COVID-19 CLASSIFICATION FROM CHEST X-RAY USING DEEP LEARNING MODELS

Submitted in partial fulfilment for the award of the degree of

B-Tech (Information Technology)

by

AKKICHETTY NITHIN SAI (19BIT0223.)

SCHOOL OF INFORMATION TECHNOLOGY & ENGINEERING

April, 2023

DECLARATION

I here by declare that the thesis entitled “A COVID-19 CLASSIFICATION FROM CHEST X-RAY USING THE DEEP LEARNING MODELS” submitted by Akkichetty nithin sai(19BIT0223), for the award of the degree of B-TECH (Information Technology) is a record of bonafide work carried out by me under the supervision of Prof. Tapan Kumar Das. I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

CERTIFICATE

This is to certify that the thesis entitled "A COVID-19 CLASSIFICATION FROM CHEST X-RAY USING THE DEEP LEARNING MODELS" submitted by AKKICHETTY NITHIN SAI (19BIT0223), School of Information Technology & Engineering, Vellore Institute of Technology, Vellore for the award of the degree B-Tech (Information Technology) is a record of bonafide work carried out by him/her under my supervision. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The Project report fulfils the requirements and regulations of VELLORE INSTITUTE OF TECHNOLOGY, VELLORE and in my opinion meets the necessary standards for submission.

Signature of the Guide **Signature of the HoD**

Internal Examiner **External Examiner**

Date: ....>

CERTIFICATE BY THE EXTERNAL GUIDE

This is to certify that the project report entitled "A COVID-19 CLASSIFICATION FROM CHEST X-RAY USING THE DEEP LEARNING MODELS" submitted by AKKICHETTY NITHIN SAI (19BIT0223) to Vellore Institute of Technology in partial fulfilment of the requirement for the award of the degree of B-Tech (Information Technology) is a record of Bonafide work carried out by him under my guidance. The project fulfils the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

<Signature of the External Supervisor>

EXTERNAL SUPERVISOR

<Title of the Supervisor >

<Full address of the Institution / organization with e-mail id, phone no.>

<Seal of the Institution / Organization>

ABSTRACT

The COVID-19 pandemic is causing a major outbreak in more than 150 countries around the world, having a severe impact on the health and life of many people globally. One of the crucial step in fighting COVID-19 is the ability to detect the infected patients early enough, and put them under special care. Detecting this disease from radiography and radiology images is perhaps one of the fastest ways to diagnose the patients. Some of the early studies showed specific abnormalities in the chest radiograms of patients infected with COVID-19. Inspired by earlier works, we study the application of deep learning models to detect COVID-19 patients from their chest radiography images. In many of the research papers I have gone through the main novel problem statement I found is further analysis should be done on the larger dataset so here I had experimented with the large dataset which is available in kaggle and used that complete dataset in my project to further examine the performance of the models on the larger dataset

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to <Prof.Tapan kumar das.>,

<Designation, School name, Vellore Institute of Technology>, for him/his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavour. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectualand expert in the field of <DEEP LEARNING>.

I would like to express my gratitude to DR.G.VISWANATHAN, Chancellor VELLORE INSTITUTE OF TECHNOLOGY, VELLORE, MR. SANKAR VISWANATHAN, DR. SEKAR VISWANATHAN, MR.G V SELVAM, Vice – Presidents VELLORE INSTITUTE OF TECHNOLOGY, VELLORE, DR.

RAMBABU KODALI, Vice – Chancellor, DR. S. NARAYANAN, Pro-Vice Chancellor and Dr. S. Sumathy, Dean, School of Information Technology & Engineering (SITE), for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. USHA DEVI G, HoD/Professor, all teaching staff and members working as limbs of our university for their not-self-catered enthusiasm coupled with timely encouragementsshowered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parentsfor their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore

Date: Name of the student

<AKKICHETTY NITHINSAI>

TABLE OF CONTENTS

CHAPTER 4

ANALYSIS & DESIGN

CHAPTER 5

IMPLEMENTATION &TESTING

CHAPTER 6

CONCLUSION

REFERENCE

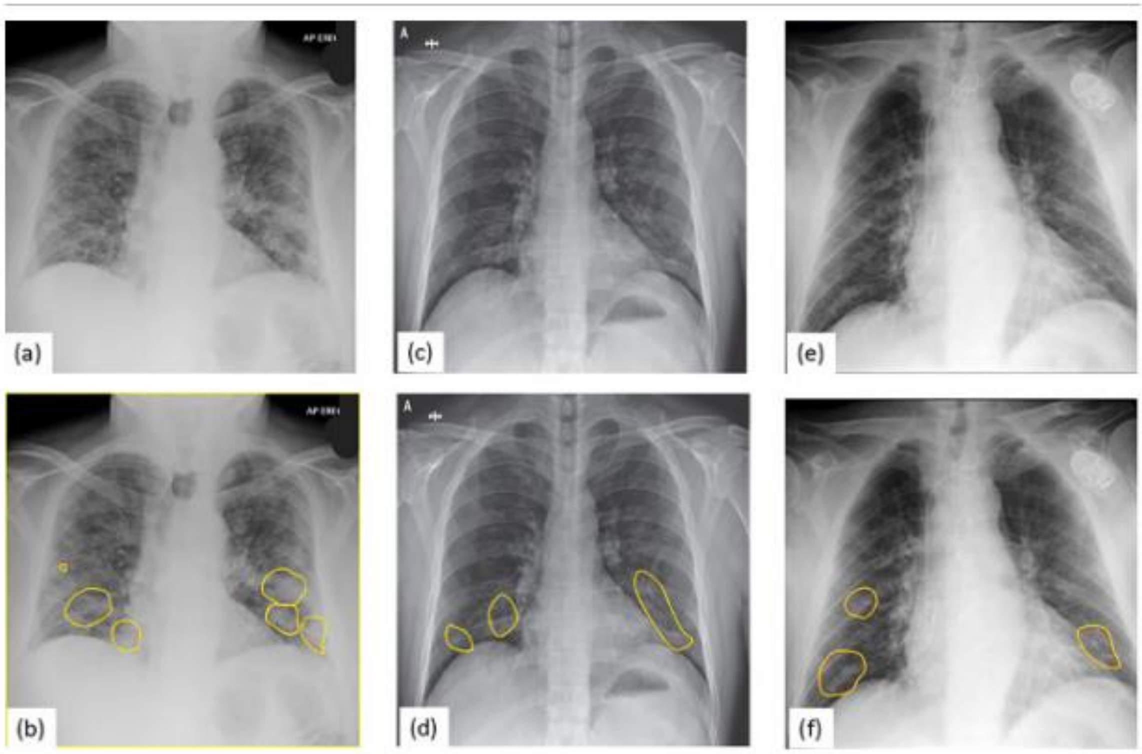
Chapter 1

INTRODUCTION

Since December 2019, a novel corona-virus (SARS-CoV-2) has spread from Wuhan to the whole China, and many other countries. By April 18, more than 2 million confirmed cases, and more than 150,000 deaths were reported in the world. Due to unavailability of therapeutic treatment or vaccine for novel COVID-19 disease, early diagnosis is of real importance to provide the opportunity of immediate isolation of the suspected person and to decrease the chance of infection to healthy population. Reverse transcription polymerase chain reaction (RT-PCR) or gene sequencing for respiratory or blood specimens are introduced as main screening methods for COVID-19. However, total positive rate of RT-PCR for throat swab samples is reported to be 30 to 60%, which accordingly yields to un-diagnosed patients, which may contagiously infect a huge population of healthy people. Chest radiography imaging (e.g., X-ray or computed tomography (CT) imaging) as a routine tool for pneumonia diagnosis is easy to perform with fast diagnosis. Chest CT has a high sensitivity for diagnosis of COVID-19 and X-ray images show visual indexes correlated with COVID-19. The reports of chest imaging demonstrated multi lobar involvement and peripheral airspace opacities. The opacities most frequently reported are ground-glass (57%) and mixed attenuation (29%). During the early course of COVID-19, ground glass pattern is seen in areas that edges the pulmonary vessels and may be difficult to appreciate visually .Asymmetric patchy or diffuse airspace opacities are also reported for COVID-19. Such subtle abnormalities can only be interpreted by expert radiologists. Considering huge rate of suspected people and limited number of trained radiologists, automatic methods for identification of such subtle abnormalities can assist the diagnosis procedure and increase the rate of early diagnosis with high accuracy.

Artificial intelligence (AI)/machine learning solutions are potentially powerful tools for solving such problems.

So far, due to the lack of availability of public images of COVID-19 patients, detailed studies reporting solutions for automatic detection of COVID-19 from X-ray (or Chest CT) images are not available. Recently a large dataset of COVID-19 X-ray images was collected and made it available in the kaggle, which made it possible for AI researchers to train deep learning models to perform automatic COVID-19 diagnostics from X-ray images. These images were extracted from academic publications reporting the results on COVID-19 X-ray.



A machine a learning framework was employed to predict COVID-19 from Chest X-ray images. Unlike the classical approaches for medical image

classification which follow a two-step procedure, we use an end-to-end deep learning framework which directly predicts the COVID-19 disease from raw images without any need of feature extraction. Deep learning based models (and more specifically convolutional neural networks (CNN)) have been shown to outperform the classical AI approaches in most of computer vision and and medical image analysis tasks in recent years, and have been used in a wide range of problems from classification, segmentation, face recognition, to super-resolution and image enhancement

Here, we have trained five popular deep learning models with the larger dataset to experiment with it which we have achieved decent results in several tasks during recent years on kaggle dataset, and analyze overall accuracy performance for COVID-19,viral pneumonia,lung opacity,normal detection. Since so far this is the large number of X-ray images publicly available for the COVID-19 class, we had simply train some of the models from scratch.

The main contributions of this project is

.Here we are using a large dataset of 21000 images for COVID-19 classification from Chest X-ray images. The images are in 4 classes that are COVID-19,normal,lungopacity,viral pneumonia are labeled by a board-certified radiologist, and only those with a clear sign are used for testing purpose.

• as of now we trained five promising deep learning models on this larger dataset, and evaluated their performance on a test set of images. Our best performing model achieved a testing accuracy of 89% and training accuracy graph of 92% and the validation accuracy of 88% for a mobilenet model,

• We also provided a detailed experimental analysis on the performance of these models, in terms of accuracy graphs,test accuracy,confusion matrix and roc curve ,

And here we have also designed and modified ann model which is feed forward neural network with one flatten layer and four hidden layers of 256,128,64,32 and final output layer of 4 neurons and achieved the test accuracy of 77

• we also provide a website for the users to check their prediction result and also user had a option to decide which algorithm to choose

1.1 Background

The COVID-19 pandemic, caused by the SARS-CoV-2 virus, has affected the world in an unprecedented manner since its emergence in late 2019. The virus primarily spreads through respiratory droplets and close contact with infected individuals. The symptoms of COVID-19 range from mild to severe, with some individuals being asymptomatic. Chest X-rays have been widely used in the diagnosis of COVID-19 due to its ease of availability, low cost, and rapid results.Chest X-rays are a useful diagnostic tool for COVID-19 as the virus primarily affects the respiratory system, causing inflammation in the lungs. The characteristic radiological findings of COVID-19 on chest X-rays include bilateral, peripheral, and ground-glass opacities in the lungs. These opacities are due to the accumulation of fluid and cellular debris in the lungs as a result of the viral infection. The use of chest X-rays in the detection of COVID-19 has been particularly useful in areas with limited resources and inadequate access to reverse transcription-polymerase chain reaction (RT-PCR) testing, which is considered the gold standard for COVID-19 diagnosis. Chest X-rays are also useful in the follow-up of patients with COVID-19, as they can monitor disease progression and response to treatment. Several studies have evaluated the sensitivity and specificity of chest X-rays in the diagnosis of COVID-19. One study reported a sensitivity of 69% and a specificity of 71% for chest X-rays in the diagnosis of COVID-19 when compared to RT-PCR testing. Another study reported a sensitivity of 58% and a specificity of 83% for chest X-rays in the diagnosis of COVID-19. Despite its usefulness in the diagnosis of COVID-19, chest X-rays have several limitations. The radiological findings of COVID-19 on chest X-rays are non-specific and can be seen in other respiratory infections, such as influenza and pneumonia. Chest X-rays are also not as sensitive as RT-PCR testing and may produce false-negative results in some cases. In conclusion, chest X-rays are a useful diagnostic tool in the detection of COVID-19, particularly in areas with limited resources and inadequate access to RT-PCR testing. However, they should be used in conjunction with other diagnostic tools and clinical evaluation to confirm the diagnosis of COVID-19.Having reviewed the related work, it is evident that despite the success of deep learning in the detection of Covid-19 from CXR and CT images, dealing with class imbalance, and also many people explored only with small dataset size . In this research, we aimed to extend the development of automated multi-class classification models based on chest X-ray images. For that, we used a large dataset which is publicly available at kaggle and present and try to i explore that and deployed that explored models into a web application to make it a real time application

1.2 PROBLEM DEFINITION

The covid-19 has impacted many lives during the first as well as the second wave the main problem was faced was due to lack of covid-19 test kits their are so many people out there not even did the check up And also some other people can't able to go to the hospitals due to the large queue in the hospitals and also senior citizens can't get their check up done So our main motive is to provide the prediction result faster by with hassle free process and also by just simply uploading the x-rays in the portal

1.3 OBJECTIVE

One of the critical factors behind the rapid spread of COVID-19 pandemic is a lengthy clinical testing time. So our main objective is to make the system that provides the best prediction result of covid-19 The imaging tool, such as Chest X-ray (CXR), can speed up the identification process. Therefore, our objective is to develop an automated CAD system for the detection of COVID-19 samples from healthy ,pneumonia and lungopacity cases using CXR images

1.4 SCOPE OF THE PROJECT

It can be used for the future researchers to analyse this data for still more larger dataset in the future and also It will be used as best alternative for the hospital covid-19 detection test because here we are making a website to it like we can able to know it before only by without visiting the hospital And it also provides the accurate results at most of the time while detecting the covid-19 And it will also be convenient for the most of the people by simply uploading the x-ray in the web application and by that easily knowing the results

Chapter 2

LITERATURE SURVEY

Title	Author	Algorithm used	Advantages	Disadvantages
CodnNet: A lightweight CNN architecture for detection of COVID-19 infection	JingdongYanga1 LeiZhanga1 XinjunTangb ManHanc	CNN	•Adding Focus layer and modifying the pooling layer to make all features reusable. •An efficient depthwise separable convolution is used to improve the classification performance. •The proposed model can save bandwidth and reduce costs of storage for large datasets.	•However, the model is not suitable for all types of lesion prediction •However, the classification performance was likely to degrade when the sample distribution changed
LCSB-inception: Reliable and effective light-chroma separated branches for Covid-19 detection from chest X-ray images	Chiagoziem C.Ukwuomaa QinZhiguanga Victor Kwaku AgbesibChukwuebuka J.EjiyiaOlusola BamisileclJeoma A.ChikwendudWilfried T.BoledMd AltabHossine	CNN	•This study introduced the Global second-order pooling at the last two convolutional blocks for comprehensive image information at the last stages of deep ConvNets, in contrast to previous approaches that only employ the max-pooling at the end of the network.	•The RT-PCR-based methods were unable to sustain the high demand for results due to a lack of testing kits, and inaccurate, and divergent results

