, 2).

Third layer, Dropout has 2.5 as its value.

Fourth layer, Conv2D consists of 32 filters and 'relu' activation function with kernel size, (3,3).

Fifth layer, MaxPooling has pool size of (2, 2).

Sixth layer, Dropout has 2.5 as its value.

Seventh layer, Conv2D consists of 64 filters and 'relu' activation function with kernel size,(3,3)

Eighth layer, MaxPooling has pool size of (2, 2).

Nineth layer, Dropout has 2.5 as its value.

Tenth layer, Conv2D consists of 64 filters and 'relu' activation function with kernel size, (3,3).

Eleventh layer, MaxPooling has pool size of (2, 2).

Twelveth layer, Dropout has 2.5 as its value.

Thirteenth layer, Flatten is used to flatten all its input into single dimension.

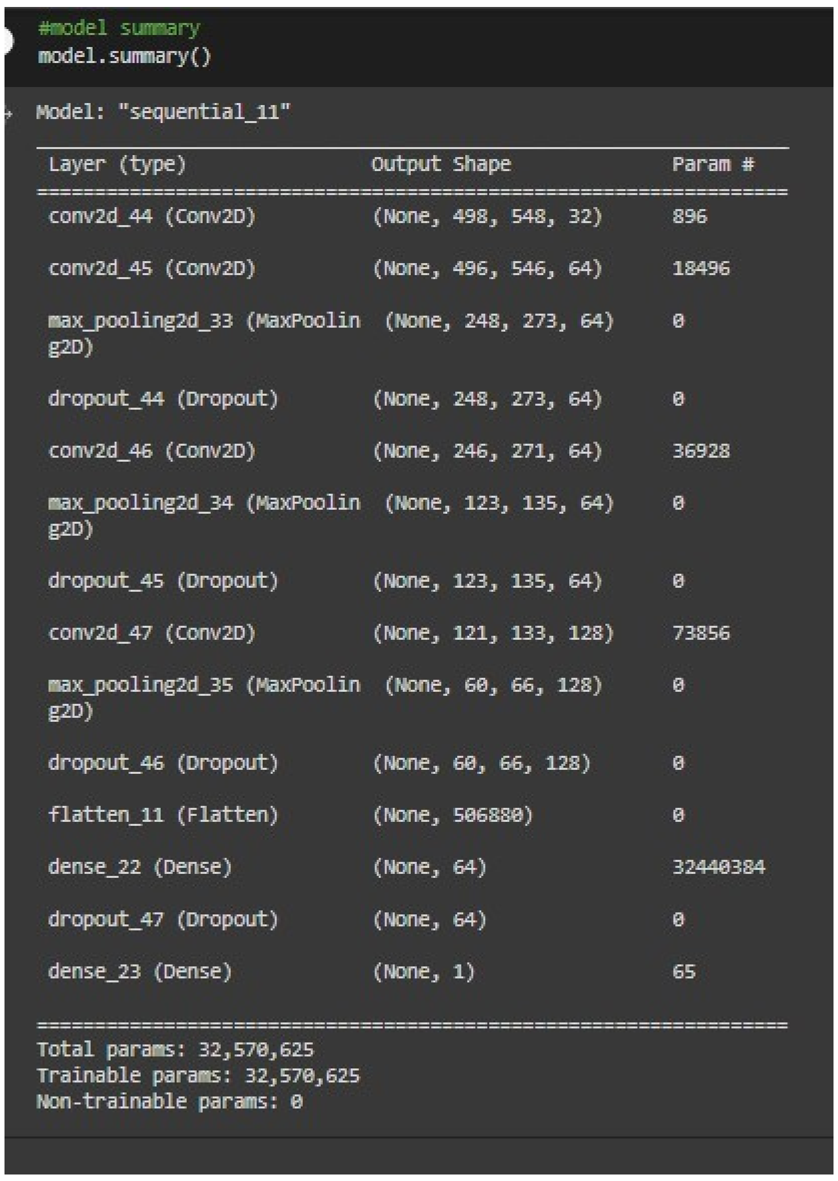
Fourteenth layer, Dense consists of 128 neurons and 'relu' activation function.

