

MINI PROJECT REPORT

on

COVID-19 DETECTION USING DEEP LEARNING

Submitted in partial fulfilment for the completion of

BE-IV Semester

In

INFORMATION TECHNOLOGY

By

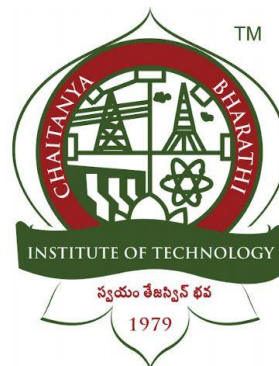
Y. KRISHNA GUPTHA (160120737155)

P. MANOJ KUMAR (160120737157)

Under the guidance of

Smt. B. Swathi Sowmya

Assistant Professor



DEPARTMENT OF INFORMATION TECHNOLOGY
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)
(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO Certified 9001:2015)
GANDIPET, HYDERABAD – 500 075

Website: www.cbit.ac.in

2021-2022

CERTIFICATE

This is to certify that the project work entitled “**COVID-19 DETECTION USING DEEP LEARNING**” was submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**, in partial fulfillment of the requirements for the award of the completion of the IV semester of B.E in Information Technology, during the academic year 2021-2022, is a record of original work done by **Y. KRISHNA GUPTHA (160120737155), P. MANOJ KUMAR (160120737157)** during the period of study in Department of IT, CBIT, HYDERABAD, under our supervision and guidance.

Project Guide

Smt. B. Swathi Sowmya

Asst. Professor, Dept. of IT,
CBIT, Hyderabad.

Head of the Department

Dr. K. Radhika

Professor, Dept. of IT,
CBIT, Hyderabad.

ACKNOWLEDGMENTS

We would like to express our deep and sincere gratitude to our project guide, Smt. B. Swathi Sowmya Asst. Professor for allowing us to do the project and providing his invaluable guidance throughout this project. It was a great privilege and honor to work and study under his guidance.

We would like to thank our Head of Department, Dr. K. Radhika, for allowing us to improve our skills, for the constant support, and provided us with the sources that are required for the completion of this project.

Our sincere thanks to the Principal, Dr. P. Ravinder Reddy, as well as the management of the institute, for giving us a good opportunity to improve our skills in a good learning atmosphere.

We are extremely grateful to our parents for their love, prayers, care, and sacrifices for educating and preparing for our future and supporting us in the completion of this project.

We would like to thank our friends for their constant help and guidance and their support in the completion of this project.

ABSTRACT

COVID-19 typically known as Coronavirus disease is an infectious disease caused by a newly discovered coronavirus, which has emerged in the Republic of China for an undetermined cause and has affected the whole world quickly. It is important to detect positive cases early to prevent the further spread of the outbreak. In order to test a COVID-19 patient, a healthcare provider uses a long swab to take a nasal sample. The diagnosis becomes even more critical when, there is a lack of reagents or testing capacity, to track the virus and its severity, and the risk of a healthcare practitioner getting affected when he comes in contact with COVID-19-positive patients. In this scenario of the COVID-19 pandemic, there is a need for streaming diagnosis based on a retrospective study of laboratory data in form of chest X-rays using deep learning.

The dataset used in this project consists of X-ray images which are provided by Kaggle.com and other Github resources in order to train and test the deep learning model. This dataset consists of 3 categories viz. Covid, Normal, Viral Pneumonia in each of the training and testing data. The total number of images taken is 15,000 which is further divided into two parts training set and a testing set with the ratio of 4:1 i.e.; The training set has 12,000 samples and the testing set has 3000 samples. In the support of this project, we have used libraries viz. TensorFlow, Keras and its applications of predefined model architectures like ResNet50 or VGG16, matplotlib, NumPy, EarlyStopping, ModelCheckpoint, streamlit for the Image visualization, analytics, and for saving the optimized model respectively. The source code is written under a Python environment. After training the model, in order to prove real-time efficiency, we evaluated our model to check the performance by plotting the trained model's accuracy and loss with respect to the validated model. The accuracy of the validated model was found to be around 94 %. Further, the deployment is achieved through a local host.

Early diagnosis is essential both for early intervention of the patient and to prevent the risk of transmission. For this purpose, chest X-ray images were used, obtained from Covid-19 and non-Covid-19 patients. After the positive diagnosis of COVID-19 patients, we aim to track the progression of the disease which may help healthcare professionals work on the correct dynamics and treatment of patients. It can also be used in situations where the possibilities are insufficient (RT-PCR test, doctor, radiologist). In future work, more successful deep learning models can be created.

TABLE OF CONTENTS

S.NO	TITLE	PAGE.
I.	TITLE PAGE	
II.	CERTIFICATE	i
III.	ACKNOWLEDGEMENT	ii
IV.	ABSTRACT	iii
V.	LIST OF FIGURES	vi
VI.	LIST OF TABLES	vii
VII.	LIST OF ABBREVIATIONS	vii
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 APPLICATIONS	1
	1.3 PROBLEM STATEMENT	1
	1.4 ORGANIZATION OF THE PROJECT	2
2	LITERATURE SURVEY	3
3	SYSTEM REQUIREMENT SPECIFICATION	4
	3.1 FUNCTIONAL REQUIREMENTS	4
	3.2 NON-FUNCTIONAL REQUIREMENTS	4
	3.3 SOFTWARE REQUIREMENTS	5
	3.4 HARDWARE REQUIREMENTS	5

4	METHODOLOGY	6
	4.1 SYSTEM ARCHITECTURE	6
	4.2 ALGORITHMS	9
5	IMPLEMENTATION	12
6	RESULTS	16
7	CONCLUSION AND FUTURE SCOPE	20
	BIBLIOGRAPHY	21

LIST OF FIGURES

FIG.NO	FIGURE NAME	PAGE NO.
4.1	Composition of dataset	06
4.2	Evaluation metrics	07
4.3	accuracy vs val_accuracy	08
4.4	Loss vs val_loss	08
4.5	VGG16 Architecture	09
4.6	VGG16 Block Diagram	09
4.7	RESNET50 Architecture	10
4.8	RESNET50 Block Diagram	10
4.9	Overall workflow Diagram	11
5.1	Front-End: Importing the required modules	12
5.2	Front-End: Function for loading the vgg16 & resnet50 models.	12
5.3	Front-End: Function for importing and predicting the images	13
5.4	Front-End: If a button is clicked, probabilities are displayed.	13
5.5	Back-End: Importing the required modules.	14
5.6	Back-End: Loading and pre-processing of images.	14
5.7	Back-End: Importing vgg16 model.	14
5.8	Back-End: Importing resnet50 model.	15
5.9	Back-End: Compiling the vgg16 model	15
5.10	Back-End: Compiling resnet50 model	15
5.11	Back-End: Optimal acc & val_accuracy for vgg16.	15
5.12	Back-End: Optimal acc & val_accuracy for resnet50	15
6.1	User Interface	16
6.2	Covid test case	17
6.3	Normal test case	18
6.4	Pneumonia test case	19