Title	Author	Algorithm used	Advantages	Disadvantages
Audio texture analysis of COVID-19 cough, breath, and speech sounds	Garima sharma Karthikeyan umapathy Srikrishnan	Many ai based algorithm is used mainly it uses the cnn model for the analysis and also other algorithms like lstm and vgg has been used	•This provide the faster results •It is also provides the better accuracy of 97% •And during the classification process it also provides the higher accuracy	•The main challenge among these methods is the unavailability of the large data sets and image acquisition process where the person needs to visit a clinic for getting a CT scan or an X-ray image.
Multi-scale causality analysis between COVID-19 cases and mobility level using ensemble empirical mode decomposition and causal decomposition	Jung hoo choon Dong kyu kim Eui jin Kim	EMD,EEMD,VMD AND FOURIER ANALYSIS,LINEAR REGRESSION	•It has a faster analysis rate •It has a better performance in finding the outbreaks compared to other algorithms	•It may not provide the accurate results analysis when the larger dataset was there
COVID-19 Trend Analysis using Machine Learning Techniques	Abhishek Jaglan Dakshan trehan Priyanjali singh	Linear regression	•It can able to easily fetch the data from the larger datasets by without errors	•The tested and the trained data may not give the correct information all the time due to the less accuracy rate by this model

Title	Author	algorithm used	advantages	Disadvantages
Forecast and prediction of Covid-19 using machine learning	Deepak painuli Divya Mishra Suyansh Bharadwaj Mayank Agrawal	RSME MODEL,ARIMA MODEL AND LINEAR REGRESSION	• It has a faster analysis rate • It has a better performance in finding the outbreaks compared to other algorithms	• It may not provide the accurate results analysis when the larger dataset was there
Development of a web based covid portal and marketplace	Dipta voumick Israt jahan min Prince deb Sourav sutradhar	CNN LSTM	• It has a better accuracy performance compared to other algorithms • This portal can be easily able to find the patient details and blood details and it can also able to find out the health centre details with accurate information	• Sometimes it may not provide the accurate results due to larger data sets

Title	Author	Algorithm used	Advantages	Disadvantages
Acute portal vein thrombosis with COVID-19 and cirrhosis	Yusuke miyazato Masahiro ishikane Makato inado	Supervised machine learning models like linear regression svm,lstm is been used	• It has a faster detection rate of covid-19 and thrombosis • And also have high performance rates	• Some times it was too slow to handle the data for larger dataset
design and development of hybrid optimisation enabled deep learning model for Covid-19 detection with comparative analysis with donna biat-gru xg boost	Jawed ahmed dhar Kamal kr srivastava Sajjad ahmed lone	DCNN,BIAT-GRUM,XGBoost	• A novel Hybrid (JHBO-DNFN) is introduced for Covid-19 prediction by audio signal. • The pre-processing process is employed for eliminating the noises present in input sample.	• The testing accuracy Was somewhat low around 90% compared to other algorithms
SVD-CLAHE boosting and balanced loss function for Covid-19 detection from an imbalanced Chest X-Ray dataset	MrinalTyagi bVibhutiBansal bVikasJain	SVD-CLAHE boosting ,cnn model,VGG-19, RESNET-50 model	• The detection rate for this was average compared to many algorithms with the accuracy rate of 95% • And has better analysis rate of identifying the covid-19 data	• However, RUS does not work efficiently for a highly-imbalanced dataset, since large number of randomly excluded images may contain significant features for the classification task.

Chapter 3

HARDWARE & SOFTWARE REQUIREMENTS

3.1H/W Configuration:

- **Processor - I3/Intel Processor**
- Hard Disk -160GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- RAM - 8Gb minimum

3.2S/W Configuration:

- Operating System : Windows 7/8/10
- Server side Script : Python, Anaconda
- IDE : vscode
- Libraries Used :Sklearn,Pandas,Numpy,matplotlib, opencv, Tensorflow, Keras, imutils,pillow, mysql.connector

- Dataset : covid19-radiography-dataset

  Technology : Python 3.6+

Chapter 4

Analysis and design

EXISTING METHOD

Existing system for detecting COVID-19 using the aforementioned virus and antibody testing modalities is time-consuming and requires additional resources and approval, which can be a luxury in many developing communities. Hence, at many medical centers, the test kits are often unavailable. Due to the shortage of kits and false-negative rate of virus and antibody tests, the authorities in Hubei Province, China momentarily employed radiological scans as a clinical investigation for COVID19

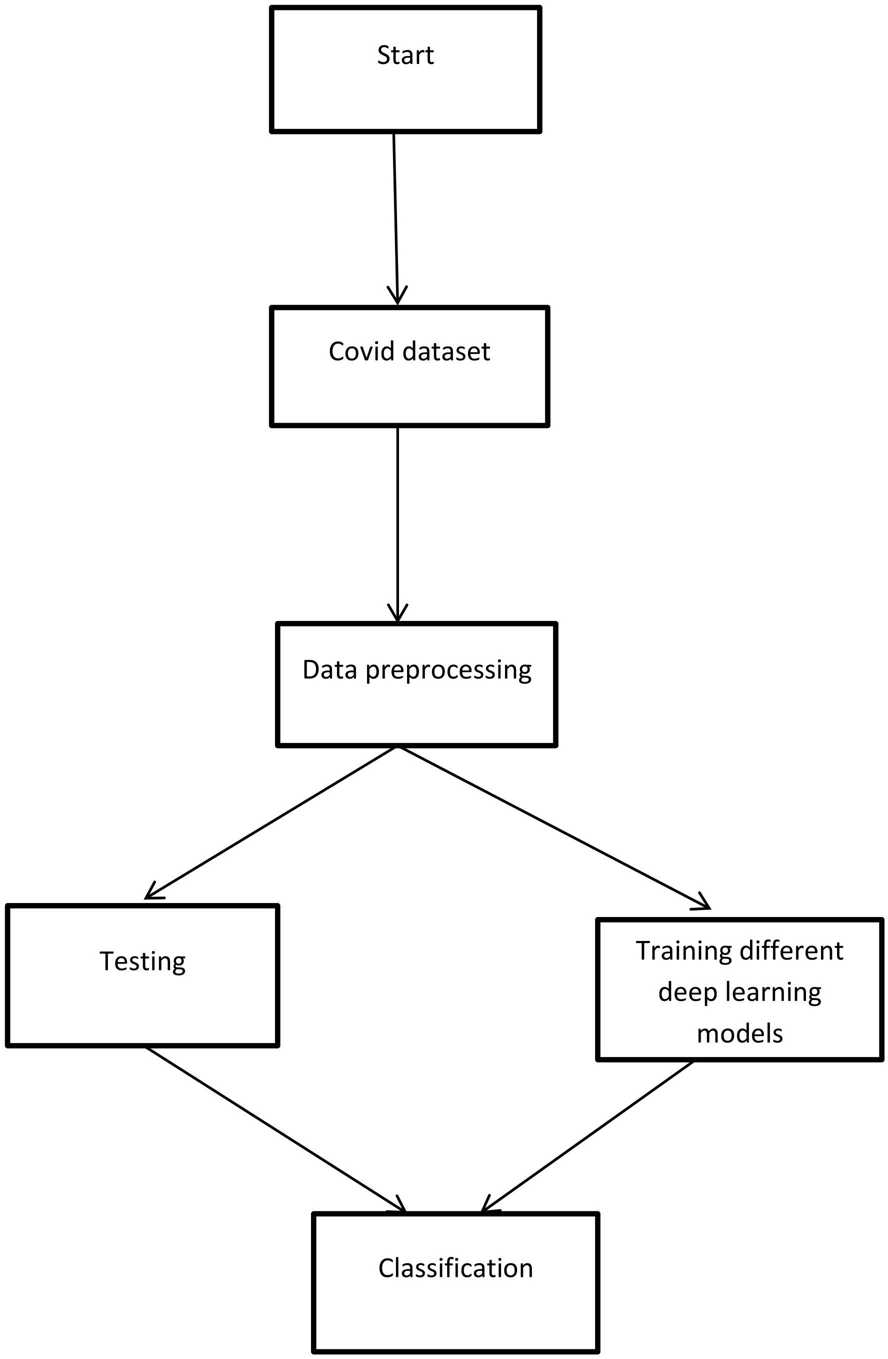
4.1 PROPOSED METHOD

In proposed system, It's worth noting that Alex Net is a relatively old model and there may be more modern architectures that perform better on this task. However, the general steps for training and deploying the model would be similar regardless of the architecture used. Additionally, you should always consider the ethical implications of deploying such a model and ensure that appropriate measures are in place to protect patient privacy and ensure accuracy and fairness. In proposed system another algorithms are also used i.e., Resnet, ANN,inceptionresnetv2 and mobilenet.

Advantages:

- Cheaper to operate.
- It can be scaled up quickly.
- Time minimising.

4.2 PROPOSED ARCHITECTURE



4.3 MODULES

System

User

1.System:

1.1 Create Dataset:

Using the larger datasets of x-ray images which has four class like covid-19,normal, lung_opacity,viral pneumonia and classifying it accordingly and, training our models.

1.2 Preprocessing:

Resizing, gray scaling and reshaping the images into appropriate format to train our model. The final dataset is split into training,testing and validation dataset with test size of 10% and validation size of 30%.

1.3Training:

Use the pre-processed training dataset to train the model using a different deep learning models.

2.1 About-Project

In this application, we have successfully created an application which takes in an X-ray image and classify it accordingly.

2.2 Upload Image

The user has to upload an image which needs to be tested for covid-19.

2.3 Prediction

The results of our model is displayed as the uploaded person chest x-ray image classifying it according to the disease.

4.4 SYSTEM DESIGN

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful inssss the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

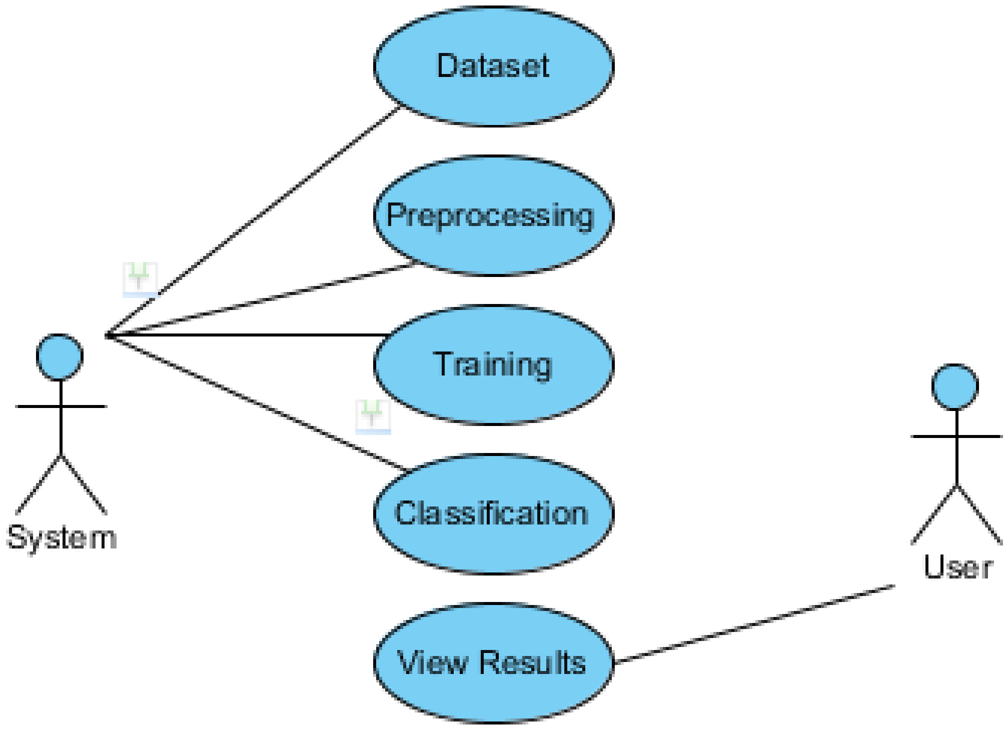
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.

5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

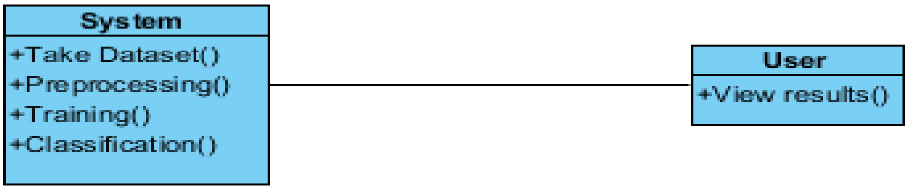
USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



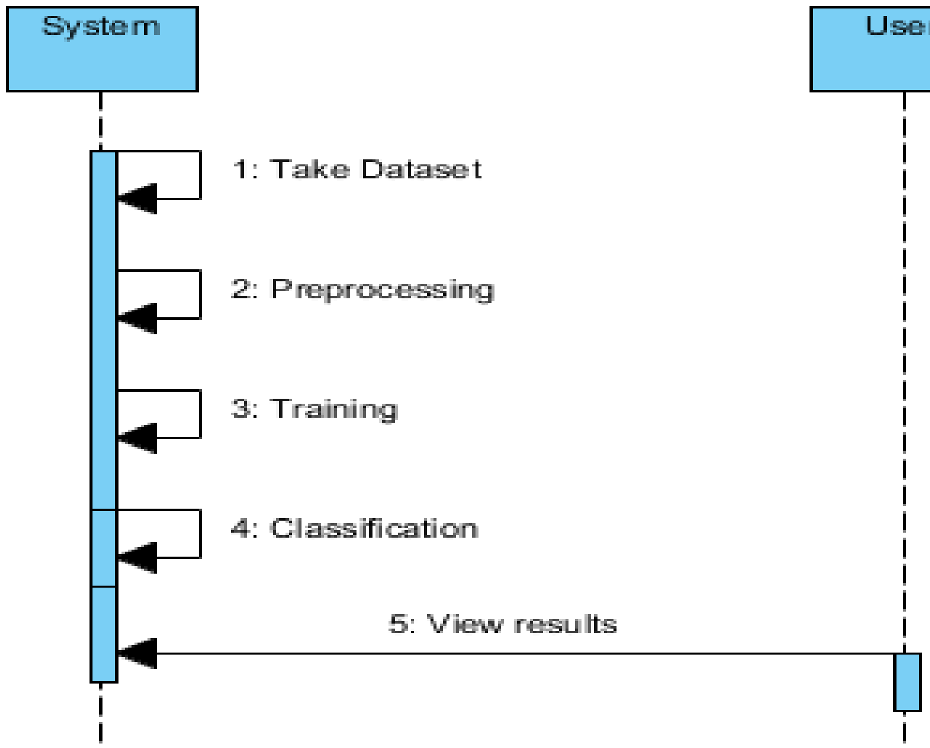
CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



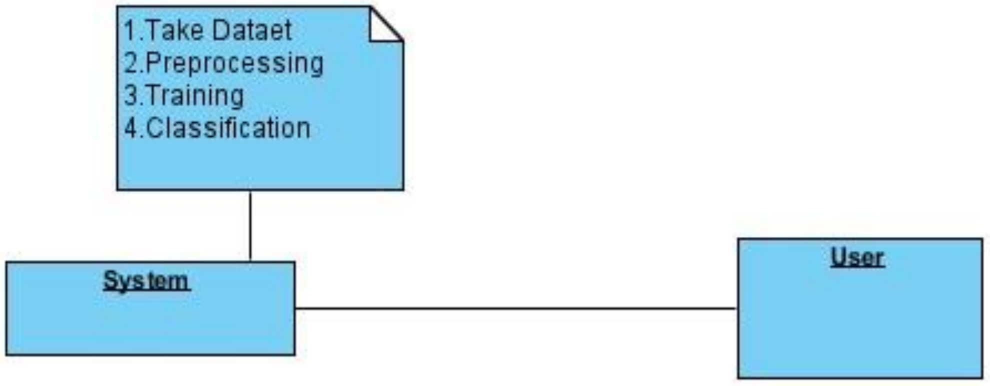
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