Fifteenth layer, Dropout has 0.5 as its value.

Sixteenth and final layer, consists of 4 neurons and 'softmax' activation function. categorical_crossentropy as loss function.

Adam() as Optimizer.
accuracy as metrics.

3.4: Model Summary:



```
#model summary
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_44 (Conv2D)          (None, 498, 548, 32)      896

 conv2d_45 (Conv2D)          (None, 496, 546, 64)      18496

 max_pooling2d_33 (MaxPoolin  (None, 248, 273, 64)     0
 g2D)

 dropout_44 (Dropout)        (None, 248, 273, 64)      0

 conv2d_46 (Conv2D)          (None, 246, 271, 64)      36928

 max_pooling2d_34 (MaxPoolin  (None, 123, 135, 64)     0
 g2D)

 dropout_45 (Dropout)        (None, 123, 135, 64)      0

 conv2d_47 (Conv2D)          (None, 121, 133, 128)     73856

 max_pooling2d_35 (MaxPoolin  (None, 60, 66, 128)      0
 g2D)

 dropout_46 (Dropout)        (None, 60, 66, 128)       0

 flatten_11 (Flatten)        (None, 506880)            0

 dense_22 (Dense)            (None, 64)                32440384

 dropout_47 (Dropout)        (None, 64)                0

 dense_23 (Dense)            (None, 1)                 65

=================================================================
Total params: 32,570,625
Trainable params: 32,570,625
Non-trainable params: 0
```

Fig3.9: model summary

3.5: Training the model :

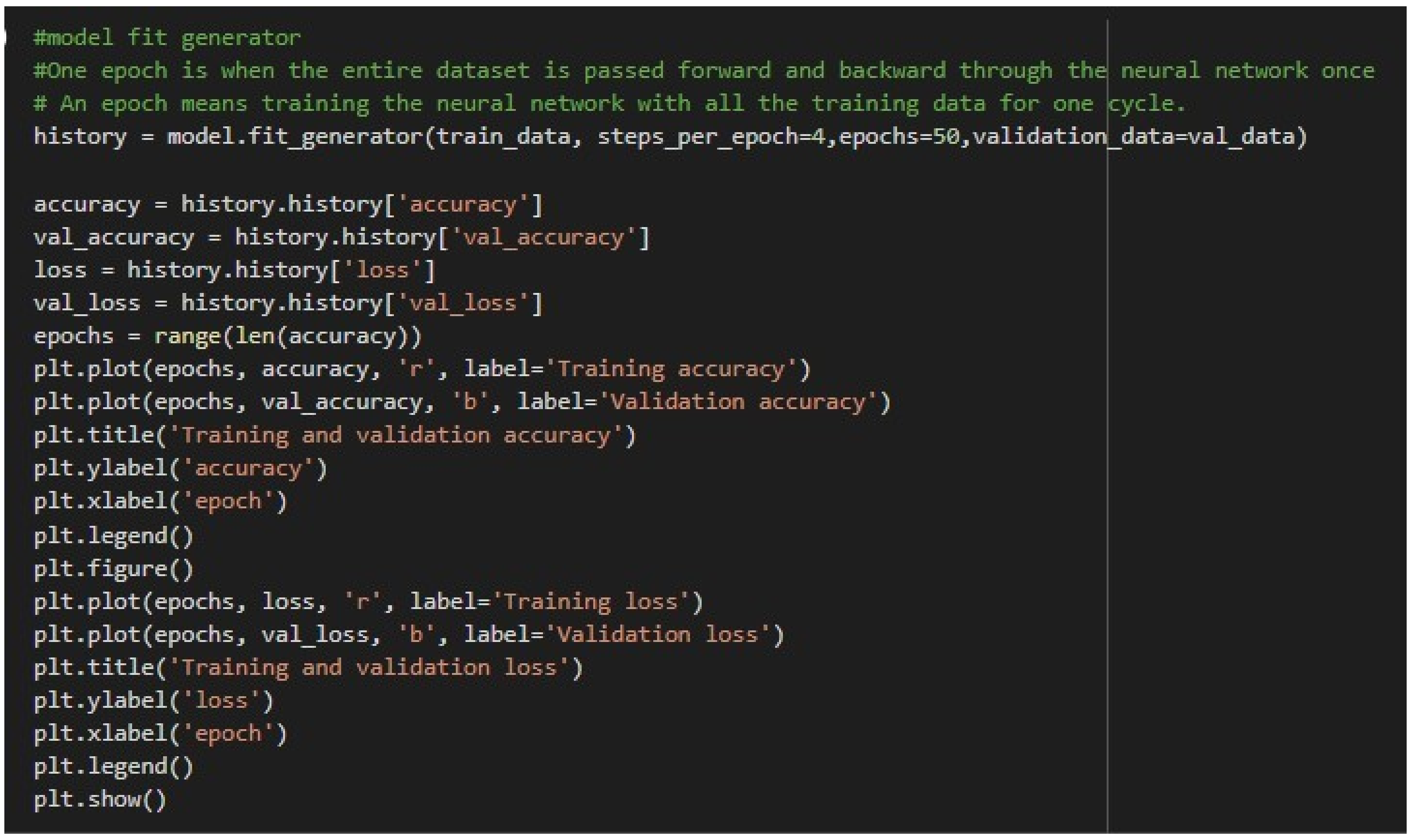
The model is trained for 50 epochs with 4 steps per epoch and also pass the validation data which created earlier in order to validate the model's performance.