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
2.1	Literature Survey	03

LIST OF ABBREVIATIONS

VGG16	: Visual Geometry Group-16
RESNET50	: Residual Networks-50
val_acc	: Validation accuracy
val_loss	: Validation loss
COVID-19	: Corona Virus Disease-19
RT-PCR	: Real-time reverse transcription-polymerase Chain Reaction
MRI	: Magnetic resonance imaging

1. INTRODUCTION

1.1 OVERVIEW

Nowadays due to the pandemic situation, it is a very risky and time taking process for medical practitioners to diagnose the presence of covid in each patient through long swabs and testing with RT-PCR or RDT physically. “Covid-19 Detection using Deep learning” is a website aimed to classify not only covid-19 but also pneumonia based on X-Ray images of a patient. Patterns in the image are detected and a predicted outcome along with precautions are displayed. It helps clinical practitioners to diagnose a patient in a run time.

1.2 APPLICATIONS

1. This website provides the practitioner an easy interface to classify X-Ray Images and diagnose them.
2. Instead of testing them physically this platform makes things easier in this pandemic situation.
3. A practitioner can upload the X-Ray image in a safe and online mode.
4. He will receive an instant reply i.e., a run-time prediction and other precautions for the disease for efficient diagnosis.

1.3 PROBLEM STATEMENT

Medical Practitioners are facing a significant problem in diagnosing the patient. With the advancement in technology, there are so many enhancements in the medical field also. As the data is increasing daily due to digitalization in the medical field, a medical practitioner needs to diagnose the patients with the least probability of themselves getting affected through manual testing of covid-19 and other several diseases. Daily there are so many applications that are challenging to manage the covid tests, and also the chances of some mistakes are high. One mistake can make a massive effect on a practitioner’s personal life. Deep Learning a subset of Artificial intelligence (AI), as a typical method for information investigation, has gotten more consideration in this field increasingly.

1.4 ORGANISATION OF THE REPORT

The mini-project is divided into 6 parts.

1. First part contains an introduction to the project.
2. Second part of the project contains the literature survey which consists of software and hardware requirements and the existing system of the project.
3. Third part contains the methodology of the project and block diagram of the proposed system.
4. Fourth part contains the implementation of the project this consists of the code used for execution.
5. Fifth part contains the testing and results of the project. It consists of screenshots of the execution of the project.
6. Sixth part contains the conclusion of the project and the future of the project.

2. LITERATURE SURVEY

S. No.	Title	Source	Year	Advantages	Gaps
1.	OptCoNet: an optimized convolutional neural network for an automatic diagnosis of COVID-19	National Library of Medicine	2020	A swarm intelligent (SI) optimization algorithm implemented during the training of the CNN helped the network achieve good accuracy.	This method has a major difficulty in choosing the right parameters for the optimizer involved.
2.	Deep learning-based detection and analysis of COVID-19 on chest X-ray images	National Library of Medicine	2020	The Xception net algorithm has the best performance and is suited to be used for automated diagnosis tasks.	The high accuracy obtained may be a cause of concern since it may be a result of overfitting.
3.	Covid-19 Classification Using Deep Learning in Chest X-Ray Images	IEEE Xplore	2020	Results are encouraging in terms of the use of computer-aided in the field of pathology.	The data set used for training is small.
4.	Robust Technique to Detect COVID-19 using Chest X-ray Images	IEEE Xplore	2020	Tracks the progression of the disease which may help healthcare professionals work on the correct dynamics and treatment of patients	Applying this technique to real-time clinical data and improving the accuracy may take a lot of time

Table (2.1) Literature Survey

3. SYSTEM REQUIREMENT SPECIFICATION

3.1 FUNCTIONAL REQUIREMENTS

- An input image of an X-Ray must not be hazy or ambiguous.
- The website must not stop working when kept running for even a long time.
- The website must function as expected for every set of test cases (X-Ray images) provided.
- The website should generate the predicted output and showcase the percentages of each output for the given input test case.

3.2 NON-FUNCTIONAL REQUIREMENTS

- Response time –The time the system takes to load and the time for responses on any action the user does is around 30 seconds to 1 minute.
- Processing time – 10 to 15 seconds of a time long is acceptable to perform key functions or export/import data.
- Throughput - The number of transactions the system can handle is not restricted.
- Storage - The amount of data to be stored for the system to function is around 170MB.
- Growth Requirements - As the system grows the storage space required to keep up with the efficiency is around 500MB
- Locations of operation - Geographic location, connection requirements and the restrictions of a local network prevail.

3.3 SOFTWARE REQUIREMENTS

Languages used:

Front-end development:

StreamLit - For creating and designing web apps.

Backend development:

1. Google Drive - To store the data.
2. Google Collaboratory – For implementing the Deep Learning model.
3. Visual Studio code for debugging and executing the code
4. Keras, TensorFlow, and other python libraries– To import Deep learning algorithms and applications to evaluate the model.

3.4 HARDWARE REQUIREMENTS

OS : Windows 7 SP1

RAM : 2GB

STORAGE : 5GB

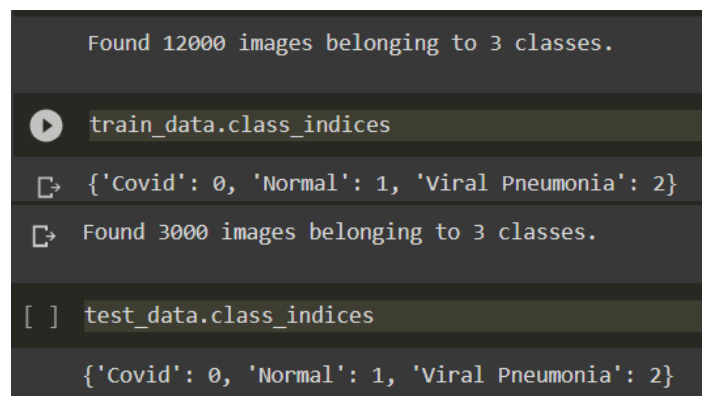
INPUT DEVICE : Keyboard or touch screen.

OTHER REQUIREMENTS: Graphical User Interface (GUI) or integrated video card.