Collaboration Diagram:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

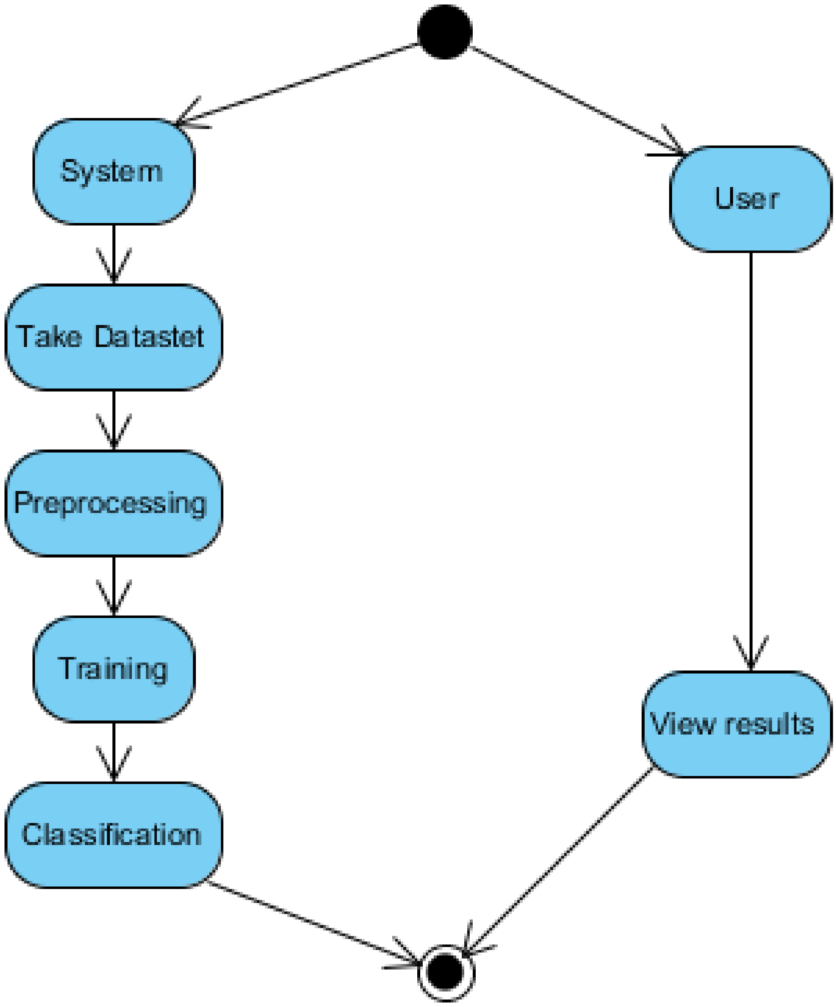


DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



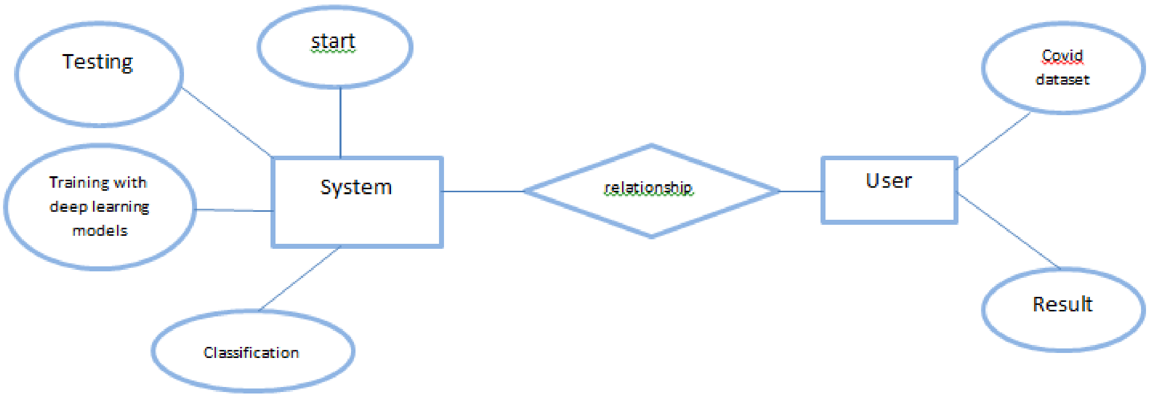
Component diagram,

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

ER Diagram:

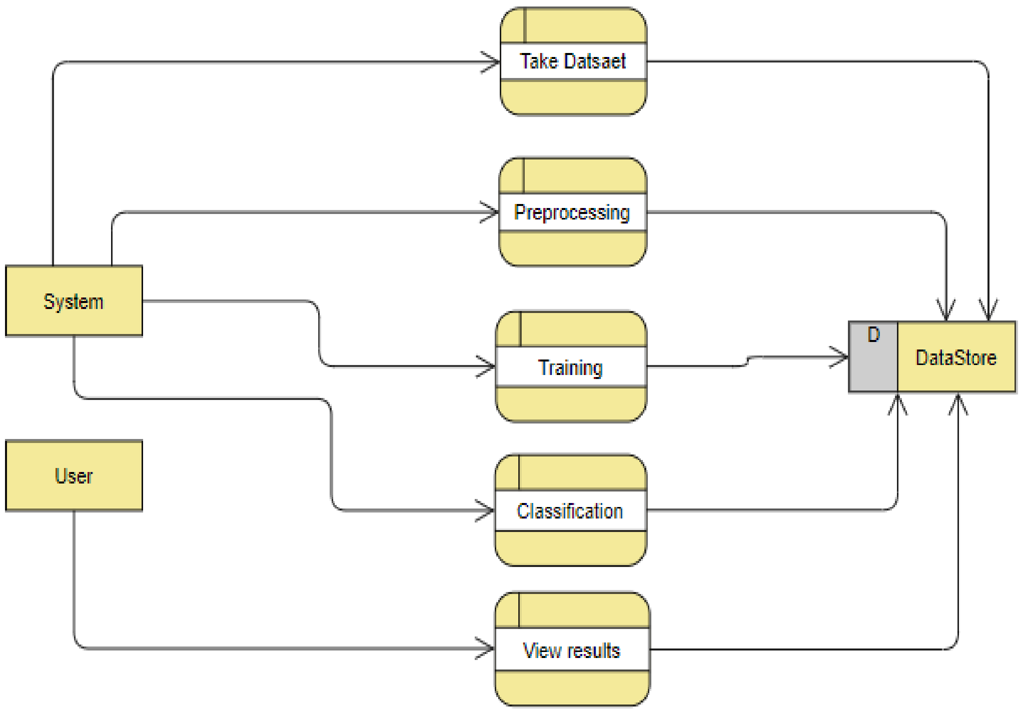
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

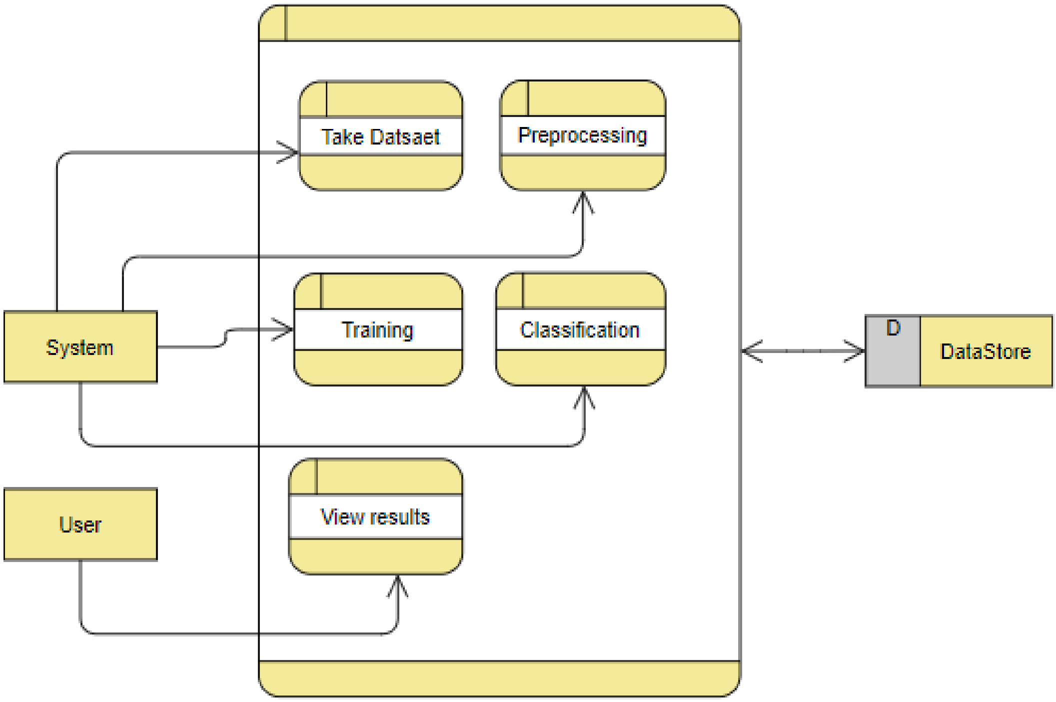
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.



DFD Diagram:

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.





Chapter 5

Implementation and testing

5.1 Dataset used

https://www.kaggle.com/datasets/preetviradiya/covid19-radiography-dataset

5.2 Code implementation

<u>Alexnet</u>

Importing all the libraries

import os

import numpy as np

```
import tensorflow as tf

import matplotlib.pyplot as plt

from keras import regularizers

from matplotlib import pyplot as plt

import tensorflow as tf

from keras.applications.inception_resnet_v2 import InceptionResNetV2

# from tf.keras.losses import sparse_categorical_crossentropy

from keras.losses import sparse_categorical_crossentropy

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout, Flatten,Conv2D,
MaxPooling2D,BatchNormalization, GlobalAveragePooling2D

from keras.preprocessing.image import ImageDataGenerator

from keras.layers import Input, Flatten, Dense, Dropout, BatchNormalization

from tensorflow.keras.optimizers import Adam, SGD

from tensorflow.keras.applications import ResNet50, MobileNet

from keras.models import Model

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import confusion_matrix, classification_report
```

```
import seaborn as sns

import pandas as pd

import shutil

import numpy as np

from sklearn.metrics import precision_recall_fscore_support
```

Datapreprocessing

```
img_height, img_width = 128, 128

batch_size = 12

train_data_dir = "Dataset/"

test_data_dir = "Test/"



# Create test directory and move 10% of the data to this directory

if not os.path.exists(test_data_dir):

    os.makedirs(test_data_dir)

    for class_name in os.listdir(train_data_dir):

        class_dir = os.path.join(train_data_dir, class_name)

        test_class_dir = os.path.join(test_data_dir, class_name)

        os.makedirs(test_class_dir)
```

```
        files = os.listdir(class_dir)

        n_test = int(len(files) * 0.1)

        test_files = files[:n_test]

        for test_file in test_files:

            src = os.path.join(class_dir, test_file)

            dst = os.path.join(test_class_dir, test_file)

            shutil.move(src, dst)


# Create data generators



train_datagen = ImageDataGenerator(rescale=1./255,

                                   shear_range = 0.2,

                                   zoom_range = 0.2,

                                   horizontal_flip = True,

                                   validation_split=0.3

                                   )

train_generator = train_datagen.flow_from_directory(train_data_dir,

                                                    target_size=(img_height,img_width),
```

```
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        subset='training')
valid_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
test_datagen = ImageDataGenerator(rescale=1./255,shear_range = 0.2,
                                  zoom_range = 0.2,
                                  horizontal_flip = True,)
test_generator = test_datagen.flow_from_directory(test_data_dir,
                                        target_size=(img_height,img_width),
                                        batch_size=batch_size,
                                        class_mode='categorical',

                                        )
```

model building

```
model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),
padding='same'))

model.add(Activation('relu'))

# Pooling

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))

# Batch Normalisation

model.add(BatchNormalization())


# Passing it to a dense layer

model.add(Flatten())

# 1st Dense Layer

model.add(Dense(4096, input_shape=(img_width,img_height,3)))

model.add(Activation('relu'))

# Add Dropout to prevent overfitting

model.add(Dropout(0.4))

# Batch Normalisation

model.add(BatchNormalization())


# 2nd Dense Layer

model.add(Dense(4096))

model.add(Activation('relu'))

# Add Dropout

model.add(Dropout(0.4))

# Batch Normalisation
```

```
model.add(BatchNormalization())

# 3rd Dense Layer
model.add(Dense(1000))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# Batch Normalisation
model.add(BatchNormalization())

# Output Layer
model.add(Dense(4))
model.add(Activation('softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 96)        34944
_________________________________________________________________
activation (Activation)      (None, 30, 30, 96)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 96)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 15, 15, 96)        384
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 15, 15, 256)       2973952
_________________________________________________________________
activation_1 (Activation)    (None, 15, 15, 256)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 384)         885120
_________________________________________________________________
activation_2 (Activation)    (None, 8, 8, 384)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 384)         1536
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 384)         1327488
_________________________________________________________________
activation_3 (Activation)    (None, 8, 8, 384)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 384)         1536
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 256)         884992
```

```
_________________________________________________________________
dense (Dense)                (None, 4096)              16781312
_________________________________________________________________
activation_5 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
batch_normalization_5 (Batch (None, 4096)              16384
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
activation_6 (Activation)    (None, 4096)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 4096)              16384
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
activation_7 (Activation)    (None, 1000)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
batch_normalization_7 (Batch (None, 1000)              4000
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 4004
_________________________________________________________________
activation_8 (Activation)    (None, 4)                 0
=================================================================
Total params: 43,812,396
Trainable params: 43,791,260
Non-trainable params: 21,136
_________________________________________________________________
```

Optimizing and fitting the model

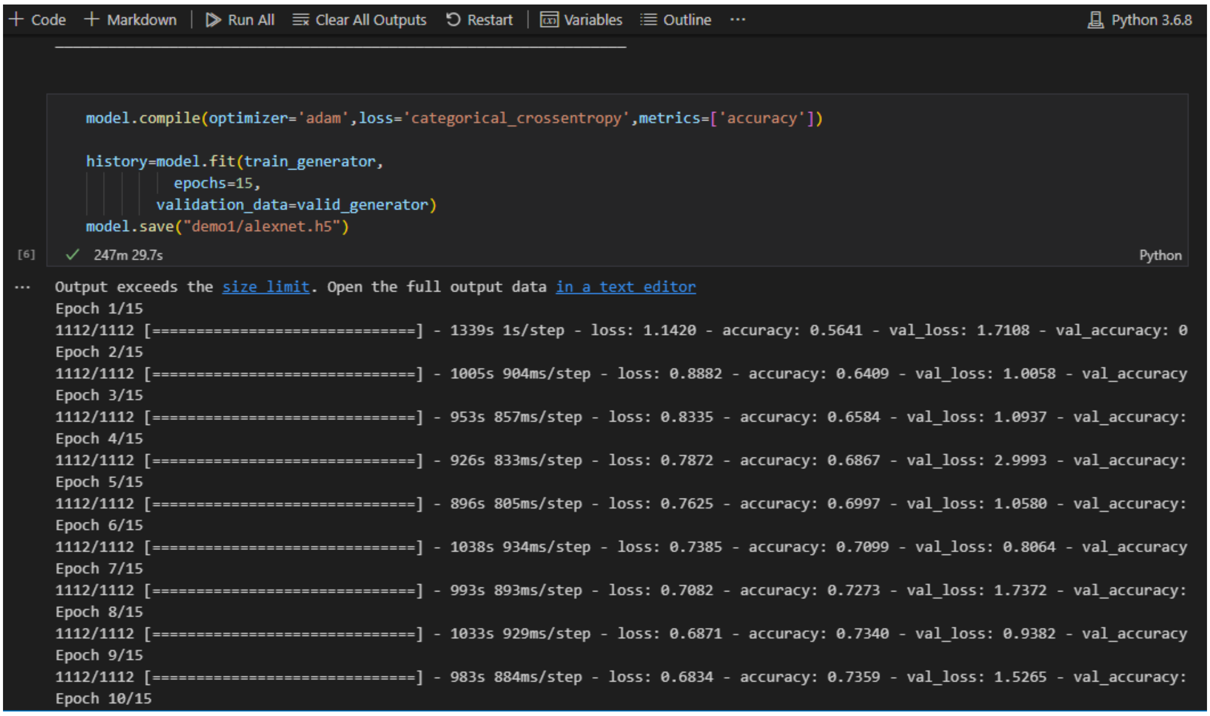
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

history=model.fit(train_generator,

epochs=15,

validation_data=valid_generator)

model.save(“demo1/alexnet.h5")