3.6: Adam optimizer:

Adam is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks. . The algorithms leverages the power of adaptive learning rates methods to find individual learning rates for each parameter.

3.7:Results and Accuracy Discussion

3.7.1: Codes for plotting graphs:

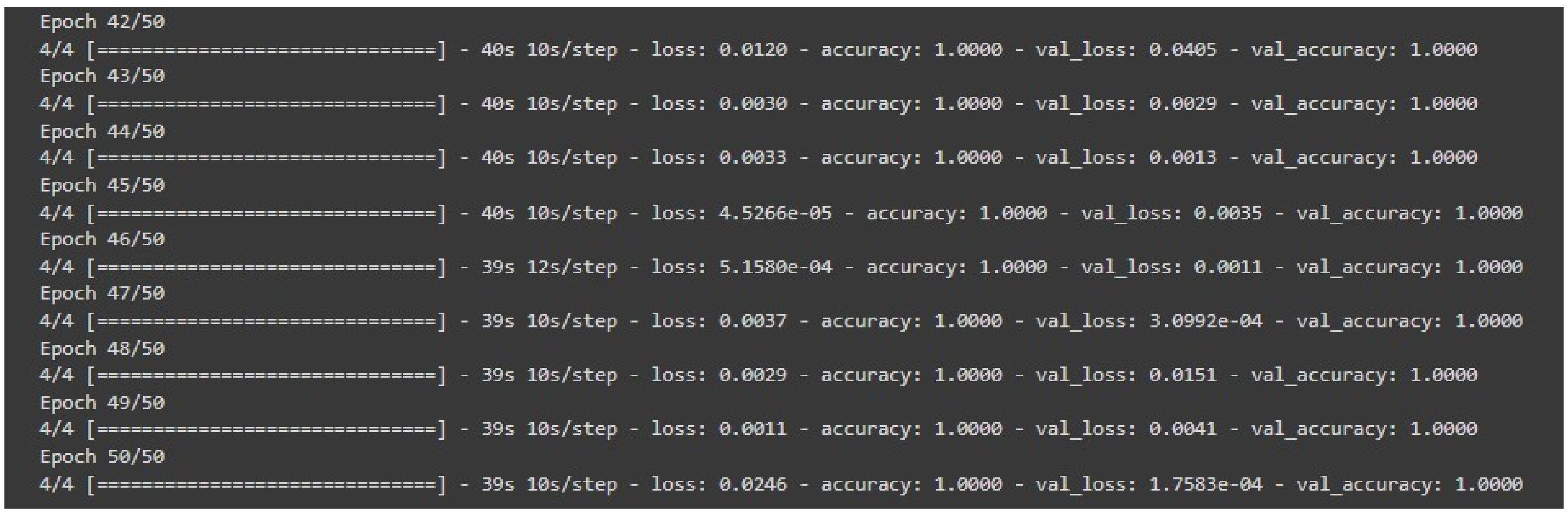


```
#model fit generator
#One epoch is when the entire dataset is passed forward and backward through the neural network once
# An epoch means training the neural network with all the training data for one cycle.
history = model.fit_generator(train_data, steps_per_epoch=4,epochs=50,validation_data=val_data)

accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'r', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()
```