4. METHODOLOGY

4.1 SYSTEM ARCHITECTURE

The proposed system focuses on classifying the type of disease that the patient is suffering from by analyzing patients' X-Ray Images as input. On the output from the Classification model, the decision on whether a person is normal or has covid or pneumonia or the permutation of other diseases can be diagnosed. Using different deep data analytics tools, covid detection and their severity can be forecasted. In this process, we trained two different **CNN architectures viz. VGG16 and RESNET50** by the same data and then compare validated data with trained data to predict the nature of the model. After evaluating two algorithms we found that the VGG16 algorithm started resulting in better accuracy when compared to RESNET50. Every new X-Ray Image of a patient uploaded on the website acts as a test data set. After the operation of testing, the model predicts whether the new image is a fit case for anyone among 3 categories.



```
Found 12000 images belonging to 3 classes.  
  
▶ train_data.class_indices  
↳ {'Covid': 0, 'Normal': 1, 'Viral Pneumonia': 2}  
↳ Found 3000 images belonging to 3 classes.  
  
[ ] test_data.class_indices  
  
{'Covid': 0, 'Normal': 1, 'Viral Pneumonia': 2}
```

Fig (4.1) COMPOSITION OF DATASET:12000 for training & 3000 for testing

The major steps that we employed in developing the Deep Learning Model are

- ▶ **STEP-1: IMPORTING REQUIRED LIBRARIES AND DATASET**
- ▶ **STEP-2: PRE-PROCESSING OF IMAGES (DATA AUGMENTATION)**
- ▶ **STEP-3: LABEL ENCODING**
- ▶ **STEP-4: IMPORT OR BUILD A CNN MODEL**
- ▶ **STEP-5: COMPILING THE MODEL**
- ▶ **STEP-6: TRAINING THE MODEL**
- ▶ **STEP-7: TESTING THE MODEL**
- ▶ **STEP-8: SAVING THE MODEL**
- ▶ **STEP-9: PLOTTING ACCURACY AND LOSS OF THE MODEL**
- ▶ **STEP-10: EVALUATION OF THE MODEL**

```
[ ] model1.history.history.keys()

dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Fig (4.2) EVALUATION METRICS

1. accuracy:

- It measures how often the classifier is correct for both true positives and true negative cases.
- Accuracy score is the number of correct predictions obtained divided by the total number of predictions.

2. loss:

- Loss is a value that represents the summation of errors in our model.
- It measures how well (or bad) our model is doing. If the errors are high, the loss will be high, which means that the model does not do a good job.

3. validation accuracy:

- The accuracy that is calculated on the data set you do not use for training, but you use for validating (or "testing") the generalization ability of your model or for "early stopping"
- It's best to rely on validation accuracy for a fair representation of model performance because a good neural network will end up fitting the training data at 100%, but would perform poorly on unseen data.

4. validation loss:

- Validation loss is a metric used to assess the performance of a deep learning model on the validation set.
- Simply, validation loss indicates how well the model fits new data.

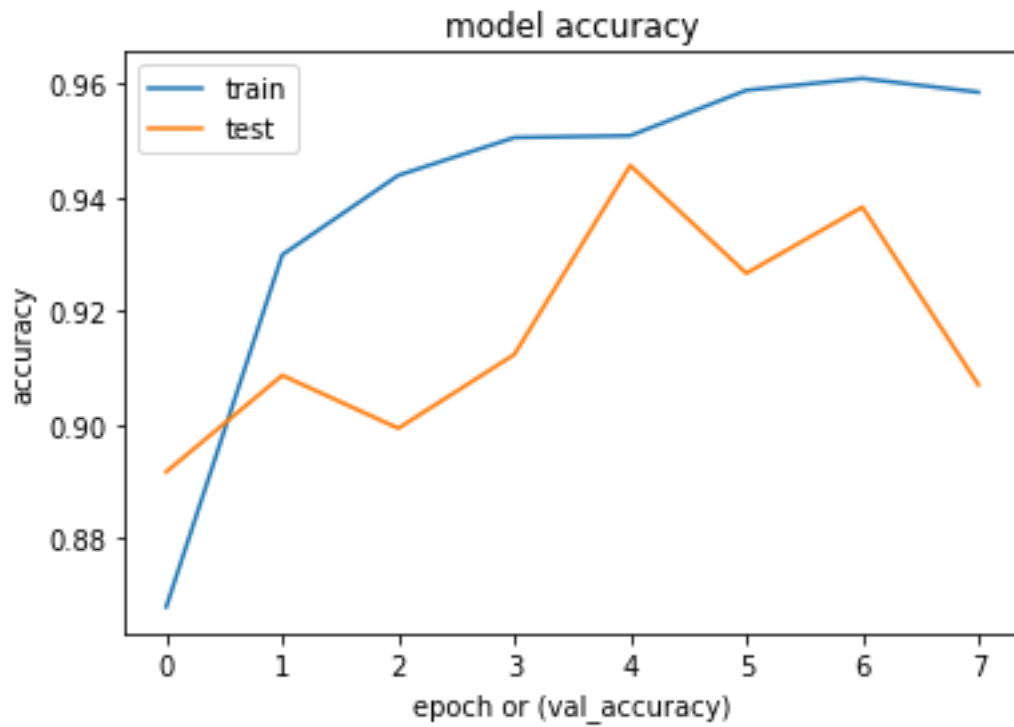


Fig (4.3) accuracy vs val_accuracy between trained and validated model.

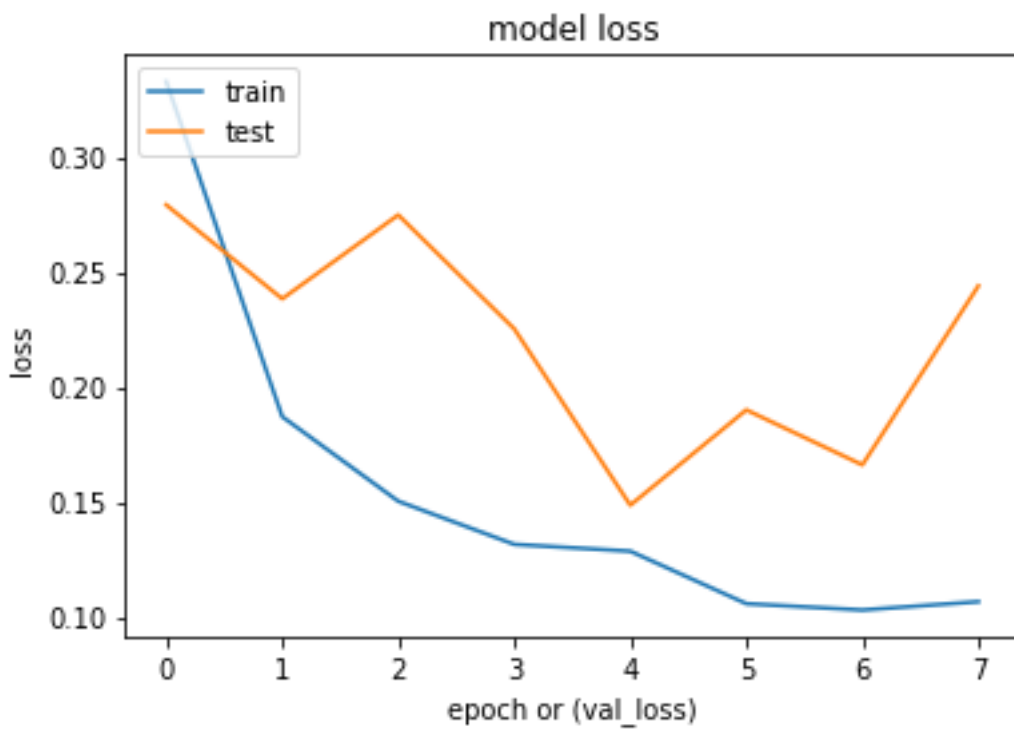


Fig (4.4) loss vs val_loss between trained and validated model.

4.2 ALGORITHMS

4.2.1 VGG16

The Visual Geometry Group is abbreviated as VGG. VGG16 is built using multiple 33 kernel-sized filters sequentially (11 and 5 in the first and second convolutional layers, respectively). VGG's input is set to a 224×224 RGB picture. The network has a depth of 16 layers and is capable of classifying images of multiple classes. The VGG architecture's primary concept is to keep the convolution size modest and constant while designing an extremely deep network.