<u>resnet50</u>

importing all the libraries

import os

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

```
from keras import regularizers
from matplotlib import pyplot as plt
import tensorflow as tf
from keras.applications.inception_resnet_v2 import InceptionResNetV2
# from tf.keras.losses import sparse_categorical_crossentropy
from keras.losses import sparse_categorical_crossentropy
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten,Conv2D,
MaxPooling2D,BatchNormalization, GlobalAveragePooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Input, Flatten, Dense, Dropout, BatchNormalization
# from keras.optimizers import Adam, SGD
from tensorflow.keras.applications import ResNet50, MobileNet
from keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import pandas as pd
import shutil
import numpy as np
from sklearn.metrics import precision_recall_fscore_support
```

data preprocessing

```
img_height, img_width = 128, 128
batch_size = 12
```

```
train_data_dir = "Dataset/"
test_data_dir = "Test/"

# Create test directory and move 10% of the data to this directory
if not os.path.exists(test_data_dir):
    os.makedirs(test_data_dir)
    for class_name in os.listdir(train_data_dir):
        class_dir = os.path.join(train_data_dir, class_name)
        test_class_dir = os.path.join(test_data_dir, class_name)
        os.makedirs(test_class_dir)
        files = os.listdir(class_dir)
        n_test = int(len(files) * 0.1)
        test_files = files[:n_test]
        for test_file in test_files:
            src = os.path.join(class_dir, test_file)
            dst = os.path.join(test_class_dir, test_file)
            shutil.move(src, dst)

# Create data generators

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
```

```
                    validation_split=0.3
                    )
train_generator = train_datagen.flow_from_directory(train_data_dir,
                                        target_size=(img_height,img_width),
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        subset='training')
valid_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
test_datagen = ImageDataGenerator(rescale=1./255,shear_range = 0.2,
                    zoom_range = 0.2,
                    horizontal_flip = True,)
test_generator = test_datagen.flow_from_directory(test_data_dir,
                                        target_size=(img_height,img_width),
                                        batch_size=batch_size,
                                        class_mode='categorical',

                                        )
```

model building

```
base_model=ResNet50(include_top=False,weights='imagenet')
```

```
x=base_model.output

x=GlobalAveragePooling2D()(x)

x=Dense(1024,activation='relu')(x)

predictions=Dense(4,activation='softmax')(x)

model=Model(inputs=base_model.input,outputs=predictions)


for layer in base_model.layers:
    layer.trainable=False
```

optimizing and fitting the model

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

history=model.fit(train_generator,
          epochs=25,
          batch_size=16,
          validation_data=valid_generator)


model.save("demo1/resnet.h5")
```

```
app > experiment2.ipynb > base_model=ResNet50(include_top=False,weights='imagenet')
+ Code  + Markdown | Run All  Clear All Outputs  Restart | Variables  Outline  ...          Python 3.6.8
```

```python
base_model=ResNet50(include_top=False,weights='imagenet')
x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x)
predictions=Dense(4,activation='softmax')(x)
model=Model(inputs=base_model.input,outputs=predictions)

for layer in base_model.layers:
    layer.trainable=False

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
history=model.fit(train_generator,
        epochs=25,
        batch_size=16,
        validation_data=valid_generator)

model.save("demo1/resnet.h5")
```

```
[12]  ✓ 348m 38.9s                                                                Python

Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/25
1112/1112 [==============================] - 686s 613ms/step - loss: 1.1046 - accuracy: 0.5471 - val_loss: 1.0090 - val_accuracy:
Epoch 2/25
1112/1112 [==============================] - 643s 578ms/step - loss: 1.0041 - accuracy: 0.5899 - val_loss: 0.9361 - val_accuracy:
Epoch 3/25
1112/1112 [==============================] - 640s 576ms/step - loss: 0.9580 - accuracy: 0.6047 - val_loss: 0.9969 - val_accuracy:
Epoch 4/25
1112/1112 [==============================] - 613s 552ms/step - loss: 0.9212 - accuracy: 0.6161 - val_loss: 0.9735 - val_accuracy:
Epoch 5/25
1112/1112 [==============================] - 619s 557ms/step - loss: 0.8990 - accuracy: 0.6254 - val_loss: 0.8615 - val_accuracy:
Epoch 6/25
1112/1112 [==============================] - 614s 552ms/step - loss: 0.8902 - accuracy: 0.6311 - val_loss: 0.8591 - val_accuracy:
```

```
Output exceeds the size limit. Open the full output data in a text editor
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, None, None,  0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, None, None, 3 0           input_3[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, None, None, 6 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, None, None, 6 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, None, None, 6 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, None, None, 6 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, None, None, 6 0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, None, None, 6 4160        pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, None, None, 6 256         conv2_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, None, None, 6 0           conv2_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, None, None, 6 36928       conv2_block1_1_relu[0][0]
...
Total params: 25,689,988
Trainable params: 2,102,276
Non-trainable params: 23,587,712
```

<u>ANN(modified ann)</u>

importing all the libraries

import os

import numpy as np

```
import tensorflow as tf
import matplotlib.pyplot as plt
from keras import regularizers
from matplotlib import pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import pandas as pd
import shutil
import numpy as np
from sklearn.metrics import precision_recall_fscore_support
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.layers import Dense,GlobalAveragePooling2D
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.applications.mobilenet import preprocess_input
import keras
from matplotlib import pyplot as plt
```

data preprocessing

```
img_height, img_width = 128, 128
batch_size = 64
```

```
train_data_dir = "Dataset2/"
test_data_dir = "Test/"

# Create test directory and move 10% of the data to this directory
if not os.path.exists(test_data_dir):
    os.makedirs(test_data_dir)
    for class_name in os.listdir(train_data_dir):
        class_dir = os.path.join(train_data_dir, class_name)
        test_class_dir = os.path.join(test_data_dir, class_name)
        os.makedirs(test_class_dir)
        files = os.listdir(class_dir)
        n_test = int(len(files) * 0.1)
        test_files = files[:n_test]
        for test_file in test_files:
            src = os.path.join(class_dir, test_file)
            dst = os.path.join(test_class_dir, test_file)
            shutil.move(src, dst)

# Create data generators

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
```

```
                validation_split=0.3
                )
train_generator = train_datagen.flow_from_directory(train_data_dir,
                        target_size=(img_height,img_width),
                        batch_size=batch_size,
                        class_mode='categorical',
                        subset='training')
valid_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
test_datagen = ImageDataGenerator(rescale=1./255,shear_range = 0.2,
                zoom_range = 0.2,
                horizontal_flip = True,)
test_generator = test_datagen.flow_from_directory(test_data_dir,
                        target_size=(img_height,img_width),
                        batch_size=batch_size,
                        class_mode='categorical',

                        )
```

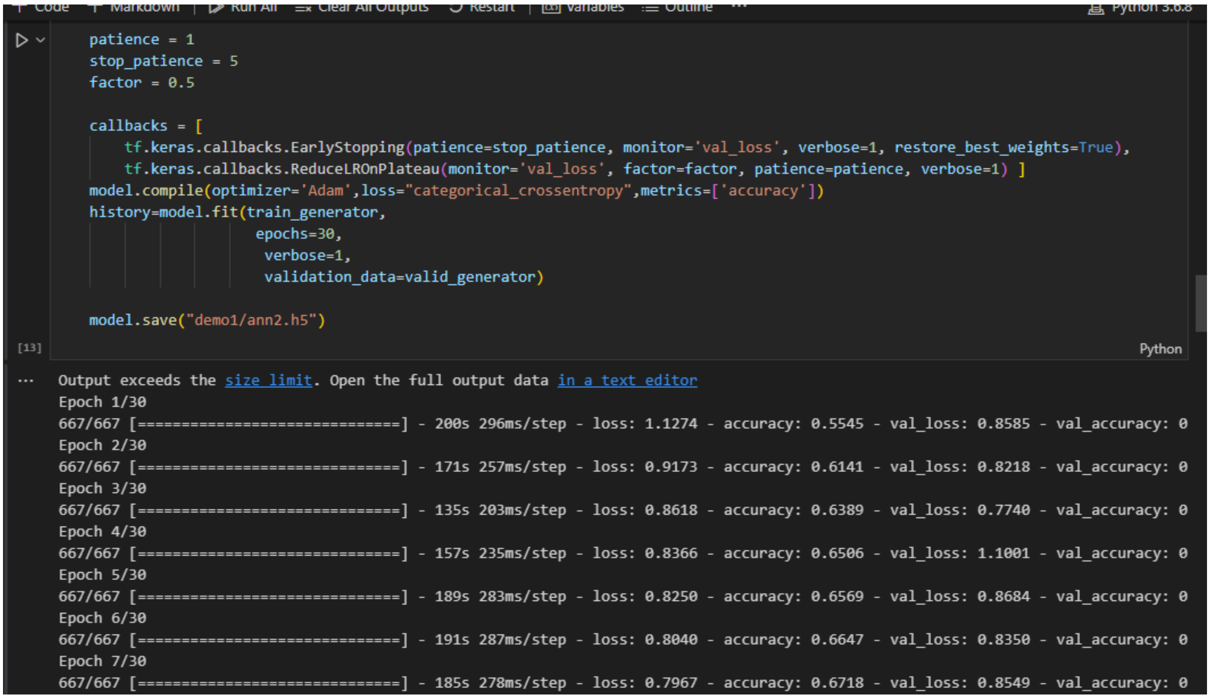
model building

```
from keras.layers import Dropout
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[128, 128, 3]),
    keras.layers.Dense(256,activation='relu'),
    keras.layers.BatchNormalization(),
    Dropout(0.2),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.BatchNormalization(),
    Dropout(0.2),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.BatchNormalization(),
    Dropout(0.2),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.BatchNormalization(),
    Dropout(0.2),
    keras.layers.Dense(4, activation='softmax')

])

model.summary()
```

```
... Output exceeds the size limit. Open the full output data in a text editor
    Model: "sequential_4"
    _________________________________________________________________
    Layer (type)                Output Shape              Param #
    =================================================================
    flatten_4 (Flatten)         (None, 49152)             0
    _________________________________________________________________
    dense_19 (Dense)            (None, 256)               12583168
    _________________________________________________________________
    batch_normalization_7 (Batch (None, 256)              1024
    _________________________________________________________________
    dropout_9 (Dropout)         (None, 256)               0
    _________________________________________________________________
    dense_20 (Dense)            (None, 128)               32896
    _________________________________________________________________
    batch_normalization_8 (Batch (None, 128)              512
    _________________________________________________________________
    dropout_10 (Dropout)        (None, 128)               0
    _________________________________________________________________
    dense_21 (Dense)            (None, 64)                8256
    _________________________________________________________________
    batch_normalization_9 (Batch (None, 64)               256
    _________________________________________________________________
    dropout_11 (Dropout)        (None, 64)                0
    _________________________________________________________________
    dense_22 (Dense)            (None, 32)                2080
    ...
    Total params: 12,628,452
    Trainable params: 12,627,492
    Non-trainable params: 960
```

Optimizing and fitting the model

```
model1.compile(optimizer='Adam',loss="categorical_crossentropy",metrics=['accuracy'])

history=model1.fit(train_generator,
                   epochs=30,
                    verbose=1,
                    validation_data=valid_generator)


model1.save("demo1/ann.h5")
```



```
patience = 1
stop_patience = 5
factor = 0.5

callbacks = [
    tf.keras.callbacks.EarlyStopping(patience=stop_patience, monitor='val_loss', verbose=1, restore_best_weights=True),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=factor, patience=patience, verbose=1) ]
model.compile(optimizer='Adam',loss="categorical_crossentropy",metrics=['accuracy'])
history=model.fit(train_generator,
                   epochs=30,
                    verbose=1,
                    validation_data=valid_generator)

model.save("demo1/ann2.h5")
```

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/30
667/667 [==============================] - 200s 296ms/step - loss: 1.1274 - accuracy: 0.5545 - val_loss: 0.8585 - val_accuracy: 0
Epoch 2/30
667/667 [==============================] - 171s 257ms/step - loss: 0.9173 - accuracy: 0.6141 - val_loss: 0.8218 - val_accuracy: 0
Epoch 3/30
667/667 [==============================] - 135s 203ms/step - loss: 0.8618 - accuracy: 0.6389 - val_loss: 0.7740 - val_accuracy: 0
Epoch 4/30
667/667 [==============================] - 157s 235ms/step - loss: 0.8366 - accuracy: 0.6506 - val_loss: 1.1001 - val_accuracy: 0
Epoch 5/30
667/667 [==============================] - 189s 283ms/step - loss: 0.8250 - accuracy: 0.6569 - val_loss: 0.8684 - val_accuracy: 0
Epoch 6/30
667/667 [==============================] - 191s 287ms/step - loss: 0.8040 - accuracy: 0.6647 - val_loss: 0.8350 - val_accuracy: 0
Epoch 7/30
667/667 [==============================] - 185s 278ms/step - loss: 0.7967 - accuracy: 0.6718 - val_loss: 0.8549 - val_accuracy: 0
```

Inceptionresnetv2

importing all the libraries

import numpy as np

import pandas as pd

%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns

import os, glob

import tensorflow as tf

from tqdm import tqdm

from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split

from keras.utils.np_utils import to_categorical

from keras.models import Model, Sequential, load_model

```
from keras.layers import Dense, concatenate, Dropout, Flatten, AvgPool2D,
Conv2D, MaxPool2D, BatchNormalization,
GlobalAveragePooling2D,Activation,Input

from tensorflow.keras.optimizers import Adam

from keras.preprocessing.image import ImageDataGenerator

import os

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from keras import regularizers

from matplotlib import pyplot as plt

import tensorflow as tf

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import confusion_matrix, classification_report

import seaborn as sns

import pandas as pd

import shutil

import numpy as np

from sklearn.metrics import precision_recall_fscore_support

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Model

from tensorflow.keras.losses import sparse_categorical_crossentropy

from tensorflow.keras.layers import Dense,GlobalAveragePooling2D

from tensorflow.keras.applications import MobileNet
```

```
from tensorflow.keras.applications.mobilenet import preprocess_input
import keras
from matplotlib import pyplot as plt
```