fig 3.10: code for plotting graphs

3.7.2 Epochs figure:

After successfully training the model for **50 epochs with 4 steps per epoch**. **The training accuracy of 100.00% and validation accuracy of 100%** are from the model is made as shown below:



```
Epoch 42/50
4/4 [==============================] - 40s 10s/step - loss: 0.0120 - accuracy: 1.0000 - val_loss: 0.0405 - val_accuracy: 1.0000
Epoch 43/50
4/4 [==============================] - 40s 10s/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.0029 - val_accuracy: 1.0000
Epoch 44/50
4/4 [==============================] - 40s 10s/step - loss: 0.0033 - accuracy: 1.0000 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 45/50
4/4 [==============================] - 40s 10s/step - loss: 4.5266e-05 - accuracy: 1.0000 - val_loss: 0.0035 - val_accuracy: 1.0000
Epoch 46/50
4/4 [==============================] - 39s 12s/step - loss: 5.1580e-04 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 47/50
4/4 [==============================] - 39s 10s/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 3.0992e-04 - val_accuracy: 1.0000
Epoch 48/50
4/4 [==============================] - 39s 10s/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.0151 - val_accuracy: 1.0000
Epoch 49/50
4/4 [==============================] - 39s 10s/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.0041 - val_accuracy: 1.0000
Epoch 50/50
4/4 [==============================] - 39s 10s/step - loss: 0.0246 - accuracy: 1.0000 - val_loss: 1.7583e-04 - val_accuracy: 1.0000
```

fig 3.11: epochs running output

.

3.7.3: Plotted Graphs:

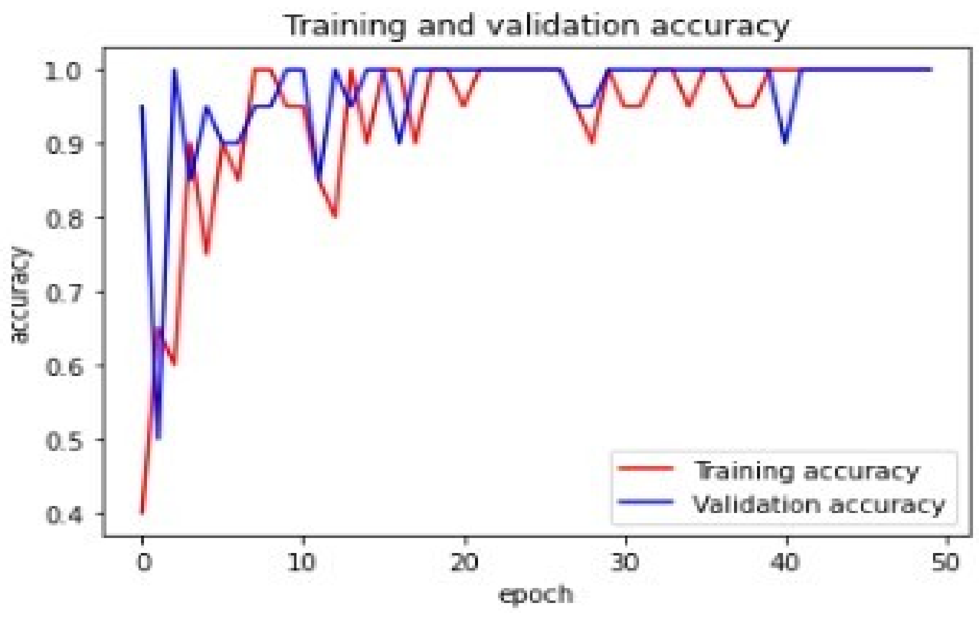


Fig.3.12 : Epochs VS Training&Validation accuracy graph.