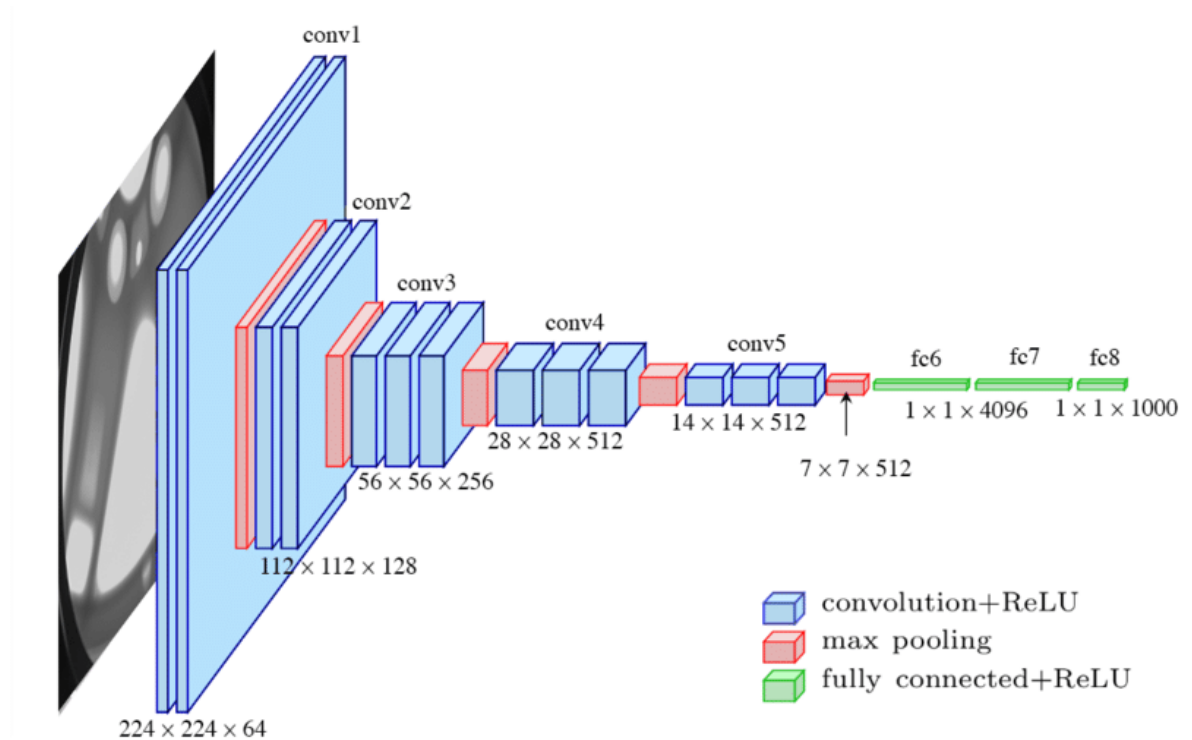


Fig (4.5) VGG16 Architecture: a convolutional neural network that is 16 layers deep.

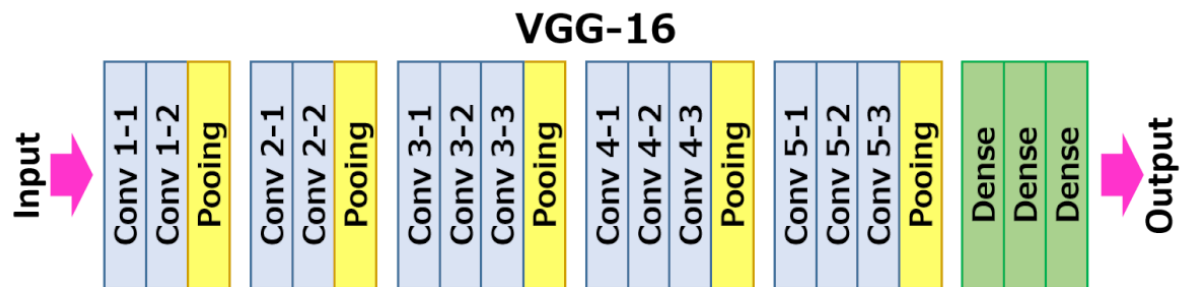


Fig (4.6) VGG16 Block Diagram: layers in the vgg16 architecture.

4.2.2 RESNET50

ResNet50's architecture is divided into 4 stages. The network may accept an input image with a height, width of multiples of 32, and channel width. The network may accept an input image with a height, width of multiples of 32, and channel width. Each ResNet architecture conducts initial convolution and max-pooling with a kernel size of 7×7 and 3×3 , respectively. Each 2-layer block is replaced with this 3-layer bottleneck block in the 34-layer net, resulting in a 50-layer ResNet.

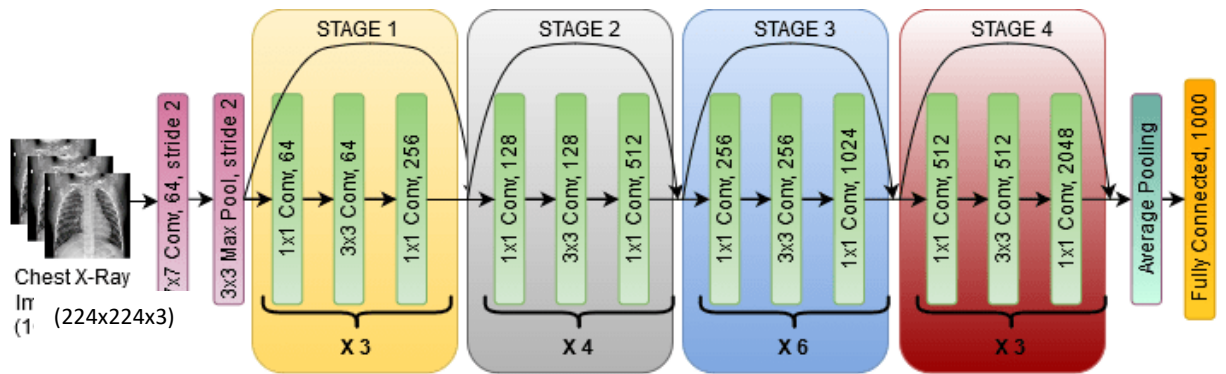


Fig (4.7) RESNET50 Architecture: a convolutional neural network that is 50 layers deep.

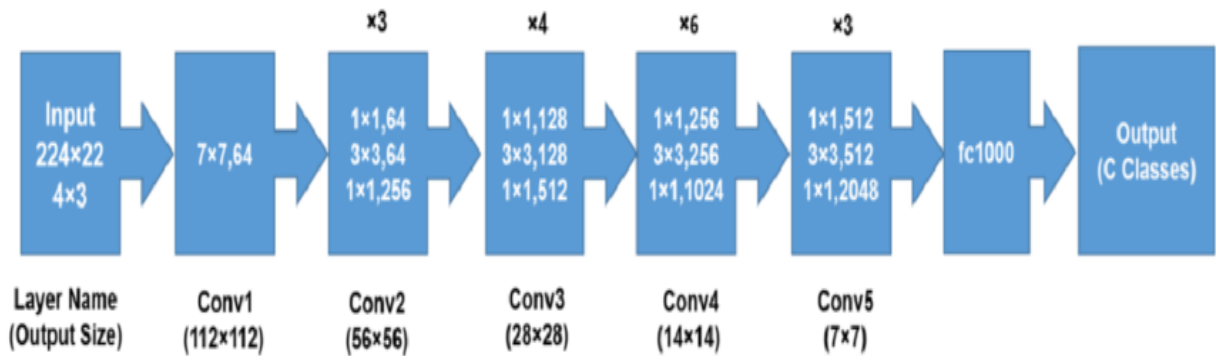


Fig (4.8) RESNET50 Block Diagram: layers in the resnet50 architecture.

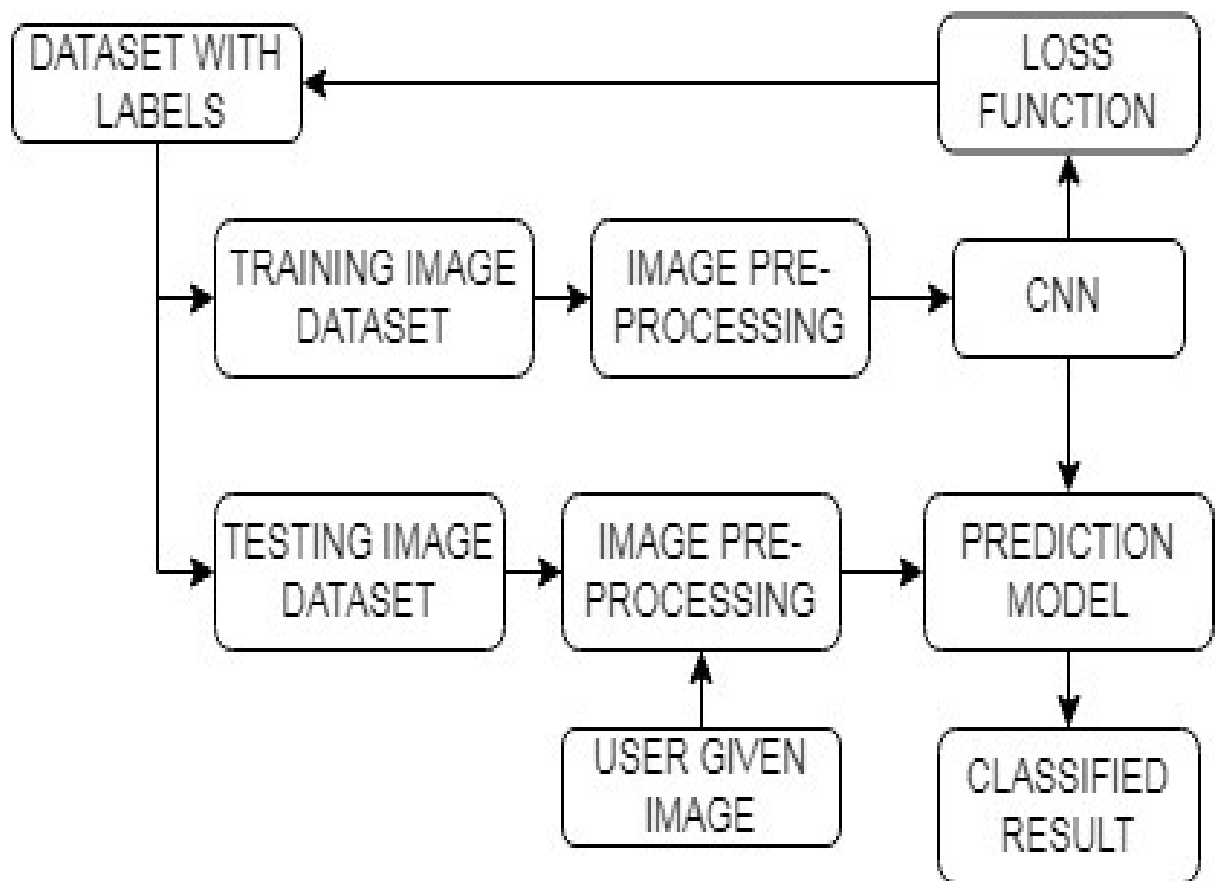


Fig (4.9) Overall system workflow Diagram from input to classified output.

5. IMPLEMENTATION

We have implemented the website in two sections; the front-end section and the back-end section. The front-end section contains the streamlit page in which the user can upload the X-Ray Image and when the “PREDICT” button is clicked result is displayed along with the probabilities of categories. In the back-end section, the Image which is uploaded by the user is given as input to the model which predicts the probability of detected disease for that particular Image at an instant.

We run the implementation of the front-end part and the back-end part in the VS Code which is a source code editor for debugging and tasks running. We then host our model by streamlit through localhost with a temporary domain.