Datapreprocessing

```
img_height, img_width =128, 128
batch_size = 20
train_data_dir = "Dataset/"
test_data_dir = "Test/"

# Create test directory and move 10% of the data to this directory
if not os.path.exists(test_data_dir):
    os.makedirs(test_data_dir)
    for class_name in os.listdir(train_data_dir):
        class_dir = os.path.join(train_data_dir, class_name)
        test_class_dir = os.path.join(test_data_dir, class_name)
        os.makedirs(test_class_dir)
        files = os.listdir(class_dir)
        n_test = int(len(files) * 0.1)
        test_files = files[:n_test]
        for test_file in test_files:
            src = os.path.join(class_dir, test_file)
            dst = os.path.join(test_class_dir, test_file)
            shutil.move(src, dst)
```

```
# Create data generators

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   validation_split=0.3
                                   )
train_generator = train_datagen.flow_from_directory(train_data_dir,
                                                    target_size=(img_height,img_width),
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    subset='training')
valid_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
test_datagen = ImageDataGenerator(rescale=1./255,shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,)
test_generator = test_datagen.flow_from_directory(test_data_dir,
                                                    target_size=(img_height,img_width),
```

```
                                    batch_size=batch_size,
                                    class_mode='categorical',

                                    )
```

Model building

```
base_model =tf.keras.applications.InceptionResNetV2(input_shape=(img_height,img_width, 3), include_top=False,
                         weights='imagenet')
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(4, activation='sigmoid'))
model.summary()
```

```
base_model =tf.keras.applications.InceptionResNetV2(input_shape=(img_height,img_width, 3), include_top=False,
                                                    weights='imagenet')
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(4, activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 2, 2, 1536)        54336736
_________________________________________________________________
global_average_pooling2d (Gl (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 64)                98368
_________________________________________________________________
batch_normalization_203 (Bat (None, 64)                256
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 54,435,620
Trainable params: 54,374,948
```

Optimizing and fitting the model

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

history=model.fit(train_generator,epochs=4,validation_data=valid_generator)

model.save("demo1/InceptionResnetv2.h5")

```
    tf.keras.callbacks.EarlyStopping(patience=stop_patience, monitor='val_loss', verbose=1, restore_best_weights=True),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=factor, patience=patience, verbose=1) ]
```

```
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=['accuracy'])
history=model.fit(train_generator,epochs=4,validation_data=valid_generator)
model.save("demo1/InceptionResnetv2.h5")
```

```
Epoch 1/4
667/667 [==============================] - 8576s 13s/step - loss: 0.5603 - accuracy: 0.8054 - val_loss: 0.5542 - val_accuracy: 0.
Epoch 2/4
667/667 [==============================] - 2519s 4s/step - loss: 0.3659 - accuracy: 0.8692 - val_loss: 0.5125 - val_accuracy: 0.8
Epoch 3/4
667/667 [==============================] - 2526s 4s/step - loss: 0.3156 - accuracy: 0.8919 - val_loss: 0.4062 - val_accuracy: 0.8
Epoch 4/4
667/667 [==============================] - 2534s 4s/step - loss: 0.2906 - accuracy: 0.8984 - val_loss: 0.6835 - val_accuracy: 0.7
```

mobilenet

importing all the libraries

import numpy as np

import pandas as pd

%matplotlib inline

```
import matplotlib.pyplot as plt
import seaborn as sns
import os, glob
import tensorflow as tf
from tqdm import tqdm
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.models import Model, Sequential, load_model
from keras.layers import Dense, concatenate, Dropout, Flatten, AvgPool2D,
Conv2D, MaxPool2D, BatchNormalization,
GlobalAveragePooling2D,Activation,Input
from tensorflow.keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from keras import regularizers
from matplotlib import pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
```

```
import pandas as pd
import shutil
import numpy as np
from sklearn.metrics import precision_recall_fscore_support
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.layers import Dense,GlobalAveragePooling2D
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.applications.mobilenet import preprocess_input
import keras
from matplotlib import pyplot as plt
```

data preprocessing

```
img_height, img_width =128, 128
batch_size = 20
train_data_dir = "Dataset/"
test_data_dir = "Test/"

# Create test directory and move 10% of the data to this directory
if not os.path.exists(test_data_dir):
    os.makedirs(test_data_dir)
    for class_name in os.listdir(train_data_dir):
        class_dir = os.path.join(train_data_dir, class_name)
        test_class_dir = os.path.join(test_data_dir, class_name)
```

```
        os.makedirs(test_class_dir)
        files = os.listdir(class_dir)
        n_test = int(len(files) * 0.1)
        test_files = files[:n_test]
        for test_file in test_files:
            src = os.path.join(class_dir, test_file)
            dst = os.path.join(test_class_dir, test_file)
            shutil.move(src, dst)

# Create data generators

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   validation_split=0.3
                                   )
train_generator = train_datagen.flow_from_directory(train_data_dir,
                                                    target_size=(img_height,img_width),
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    subset='training')
valid_generator = train_datagen.flow_from_directory(
    train_data_dir,
```

```
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
test_datagen = ImageDataGenerator(rescale=1./255,shear_range = 0.2,
                                  zoom_range = 0.2,
                                  horizontal_flip = True,)
test_generator = test_datagen.flow_from_directory(test_data_dir,
                                                  target_size=(img_height,img_width),
                                                  batch_size=batch_size,
                                                  class_mode='categorical',

                                                  )
```

<u>model building</u>

```
base_model=MobileNet(input_shape=(img_height,img_width,
3),weights='imagenet',include_top=False)
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(1024,activation='relu'))
model.add(Dense(1024,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(4,activation='softmax'))
model.summary()
```

```python
base_model=MobileNet(input_shape=(img_height,img_width, 3),weights='imagenet',include_top=False)
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(1024,activation='relu'))
model.add(Dense(1024,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(4,activation='softmax'))
model.summary()
```

[9] ✓ 1.4s Python

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
mobilenet_1.00_128 (Function (None, 4, 4, 1024)       3228864
_________________________________________________________________
global_average_pooling2d_2 ( (None, 1024)             0
_________________________________________________________________
dense_8 (Dense)             (None, 1024)              1049600
_________________________________________________________________
dense_9 (Dense)             (None, 1024)              1049600
_________________________________________________________________
dense_10 (Dense)            (None, 512)               524800
_________________________________________________________________
dense_11 (Dense)            (None, 4)                 2052
=================================================================
Total params: 5,854,916
Trainable params: 5,833,028
Non-trainable params: 21,888
```

<u>compiling and optimizing the model</u>

patience = 1

stop_patience = 5

factor = 0.5

callbacks = [

tf.keras.callbacks.EarlyStopping(patience=stop_patience, monitor='val_loss', verbose=1, restore_best_weights=True),

tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=factor, patience=patience, verbose=1)]

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

history=model.fit(train_generator,epochs=10,validation_data=valid_generator)

model.save("demo1/mobilenet.h5")

```python
patience = 1
stop_patience = 5
factor = 0.5

callbacks = [
    tf.keras.callbacks.EarlyStopping(patience=stop_patience, monitor='val_loss', verbose=1, restore_best_weights=True),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=factor, patience=patience, verbose=1) ]
model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=['accuracy'])
history=model.fit(train_generator,epochs=10,validation_data=valid_generator)
model.save("demo1/mobilenet.h5")
```

```
Epoch 1/10
667/667 [==============================] - 665s 989ms/step - loss: 0.5401 - accuracy: 0.8155 - val_loss: 0.5678 - val_accuracy: 0
Epoch 2/10
667/667 [==============================] - 550s 825ms/step - loss: 0.3901 - accuracy: 0.8684 - val_loss: 0.4258 - val_accuracy: 0
Epoch 3/10
667/667 [==============================] - 550s 825ms/step - loss: 0.3497 - accuracy: 0.8811 - val_loss: 0.3752 - val_accuracy: 0
Epoch 4/10
667/667 [==============================] - 550s 824ms/step - loss: 0.3040 - accuracy: 0.8931 - val_loss: 0.5007 - val_accuracy: 0
Epoch 5/10
667/667 [==============================] - 549s 823ms/step - loss: 0.2995 - accuracy: 0.8987 - val_loss: 0.2990 - val_accuracy: 0
Epoch 6/10
667/667 [==============================] - 546s 818ms/step - loss: 0.2772 - accuracy: 0.9023 - val_loss: 0.2824 - val_accuracy: 0
Epoch 7/10
667/667 [==============================] - 546s 818ms/step - loss: 0.2601 - accuracy: 0.9093 - val_loss: 0.2889 - val_accuracy: 0
Epoch 8/10
667/667 [==============================] - 547s 820ms/step - loss: 0.2551 - accuracy: 0.9118 - val_loss: 0.3069 - val_accuracy: 0
Epoch 9/10
```

website creation code

sample code

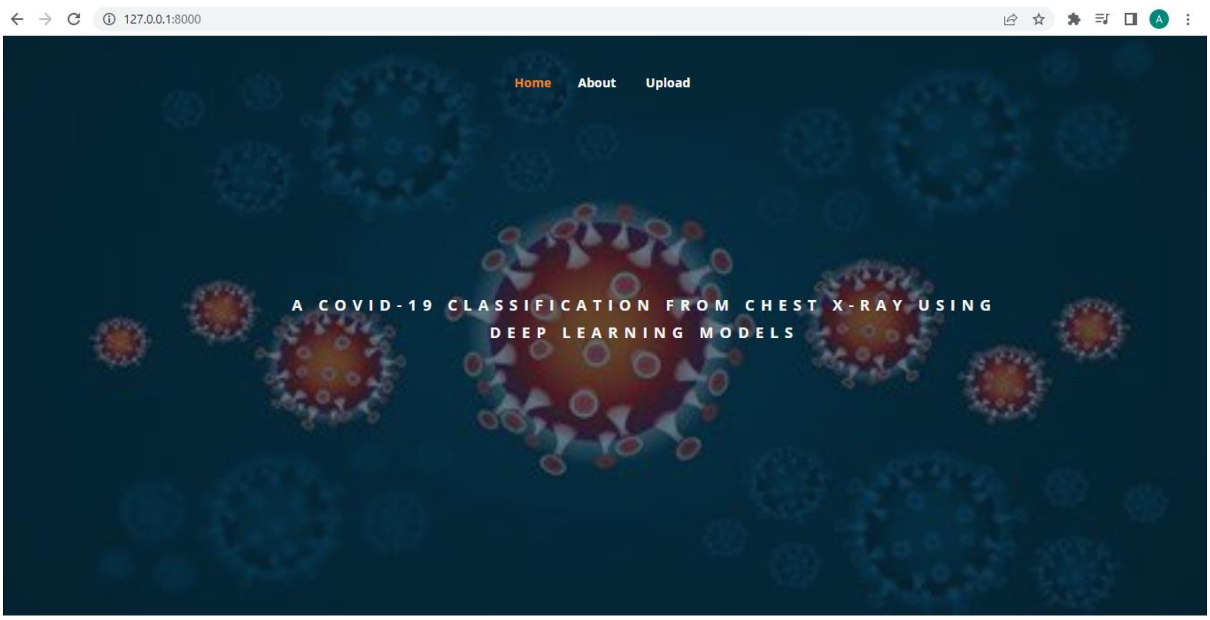
views.py

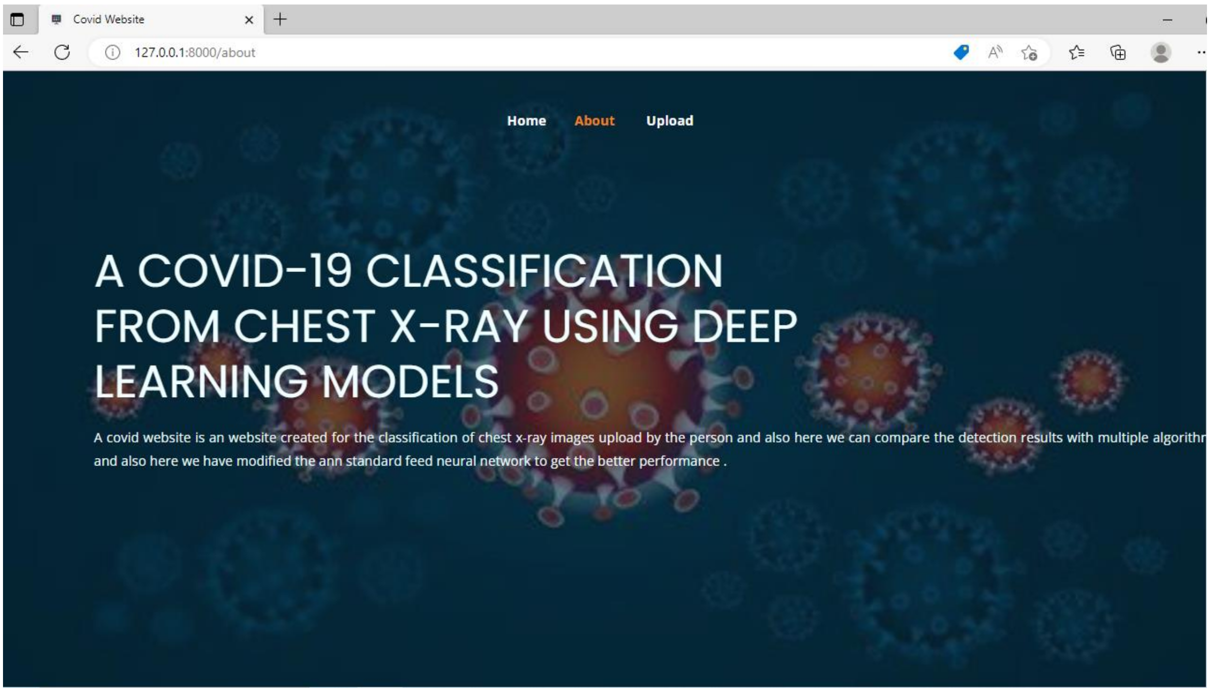
```python
from django.shortcuts import render
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np
from . models import Brain
import os


# Create your views here.
Classes=['COVID', 'Lung_Opacity', 'Normal', 'Viral Pneumonia']


def index(request):
```

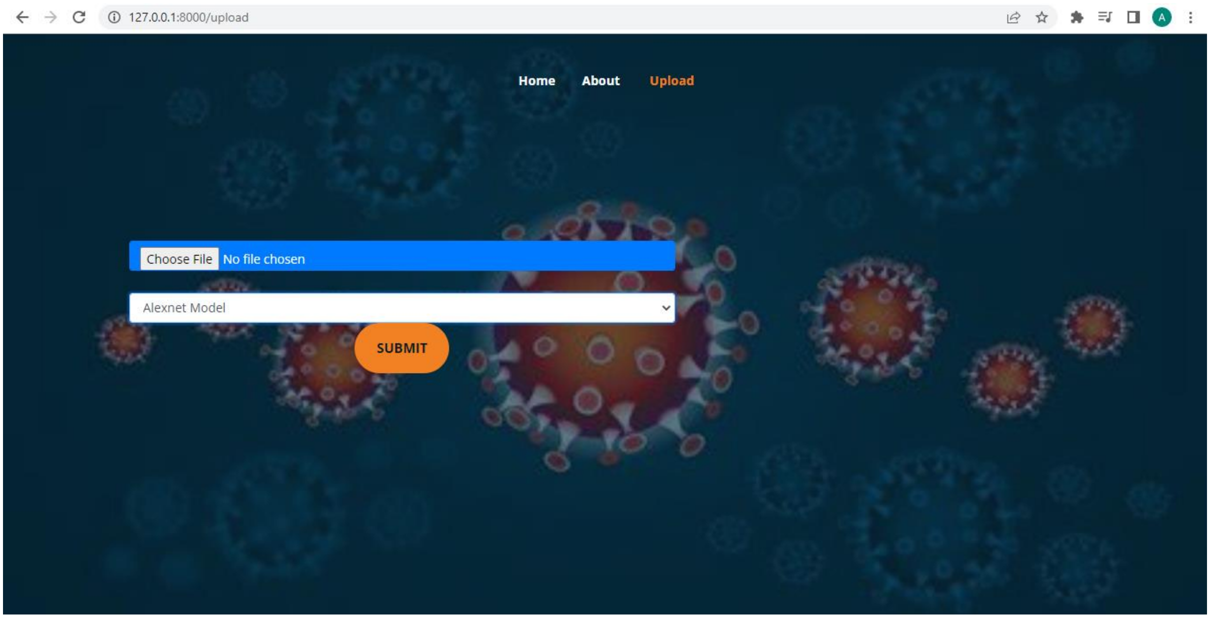
```
    return render(request, "index.html")



def about(request):
    return render(request,"about.html")



def upload(request):
    if request.method=='POST':

        m1 = int(request.POST['alg'])