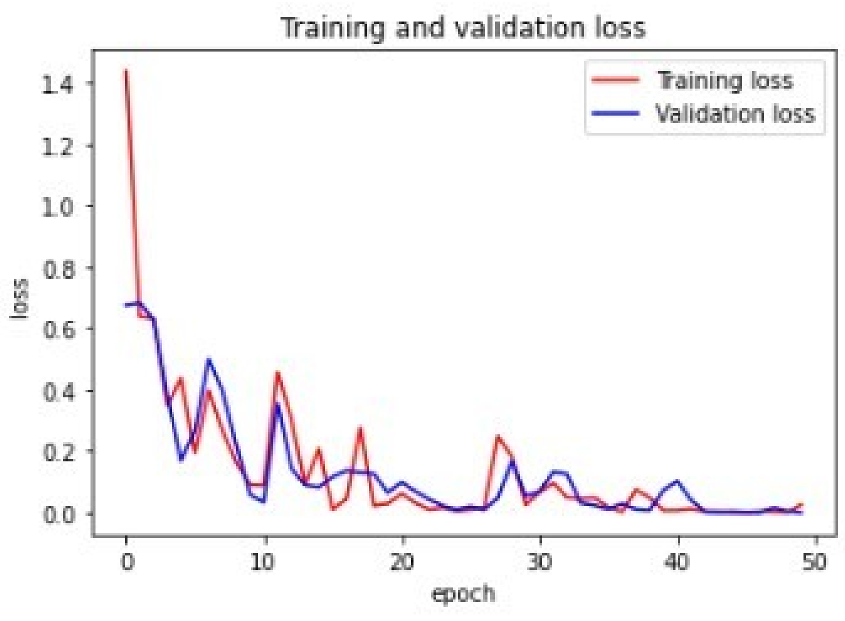


Fig.3.13 : Epochs VS Training&Validation loss graph.

3.8: Testing With Unlabeled Data:

After training the model, I fed the CNN with 20 unlabelled images which has 10 images each of each class.

After feeding in these 20 from class Covid positive 10 out of 10 images were predicted correctly,,and from normal class,10 out of 10 were predicted correctly by the model.Accuracy of 100.00% is obtained to classify Covid positive and 100.00% for normal.

Fig 3.12 displays the Accuracy of the tested labels

Accuracy = (TP+TN)/(TP+TN+FP+FN)

Where

TP = True Positives,

TN = True Negatives,

FP = False Positives, and

FN = False Negatives

CHAPTER 4

Discussion:

Many techniques have been using the RT-PCR test and viral antigen 338 detection techniques to identify COVID-19 cases, but none reported to claim 339 the 100% accuracy in this case.

Besides, most of this existing process are cost heavy, time-consuming and requires specific instructions to implement them. However, sample collection is from a large population is a slow process and hence, the infection may remain undetected.

Instead, chest-X-ray images are more accessible than other diagnostic measures. This proposed CNN model can automatically detect these cases more accurately with high specificity .

CHAPTER 5

Conclusion and Future Work:

5.1 CONCLUSION:

The study proposed a CNN model, which analysed chest X-ray images of COVID-19, healthy and other viral pneumonia patients to classify and diagnosis COVID-19 patients automatically in a short period of time.

Again, various machine and deep learning-based approach are used to compare the performance and the outcomes of our proposed CNN model. Hence, proposed CNN yielded the highest 100.00% accuracy and 100.00% specificity, while most of the deep learning methods did not provided good results like general classifiers.

5.2 FUTURE WORK:

Hence, in future, we will collect a large number of images fromvarious sources and analyze them to get more feasible outcomes. This approach may be helpful for clinical practices and detection of COVID-19 cases to prevent future community transmission.

Reference:

Ai, T., Yang, Z., Hou, H., Zhan, C., Chen, C., Lv, W., Tao, Q., Sun, Z., & Xia, L. (2020). Radiology, 296(2), E32–E40.
Apostolopoulos, I. D., & Mpesiana, T. A. (2020). Covid-19: automatic detection from Xray images utilizing transfer learning with convolutional neural networks. Physical and Engineering Sciences in Medicine, 43(2), 635–640.