```
#DL_Mini_Project.py
import streamlit as st
import tensorflow as tf
import streamlit as st
import cv2
from PIL import Image, ImageOps
import numpy as np
import base64
```

Fig (5.1) FRONT-END: Importing the required modules.

```
def load_model():
    model1=tf.keras.models.load_model('vgg16/saved_model.pb')
    model2=tf.keras.models.load_model('resnet50/saved_model.pb')
    return model1,model2
with st.spinner('Models are being loaded..'):
    model1,model2=load_model()
```

Fig (5.2) FRONT-END: Function for loading the vgg16 & resnet50 models.

```

def import_and_predict(image_data, model):
    size = (256,256)
    image = ImageOps.fit(image_data, size, Image.ANTIALIAS)
    image = np.asarray(image)
    img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    img_resize = (cv2.resize(img, dsize=(256, 256),interpolation=cv2.INTER_CUBIC))/255.
    img_reshape = img_resize[np.newaxis,...]
    prediction = model.predict(img_reshape)
    return prediction

```

Fig (5.3) FRONT-END: Function for importing and predicting the images.

```

if st.button('PREDICT'):
    st.write('Result...')
    vgg16_predictions = import_and_predict(img, model1)
    resnet50_predictions = import_and_predict(img, model2)

    score1 = tf.nn.softmax(vgg16_predictions[0])
    score2 = tf.nn.softmax(resnet50_predictions[0])
    score1 = score1.numpy()
    score2 = score2.numpy()
    vgg16_scores,resnet50_scores = [],[]
    st.title(f'vgg16 predicted :{CATEGORIES[np.argmax(score1)]}')
    st.title(f'resnet50 predicted :{CATEGORIES[np.argmax(score2)]}')
    print(f'vgg16 output:{CATEGORIES[np.argmax(score1)]}')
    print(f'resnet50 output:{CATEGORIES[np.argmax(score2)]}')
    st.title('vgg16 scores :')
    for index,item in enumerate(CATEGORIES):
        #prints the probability of all the categories
        vgg16_scores.append(score1[index]*100)
        st.write(f'{item} : {vgg16_scores[index]}%')
        print(f'{item} : {vgg16_scores[index]}%')
    st.bar_chart(data = vgg16_scores, width=0, height=0, use_container_width=True)

    st.title('resnet50 scores :')
    for index,item in enumerate(CATEGORIES):
        #prints the probability of all the categories
        resnet50_scores.append(score2[index])
        st.write(f'{item} : {resnet50_scores[index]*100}%')
        print(f'{item} : {resnet50_scores[index]*100}%')
    st.bar_chart(data = resnet50_scores, width=0, height=0, use_container_width=True)

```

Fig (5.4) FRONT-END: If button is clicked, probabilities are displayed

```

import numpy as np
import matplotlib.pyplot as plt

import keras
from keras.applications.vgg16 import VGG16
from keras.applications.resnet import ResNet50
from keras.layers import Dense, Conv2D, MaxPool2D, Dropout, Flatten
from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

```

Fig (5.5) BACK-END: Importing the required modules.

Loading and Pre-processing the training data

```

train_datagen = image.ImageDataGenerator(
    rescale = 1/225, horizontal_flip = True, zoom_range = 0.2, shear_range = 0.2
)
train_data = train_datagen.flow_from_directory(directory = "/content/drive/MyDrive/MP-2/Dataset/Train", target_size = (256,256), batch_size = 64,
                                              class_mode = "categorical")

```

Found 12000 images belonging to 3 classes.

```

[ ] train_data.class_indices

{'Covid': 0, 'Normal': 1, 'Viral Pneumonia': 2}

```

Loading and Pre-processing the testing data

```

[ ] test_datagen = image.ImageDataGenerator(rescale = 1./225)
test_data = test_datagen.flow_from_directory(directory = "/content/drive/MyDrive/MP-2/Dataset/Test", target_size = (256,256), batch_size = 64,
                                             class_mode = "categorical")

```

Found 3000 images belonging to 3 classes.

```

[ ] test_data.class_indices

{'Covid': 0, 'Normal': 1, 'Viral Pneumonia': 2}

```

Fig (5.6) BACK-END: Loading and pre-processing of images.

```

[ ] vgg = VGG16( input_shape=(256,256,3), include_top= False) # include_top will consider the new weights

```

```

[ ] for layer in vgg.layers:
    layer.trainable = False
    # Dont Train the parameters again

```

```

x = Flatten()(vgg.output)
x = Dense(units=3, activation='sigmoid', name = 'predictions')(x)
model1 = Model(vgg.input, x)

```

Fig (5.7) BACK-END: Importing vgg16 model.

```

▶ res = ResNet50( input_shape=(256,256,3), include_top= False) # include_top will consider the new weights

[ ] for layer in res.layers:                # Dont Train the parameters again
    layer.trainable = False

[ ] x = Flatten()(res.output)
    x = Dense(units=3 , activation='sigmoid', name = 'predictions' )(x)

    # creating our model.
    model2 = Model(res.input, x)

```

Fig (5.8) BACK-END: Importing resnet50 model.

```

[ ] model1.compile(optimizer='adam', loss = 'categorical_crossentropy', metrics=['accuracy'])

[ ] es = EarlyStopping(monitor= "val_accuracy" , min_delta= 0.001, patience= 3, verbose=1)
    mc = ModelCheckpoint(filepath="/content/drive/MyDrive/MP-2/vgg16", monitor="val_accuracy", verbose=1, save_best_only= True)

[ ] model1.fit(train_data, epochs = 10, validation_data= test_data, callbacks=[es,mc])

```

Fig (5.9) BACK-END: Compiling the vgg16 model

```

[ ] model2.compile( optimizer= 'adam' , loss = 'categorical_crossentropy', metrics=['accuracy'])

[ ] es = EarlyStopping(monitor= "val_accuracy" , min_delta= 0.001, patience= 3, verbose=1)
    mc = ModelCheckpoint(filepath="/content/drive/MyDrive/MP-2/resnet50", monitor="val_accuracy", verbose=1, save_best_only= True)

▶ model2.fit_generator(train_data, epochs= 10, validation_data= test_data, callbacks=[es,mc])

```

Fig (5.10) BACK-END: Compiling resnet50 model

```

Epoch 5/10
188/188 [=====] - ETA: 0s - loss: 0.1287 - accuracy: 0.9509
Epoch 5: val_accuracy improved from 0.91233 to 0.94567, saving model to /content/drive/MyDrive/MP-2/vgg16
INFO:tensorflow:Assets written to: /content/drive/MyDrive/MP-2/vgg16/assets
188/188 [=====] - 301s 2s/step - loss: 0.1287 - accuracy: 0.9509 - val_loss: 0.1487 - val_accuracy: 0.9457

```

Fig (5.11) BACK-END: Optimal accuracy & validation accuracy for vgg16.