        File=request.FILES['brain']
        s=Brain(image=File)
        s.save()
        path1='app/static/saved/' + s.Imagename()

        print(path1)

        if m1==1:
            model=load_model('app/demo1/alexnet.h5',compile=False)
            x1=image.load_img(path1,target_size=(224,224))
            x1=image.img_to_array(x1)
            x1/=255
```

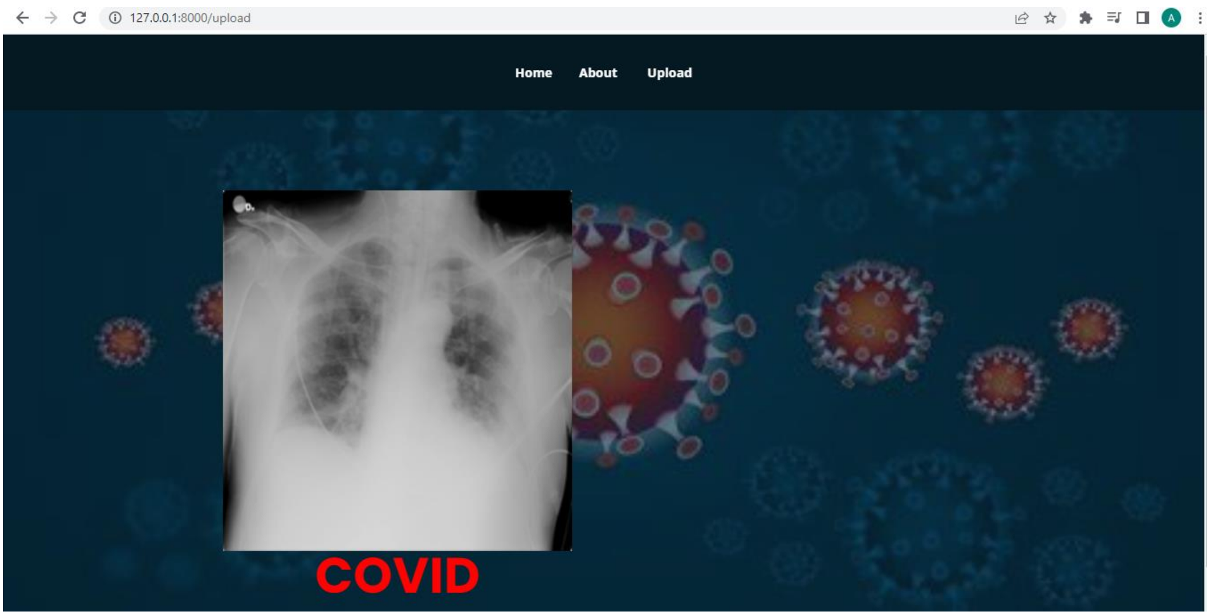
```
elif m1==2:
    model=load_model('app/demo1/resnet.h5',compile=False)
    x1=image.load_img(path1,target_size=(128,128))
    x1=image.img_to_array(x1)
    x1/=255

elif m1==3:
    model=load_model('app/demo1/ann.h5',compile=False)
    x1=image.load_img(path1,target_size=(128,128))
    x1=image.img_to_array(x1)
    x1/=255
elif m1==4:
    model=load_model('app/demo1/mobilenet.h5',compile=False)
    x1=image.load_img(path1,target_size=(224,224))
    x1=image.img_to_array(x1)
    x1/=255

elif m1==5:
    model=load_model('app/demo1/InceptionResNetv2.h5',compile=False)
    x1=image.load_img(path1,target_size=(128,128))
    x1=image.img_to_array(x1)
    x1/=255
```

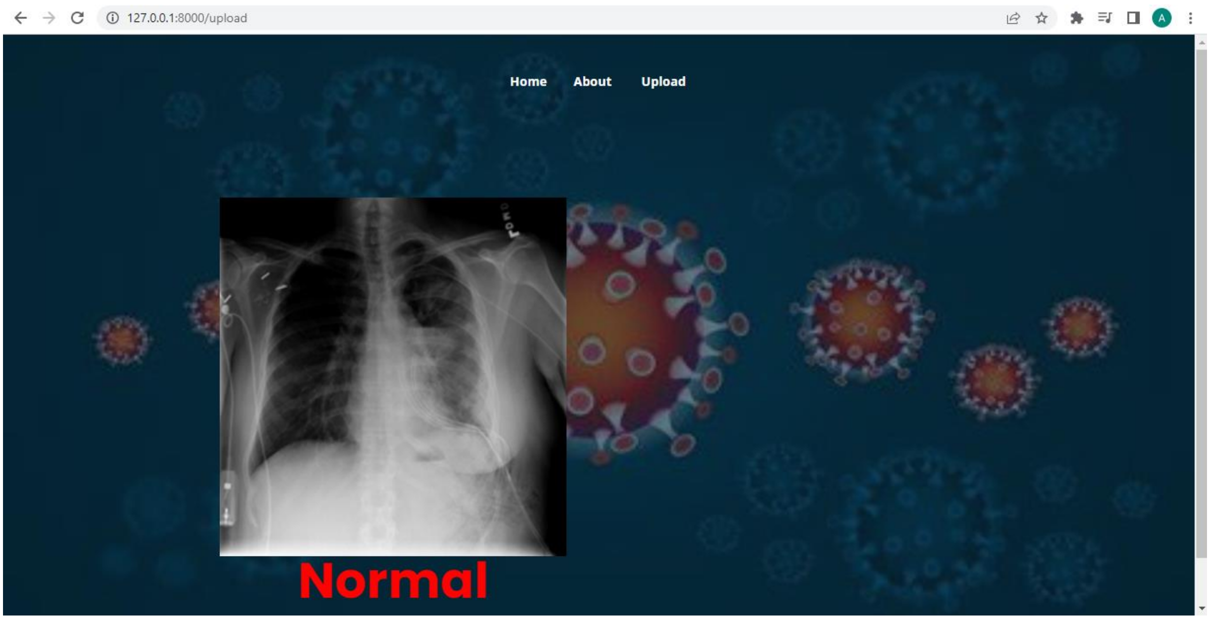
```
        x1=np.expand_dims(x1,axis=0)
        result= model.predict(x1)
        pred = Classes[np.argmax(result)]
        print(pred)
        return render(request,"result.html",{"message":pred,"path":'/static/saved/' + s.Imagename()})

    return render (request,"upload.html")
```

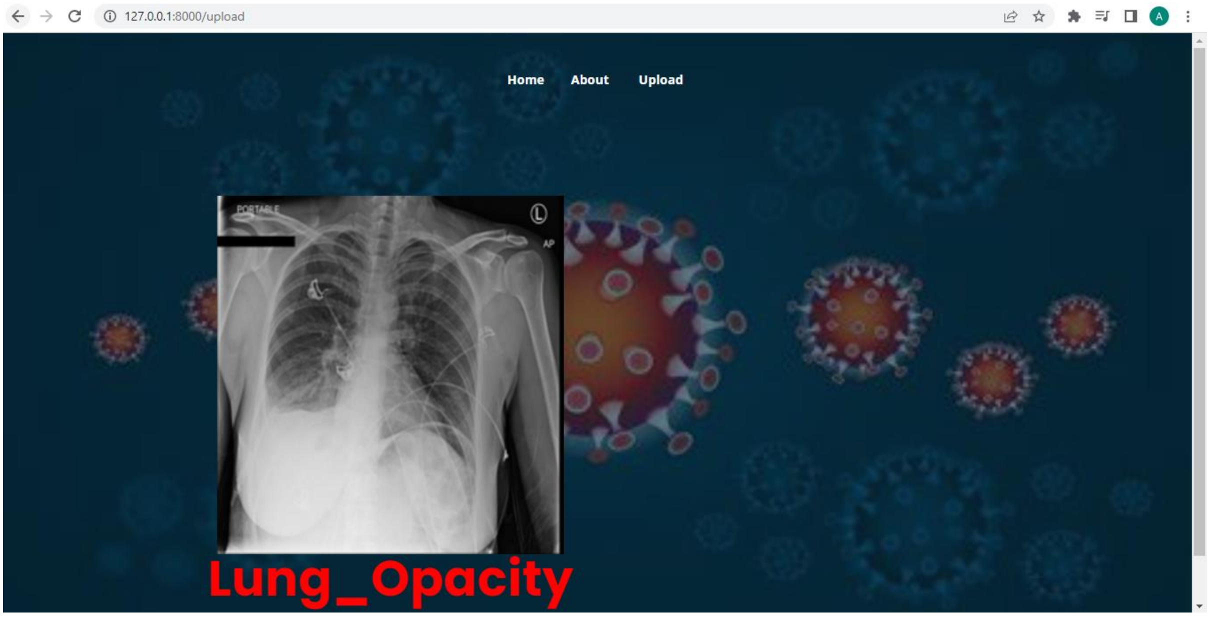


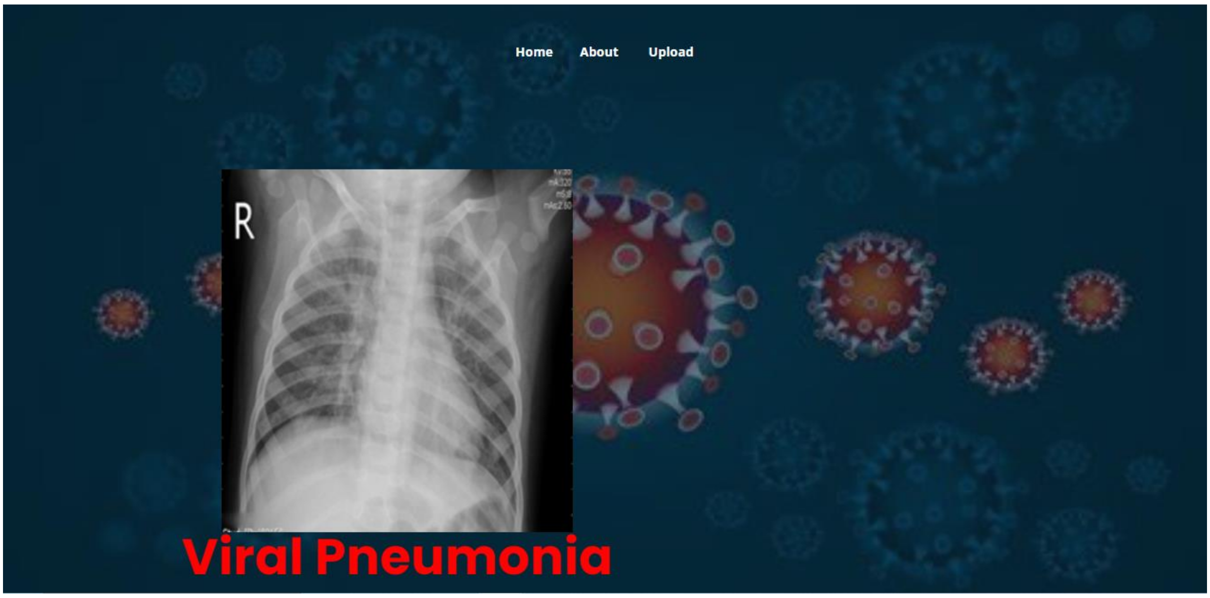

Covid Website
127.0.0.1:8000/about
Home
About
Upload
A COVID-19 CLASSIFICATION FROM CHEST X-RAY USING DEEP LEARNING MODELS
A covid website is an website created for the classification of chest x-ray images upload by the person and also here we can compare the detection results with multiple algorithr
and also here we have modified the ann standard feed neural network to get the better performance .


127.0.0.1:8000/upload
Home
About
Upload
Choose File
No file chosen
Alexnet Model
SUBMIT


127.0.0.1:8000/upload
Home
About
Upload
COVID


127.0.0.1:8000/upload
Home
About
Upload
Normal


127.0.0.1:8000/upload
Home
About
Upload
Lung_Opacity


Home
About
Upload
R
Viral Pneumonia

5.4 TEST PLAN & DATA VERIFICATION

1. **Testing for the system**

SNO	Test case	Preconditions	Input data	Expected results
1.	Test the user can able to upload the image correctly	There should be all required setup in system	Provide the image in correct format	Image to be tested should be defined
2.	Detect the disease from the image uploaded	The image should be clear with pixels	Pre-processing should be done	Prediction of disease results should be displayed
3	Image identification in system	the image should be processed	Provide valid image format	Identifying the diseases from the uploaded image

Chapter 6

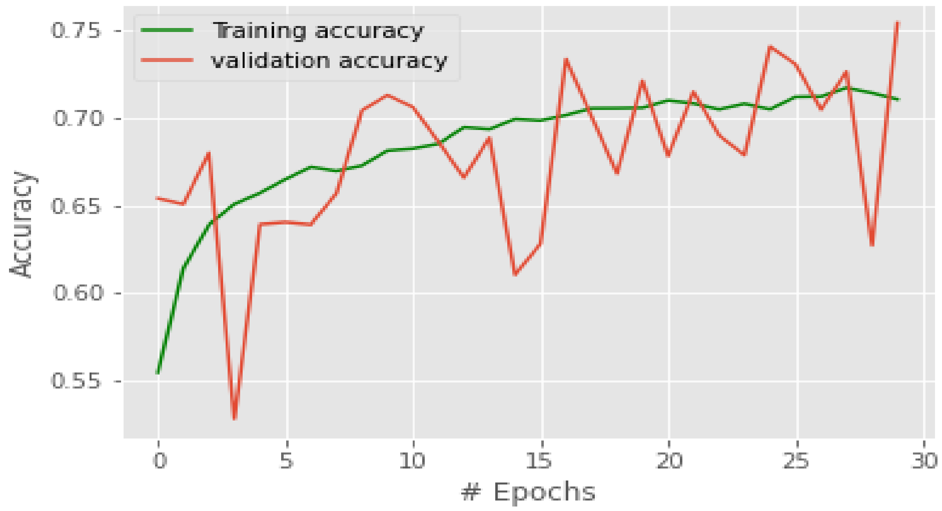
6.1 <u>Evalution metrics</u>

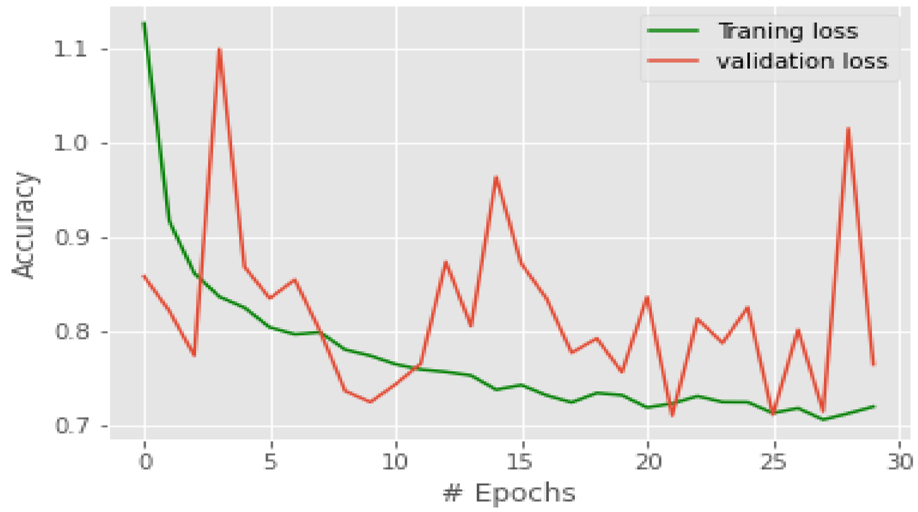
code for training accuracy

<u>ann</u>