Medical Imaging Databank of the Valencia region BIMCV (2020). BIMCV-Covid19 – BIMCV.
Bravo Ortíz, M. A., Arteaga Arteaga, H. B., Tabares Soto, R., Padilla Buriticá, J. I., & Orozco-Arias, S. (2021). Cervical cancer classification using convolutional neural networks, transfer learning and data augmentation. Revista EIA, 18(35), 1–12.

Bustos, A., Pertusa, A., Salinas, J. M., & de la Iglesia-Vayá, M. (2020). PadChest: A large chest x-ray image dataset with multi-label annotated reports. Medical Image Analysis, 66, Article 101797.

Civit-Masot, J., Luna-Perejón, F., Morales, M. D., & Civit, A. (2020). Deep learning system for COVID-19 diagnosis aid using X-ray pulmonary images. Applied Sciences (Switzerland), 10(13).

Cohen, J. P., Morrison, P., & Dao, L. (2020). COVID-19 image data collection. ArXiv, arXiv:2003.11597. COVID-19 X rays (2020). Kaggle.

Covid'19 chest X-ray images dataset for covid positive cases:-
https://github.com/ieee8023/covid-chestxray-dataset

Covid'19 chest X-ray images dataset for negative/normal cases:-
https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Appendix:

```
#Mounting google drive
from google.colab import drive
drive.mount('/content/drive')


#To ignore the warnings
import warnings
warnings.filterwarnings("ignore")


#Import basic python librairies for image preprocessing
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.python.pywrap_tensorflow_internal import *
import keras
from keras.layers import Dense, Conv2D, MaxPool2D, Dropout, Flatten
from keras.models import Sequential
from keras.preprocessing import image
from PIL import Image


#training image data generator
#---try to generate image---
from keras.preprocessing import image
train_datagen = image.ImageDataGenerator(rescale=1/255,horizontal_flip =True, zoom_range=0.2)
train_data=train_datagen.flow_from_directory(directory='/content/drive/MyDrive/covid-
19_detection_dataset/training', target_size =(500,550), batch_size=6, class_mode ='binary')


 # class indication
train_data.class_indices


#Validation image data
# batch size is the number of samples that will be passed through to the network at one time.
val_datagen = image.ImageDataGenerator(rescale=1/255)
val_data=train_datagen.flow_from_directory(directory='/content/drive/MyDrive/covid-
19_detection_dataset/validitation',target_size =(500,550),batch_size=4, class_mode ='binary')

#Build CNN model (input layer)
#rectified linear activation function or ReLU for short is a piecewise linear function
#that will output the input directly if it is positive, otherwise, it will output zero
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Dropout, Flatten
model=Sequential()
model.add(Conv2D(filters=32, kernel_size=(3,3),activation='relu',input_shape=(500,550,3)))
model.add(Conv2D(filters = 64, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.25))

model.add(Conv2D(filters = 64, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.25))
model.add(Conv2D(filters = 128, kernel_size=(3,3),activation='relu'))
model.add(MaxPool2D())
model.add(Dropout(0.25))
model.add(Flatten())
model.add( Dense(64,activation='relu'))
```

```
#Doutput layer
model.add(Dropout(0.25))
model.add( Dense(1,activation='sigmoid'))


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


#model summary
model.summary()
```

Plotting Graphs

```
#model fit generator
#One epoch is when the entire dataset is passed forward and backward through the neural network once
# An epoch means training the neural network with all the training data for one cycle.
history = model.fit_generator(train_data, steps_per_epoch=4,epochs=50,validation_data=val_data)

accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'r', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()


#!pip install Pillow==5.3.0
from PIL import Image


#checking how model is performing
path="/content/drive/MyDrive/covid-19_detection_dataset/validitation/Covid/covid_8.jpeg"
img=Image.open(path)
img=image.img_to_array(img)/255
plt.imshow(img)
img.resize(500,550,3)
img=np.array([img])
img.shape


#model prediction
model.predict(img)
```