```

Epoch 4/10
188/188 [=====] - ETA: 0s - loss: 0.7734 - accuracy: 0.7115
Epoch 4: val_accuracy improved from 0.74500 to 0.75767, saving model to /content/drive/MyDrive/MP-2/resnet50
INFO:tensorflow:Assets written to: /content/drive/MyDrive/MP-2/resnet50/assets
188/188 [=====] - 300s 2s/step - loss: 0.7734 - accuracy: 0.7115 - val_loss: 0.5712 - val_accuracy: 0.7577

```

Fig (5.12) BACK-END: Optimal accuracy & validation accuracy for resnet50

6.TESTING & RESULTS

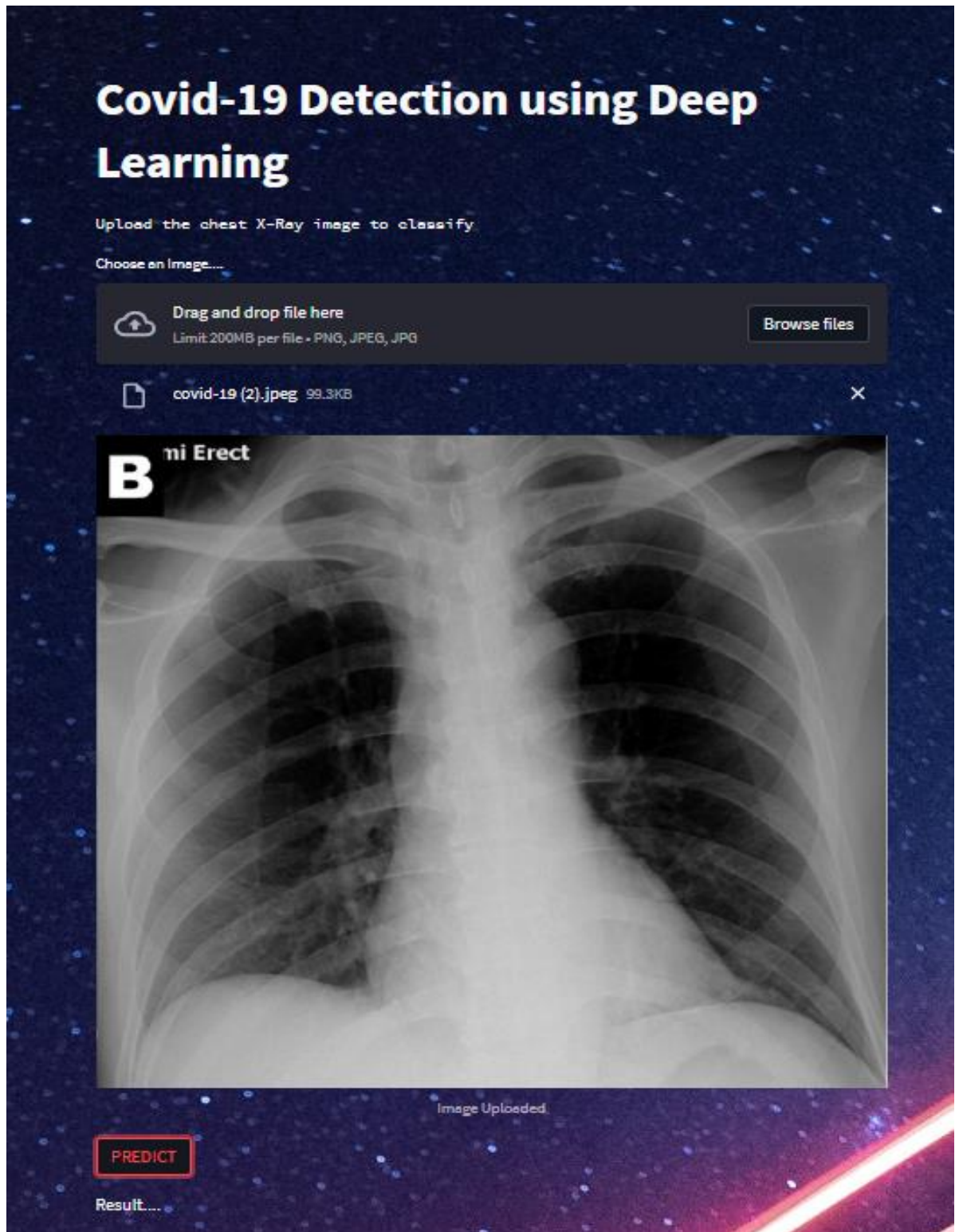


Fig (6.1) USER INTERFACE: The practitioner may upload x-ray images manually from his device.



Fig (6.2) COVID TEST CASE: The person with covid may also have viral pneumonia traces, the output is completely dependent on the probabilities that vgg16 and resnet50 produces.

vgg16 predicted :Normal

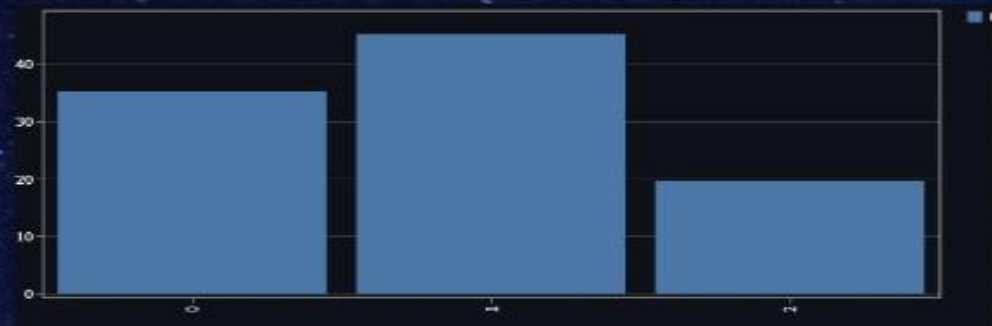
resnet50 predicted :Normal

vgg16 scores :

Covid : 35.19926965236864%

Normal : 45.20840989278428%

Viral Pneumonia : 19.59431767463684%

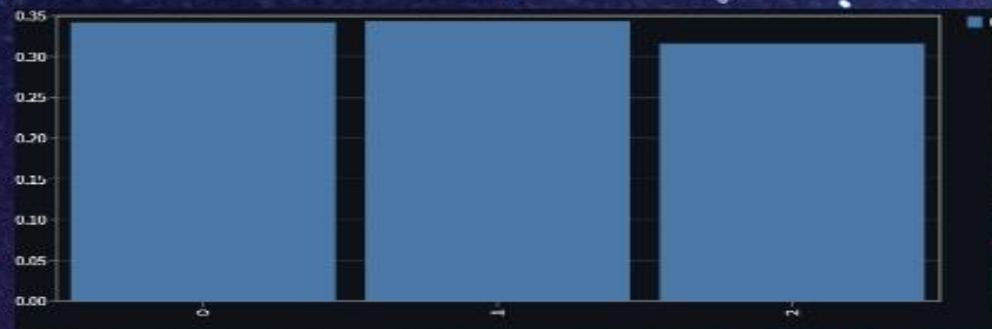


resnet50 scores :

Covid : 34.135329723358154%

Normal : 34.30943191051483%

Viral Pneumonia : 31.555241346359253%



Normal, no additional diagnosis is required!!

Fig (6.3) NORMAL TEST CASE: the person who's normal may also have viral pneumonia traces or vice versa along with a possibility that they are completely normal, the output is completely dependent on the probabilities that vgg16 and resnet50 produce.

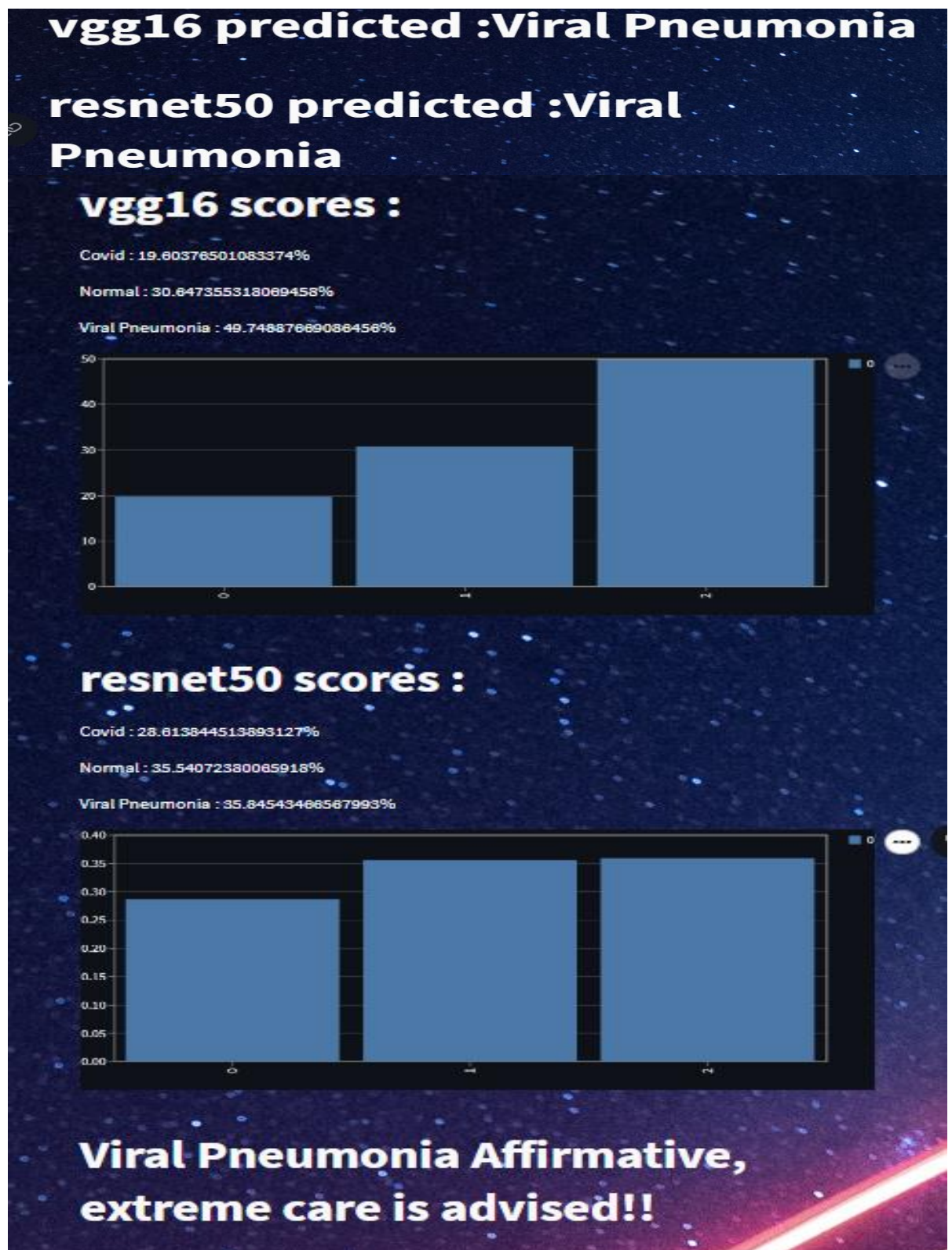


Fig (6.4) PNEUMONIA TEST CASE: The person whose output is “viral pneumonia” is advised to take extreme care.

7. CONCLUSION & FUTURE SCOPE

CONCLUSION:

The interaction of expectation begins with scaling and augmentation of Images, importing the CNN model structure and compiling the model, and validating on test data. On the Data set, the best-case accuracy acquired from VGG16 is 0.9610 and that of RESNET50 is 0.7653. The accompanying ends are reached after examination that those patients whose predicted outcome was covid are most probably affected with pneumonia too and were suggested to take extreme care. More often than not, those patients who have an almost equal probability of pneumonia and normal were suggested to take additional diagnosis. Some other equal probability occurring attributes were handled through precise precautions.

FUTURE SCOPE:

Modern algorithms are available in the market to retrieve optimal accuracy with less loss in the upcoming future. The input type can be further extended to CT scans and MRI scans so that additional diagnosis is possible, MySQL can be implemented on this project to access real-time data from the database. In real-life scenarios, the confidential scans from the hospitals can be retrieved and used for the detection of various diseases.

BIBLIOGRAPHY

- <https://www.kaggle.com/datasets/alifrahman/covid19-chest-xray-image-dataset>
- [https://github.com/junaaidiqbalsyed/Covid_detection_CNN/blob/master/Covid_detection_using_chest_X_Ray_\(Day_2\).ipynb](https://github.com/junaaidiqbalsyed/Covid_detection_CNN/blob/master/Covid_detection_using_chest_X_Ray_(Day_2).ipynb)
- <https://ieeexplore.ieee.org/document/9299315>
- <https://www.kaggle.com/code/bachrr/detecting-covid-19-in-x-ray-images-with-tensorflow/notebook>
- https://www.youtube.com/watch?v=ejkRh9obVjk&ab_channel=edureka%21