```
plt.style.use("ggplot")
plt.figure()
plt.plot(history.history['accuracy'],'r',label='Training accuracy',color='green')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
```

```
plt.legend()
plt.savefig("demo1/ann_acc.png")
plt.show()
plt.style.use("ggplot")
plt.figure()
plt.plot(history.history['loss'],'r',label='Training loss',color='green')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("demo1/ann_loss.png")
plt.show()
acc=history.history['accuracy'][-1]
print(acc)
```





Code for testing accuracy

modelAccuracy = model.evaluate(test_generator, verbose=0)

print('Test Accuracy is {}%'.format(modelAccuracy[1] * 100))

```
modelAccuracy = model.evaluate(test_generator, verbose=0)
print('Test Accuracy is {}%'.format(modelAccuracy[1] * 100))
[19]                                                    Python
... Test Accuracy is 77.62535214424133%
```

Code for confusion matrix

y_pred = model.predict(test_generator) # predict on test_generator

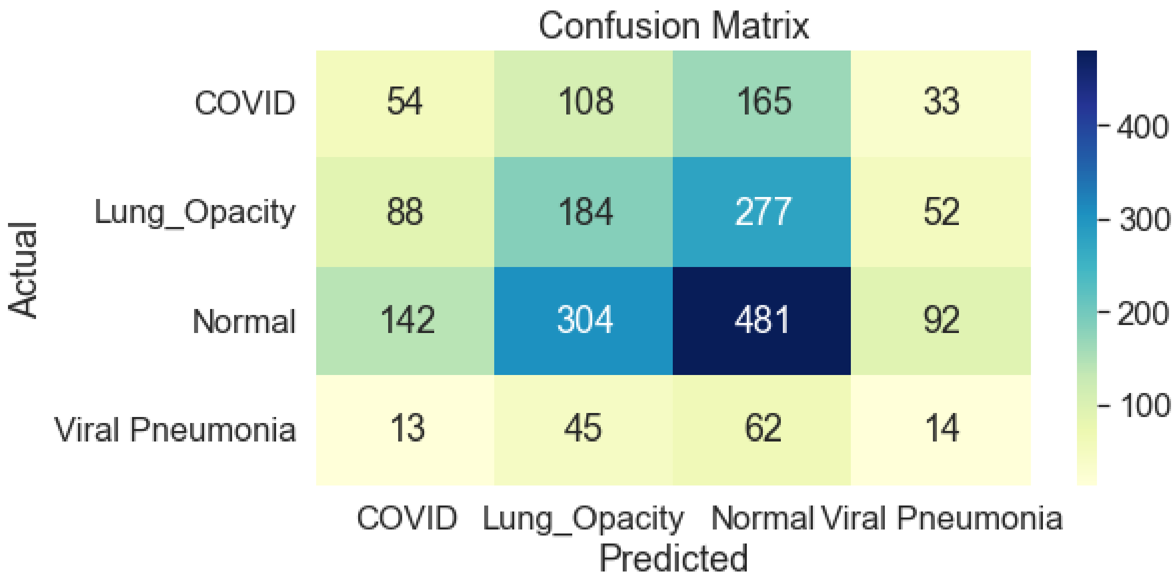
y_pred_classes = np.argmax(y_pred, axis=1) # obtain predicted class labels

conf_mat = confusion_matrix(test_generator.classes, y_pred_classes)

class_names = list(test_generator.class_indices.keys())

conf_mat_df = pd.DataFrame(conf_mat, index=class_names, columns=class_names)

```
plt.figure(figsize=(8,4))

sns.set(font_scale=1.5, color_codes=True, palette='deep')

sns.heatmap(conf_mat_df, annot=True, fmt='d', cmap="YlGnBu")

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()
```

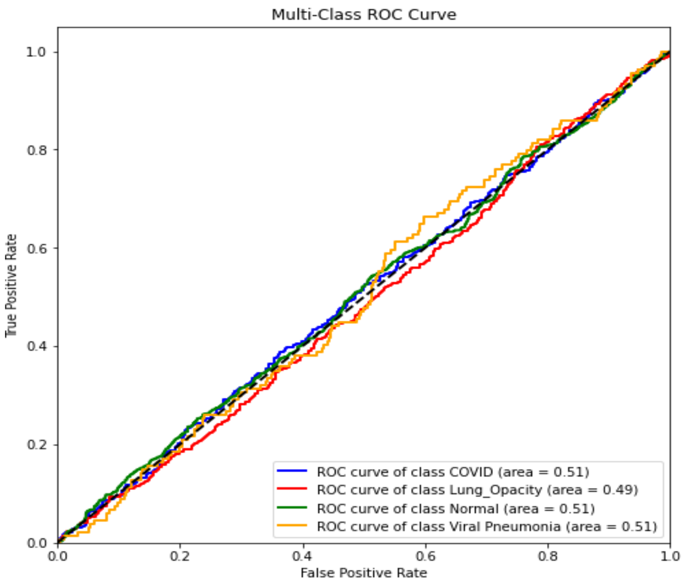


Code of roc curve

```
from sklearn.metrics import roc_auc_score, roc_curve

from sklearn.preprocessing import LabelBinarizer

import matplotlib.pyplot as plt

import keras


# Load the saved model

model = keras.models.load_model('demo1/ann2.h5')
```

```
class_names = list(test_generator.class_indices.keys())
# Make predictions on the test data
y_pred_proba = model.predict(test_generator)

# Calculate the AUC for each class
lb = LabelBinarizer()
lb.fit(test_generator.classes)
y_true = lb.transform(test_generator.classes)
aucs = []
for i in range(test_generator.num_classes):
    auc = roc_auc_score(y_true[:, i], y_pred_proba[:, i])
    aucs.append(auc)

# Plot the ROC curve
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(test_generator.num_classes):
    fpr[i], tpr[i], _ = roc_curve(y_true[:, i], y_pred_proba[:, i])
    roc_auc[i] = aucs[i]

plt.figure(figsize=(8, 8))
colors = ['blue', 'red', 'green', 'orange']
```

```
for i, color in zip(range(test_generator.num_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(class_names[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Multi-Class ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

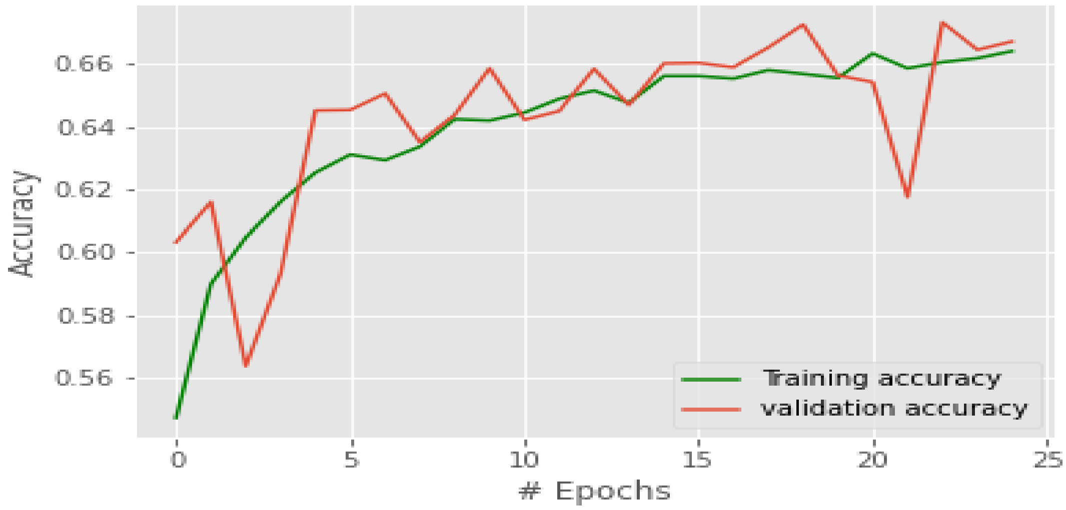


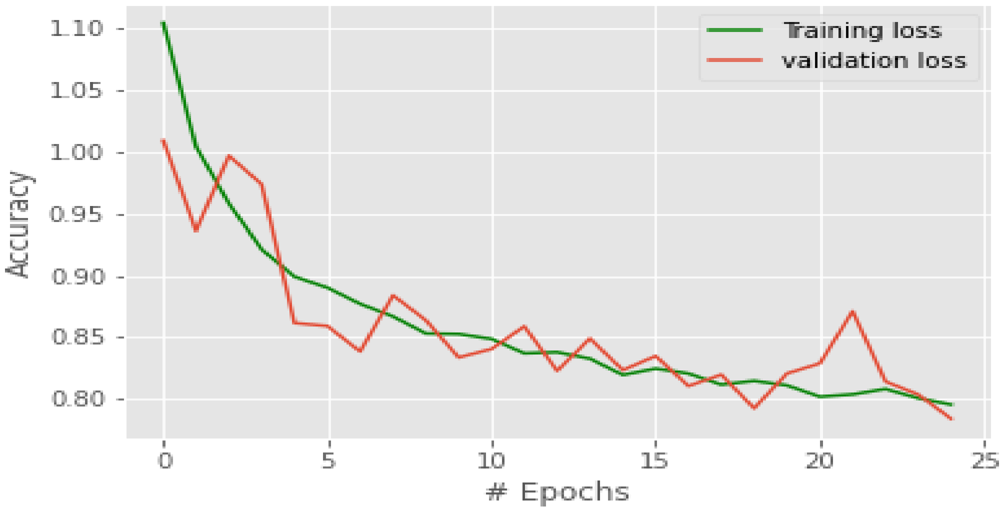
Resnet 50

Code for training accuracy

```
plt.style.use("ggplot")
```

```
plt.figure()
plt.plot(history.history['accuracy'],'r',label='Training accuracy',color='green')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("demo1/resnet_acc.png")
plt.show()
plt.style.use("ggplot")
plt.figure()
plt.plot(history.history['loss'],'r',label='Training loss',color='green')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("demo1/resnet_loss.png")
plt.show()
acc=history.history['accuracy'][-1]
print(acc)
```





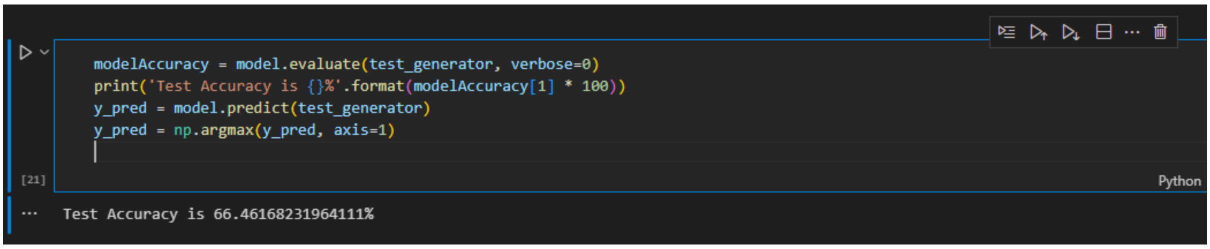
Code for test accuracy

modelAccuracy = model.evaluate(test_generator, verbose=0)

print('Test Accuracy is {}%'.format(modelAccuracy[1] * 100))

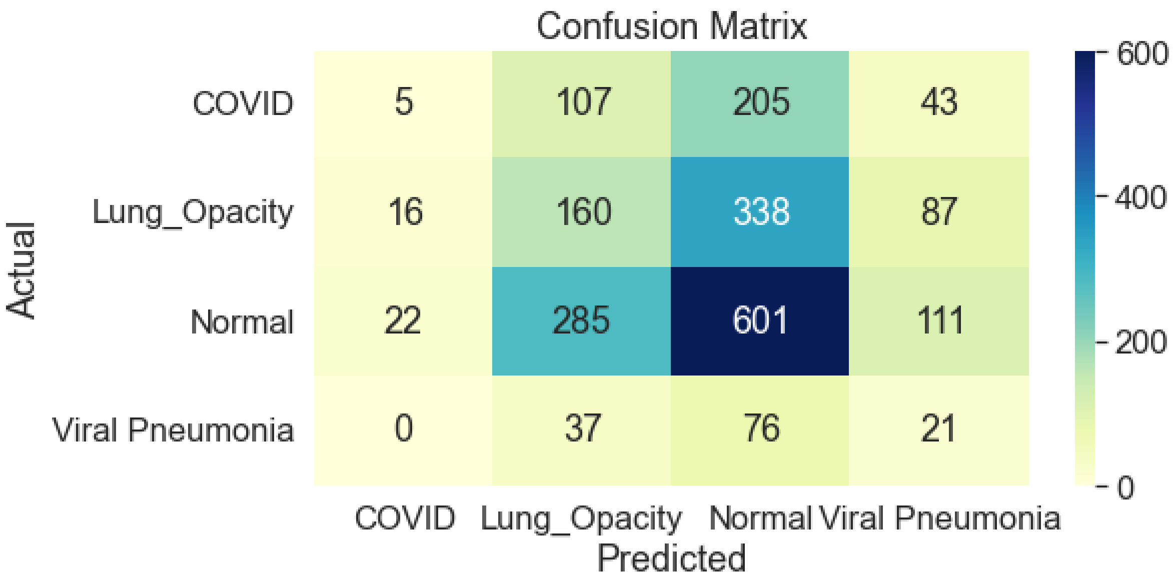
y_pred = model.predict(test_generator)

y_pred = np.argmax(y_pred, axis=1)



Code for confusion matrix

```
y_pred = model.predict(test_generator) # predict on test_generator

y_pred_classes = np.argmax(y_pred, axis=1) # obtain predicted class labels

conf_mat = confusion_matrix(test_generator.classes, y_pred_classes)

class_names = list(test_generator.class_indices.keys())

conf_mat_df = pd.DataFrame(conf_mat, index=class_names,
columns=class_names)

plt.figure(figsize=(8,4))

sns.set(font_scale=1.5, color_codes=True, palette='deep')

sns.heatmap(conf_mat_df, annot=True, fmt='d', cmap="YlGnBu")

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()
```

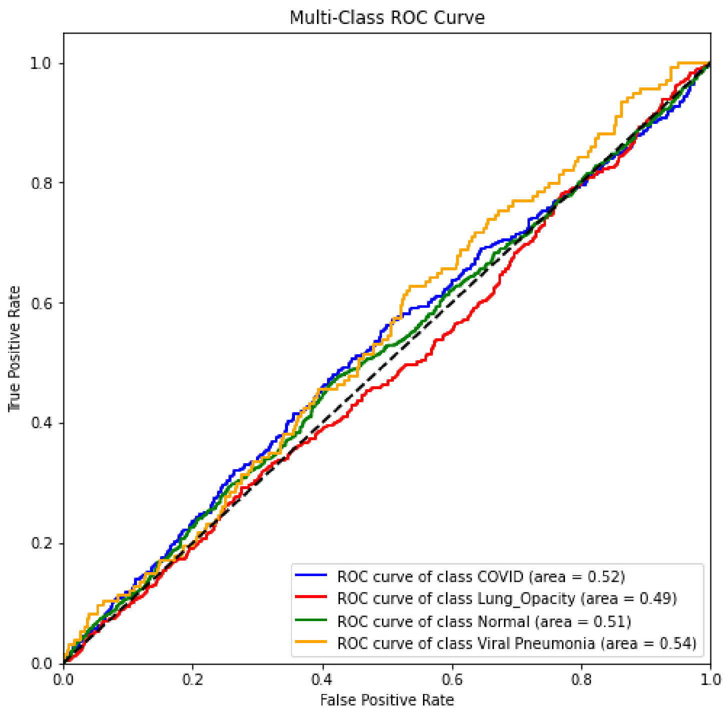


Code for the roc curve

```
from sklearn.metrics import roc_auc_score, roc_curve
```

```
from sklearn.preprocessing import LabelBinarizer
import matplotlib.pyplot as plt
import keras

# Load the saved model
model = keras.models.load_model('demo1/resnet.h5')

class_names = list(test_generator.class_indices.keys())
# Make predictions on the test data
y_pred_proba = model.predict(test_generator)

# Calculate the AUC for each class
lb = LabelBinarizer()
lb.fit(test_generator.classes)
y_true = lb.transform(test_generator.classes)
aucs = []
for i in range(test_generator.num_classes):
    auc = roc_auc_score(y_true[:, i], y_pred_proba[:, i])
    aucs.append(auc)

# Plot the ROC curve
fpr = dict()
tpr = dict()
roc_auc = dict()
```

```
for i in range(test_generator.num_classes):
    fpr[i], tpr[i], _ = roc_curve(y_true[:, i], y_pred_proba[:, i])
    roc_auc[i] = aucs[i]

plt.figure(figsize=(8, 8))
colors = ['blue', 'red', 'green', 'orange']
for i, color in zip(range(test_generator.num_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(class_names[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Multi-Class ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

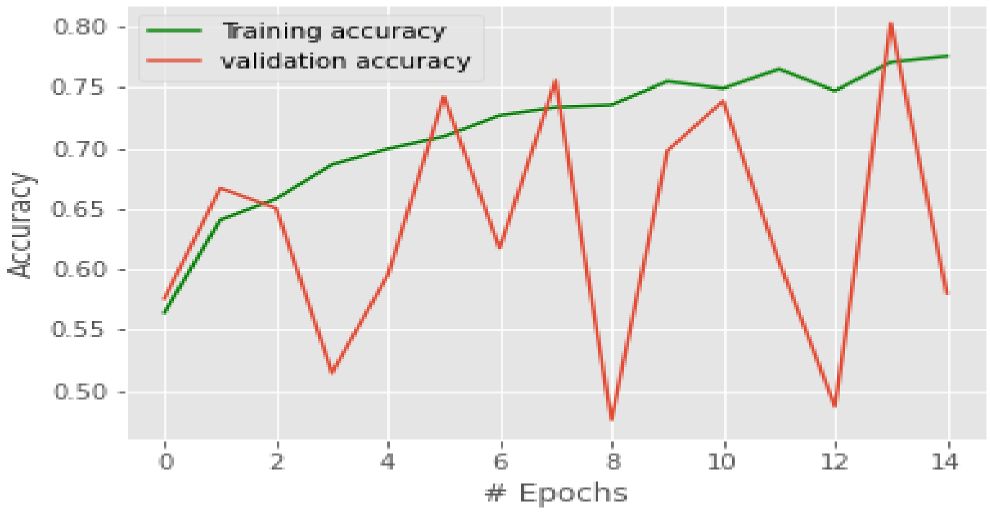


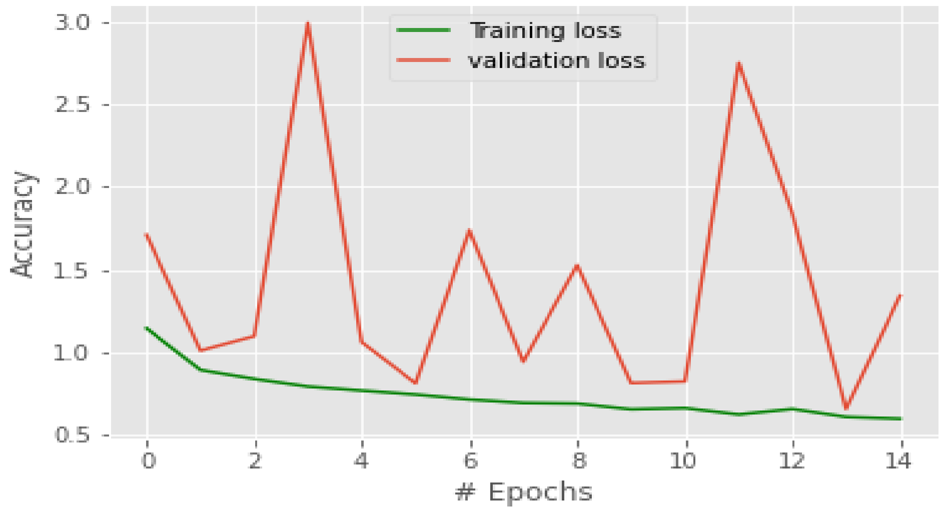
Alexnet

Code for the training accuracy

```
plt.style.use("ggplot")

plt.figure()

plt.plot(history.history['accuracy'],'r',label='Training accuracy',color='green')

plt.plot(history.history['val_accuracy'],label='validation accuracy')

plt.xlabel('# Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.savefig("demo/akex_acc.png")

plt.show()

plt.style.use("ggplot")

plt.figure()

plt.plot(history.history['loss'],'r',label='Training loss',color='green')
```

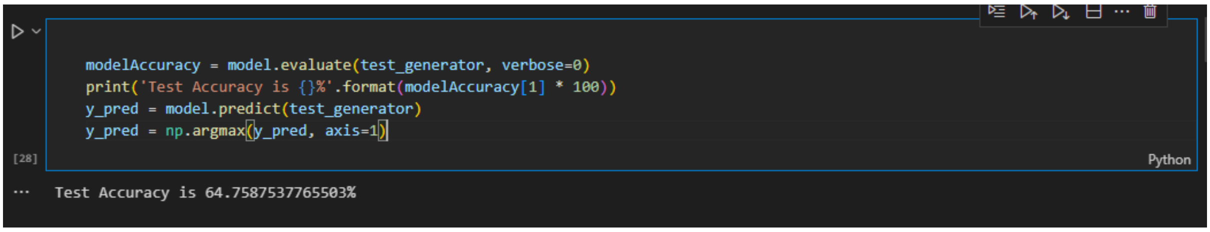
```
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("demo1/alex_loss.png")
plt.show()
acc=history.history['accuracy'][-1]
print(acc)
```





Code for test accuracy

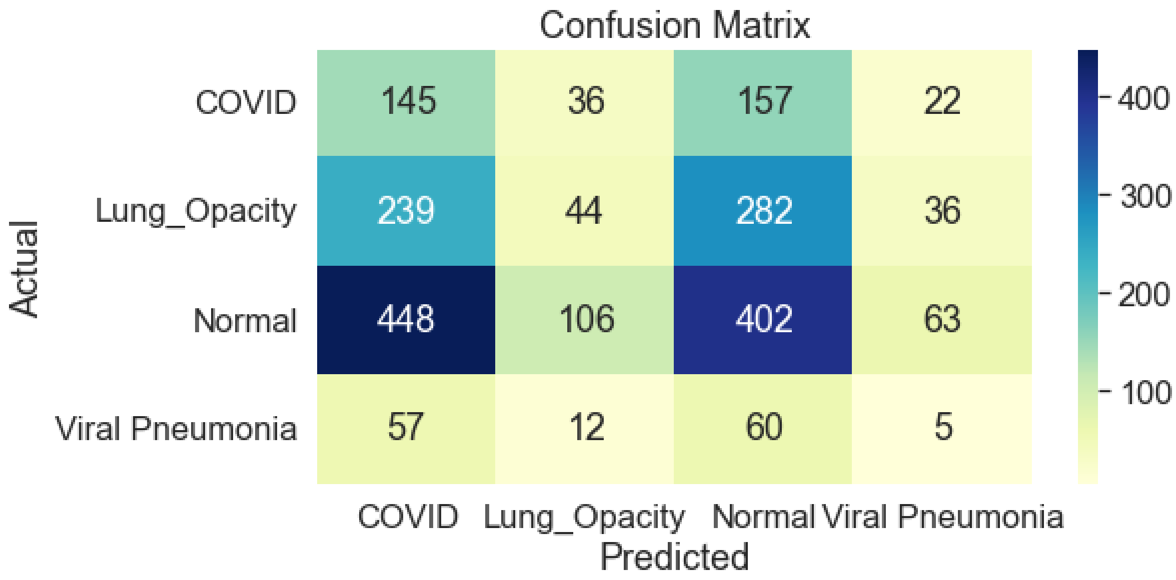
```
modelAccuracy = model.evaluate(test_generator, verbose=0)

print('Test Accuracy is {}%'.format(modelAccuracy[1] * 100))

y_pred = model.predict(test_generator)

y_pred = np.argmax(y_pred, axis=1)
```



```
modelAccuracy = model.evaluate(test_generator, verbose=0)
print('Test Accuracy is {}%'.format(modelAccuracy[1] * 100))
y_pred = model.predict(test_generator)
y_pred = np.argmax(y_pred, axis=1)
```


Code for confusion matrix

```
y_pred = model.predict(test_generator) # predict on test_generator

y_pred_classes = np.argmax(y_pred, axis=1) # obtain predicted class labels

conf_mat = confusion_matrix(test_generator.classes, y_pred_classes)

class_names = list(test_generator.class_indices.keys())

conf_mat_df = pd.DataFrame(conf_mat, index=class_names,
columns=class_names)

plt.figure(figsize=(8,4))

sns.set(font_scale=1.5, color_codes=True, palette='deep')

sns.heatmap(conf_mat_df, annot=True, fmt='d', cmap="YlGnBu")

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()
```

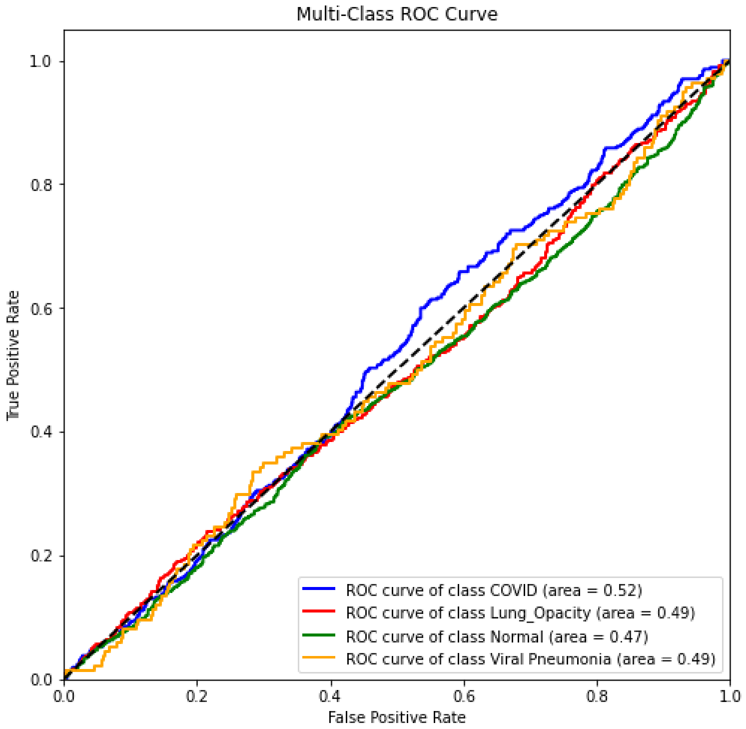


Code for roc curve

```
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.preprocessing import LabelBinarizer
import matplotlib.pyplot as plt
import keras

# Load the saved model
model = keras.models.load_model('demo1/alexnet.h5')

class_names = list(test_generator.class_indices.keys())
# Make predictions on the test data
y_pred_proba = model.predict(test_generator)

# Calculate the AUC for each class
lb = LabelBinarizer()
lb.fit(test_generator.classes)
```

```
y_true = lb.transform(test_generator.classes)
aucs = []
for i in range(test_generator.num_classes):
    auc = roc_auc_score(y_true[:, i], y_pred_proba[:, i])
    aucs.append(auc)

# Plot the ROC curve
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(test_generator.num_classes):
    fpr[i], tpr[i], _ = roc_curve(y_true[:, i], y_pred_proba[:, i])
    roc_auc[i] = aucs[i]

plt.figure(figsize=(8, 8))
colors = ['blue', 'red', 'green', 'orange']
for i, color in zip(range(test_generator.num_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(class_names[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Multi-Class ROC Curve')
plt.legend(loc="lower right")
plt.show()
```



Inceptionresnetv2

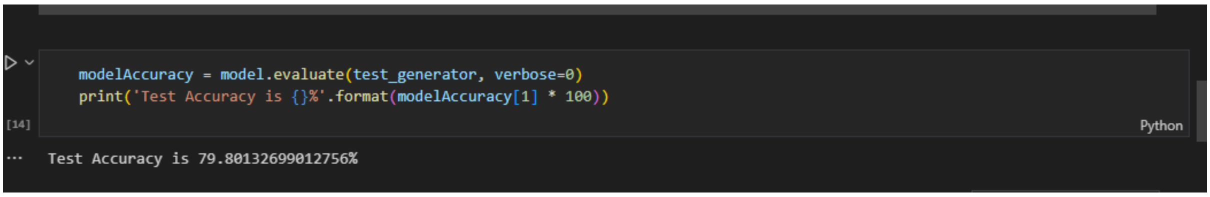
Code for training accuracy

```
plt.style.use("ggplot")
plt.figure()
plt.plot(history.history['accuracy'],'r',label='Training accuracy',color='green')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
```

```
plt.legend()

plt.savefig("demo1/InceptionResNetV2_acc.png")

plt.show()

plt.style.use("ggplot")

plt.figure()

plt.plot(history.history['loss'],'r',label='Training loss',color='green')

plt.plot(history.history['val_loss'],label='validation loss')

plt.xlabel('# Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.savefig("demo1/InceptionResNetV2_loss.png")

plt.show()
```

Code for test accuracy

```
modelAccuracy = model.evaluate(test_generator, verbose=0)

print('Test Accuracy is {}%'.format(modelAccuracy[1] * 100))

y_pred = model.predict(test_generator)

y_pred = np.argmax(y_pred, axis=1)
```



Code for confusion matrix

```
y_pred = model.predict(test_generator) # predict on test_generator

y_pred_classes = np.argmax(y_pred, axis=1) # obtain predicted class labels

conf_mat = confusion_matrix(test_generator.classes, y_pred_classes)
```

class_names = list(test_generator.class_indices.keys())

conf_mat_df = pd.DataFrame(conf_mat, index=class_names, columns=class_names)

plt.figure(figsize=(8,4))

sns.set(font_scale=1.5, color_codes=True, palette='deep')

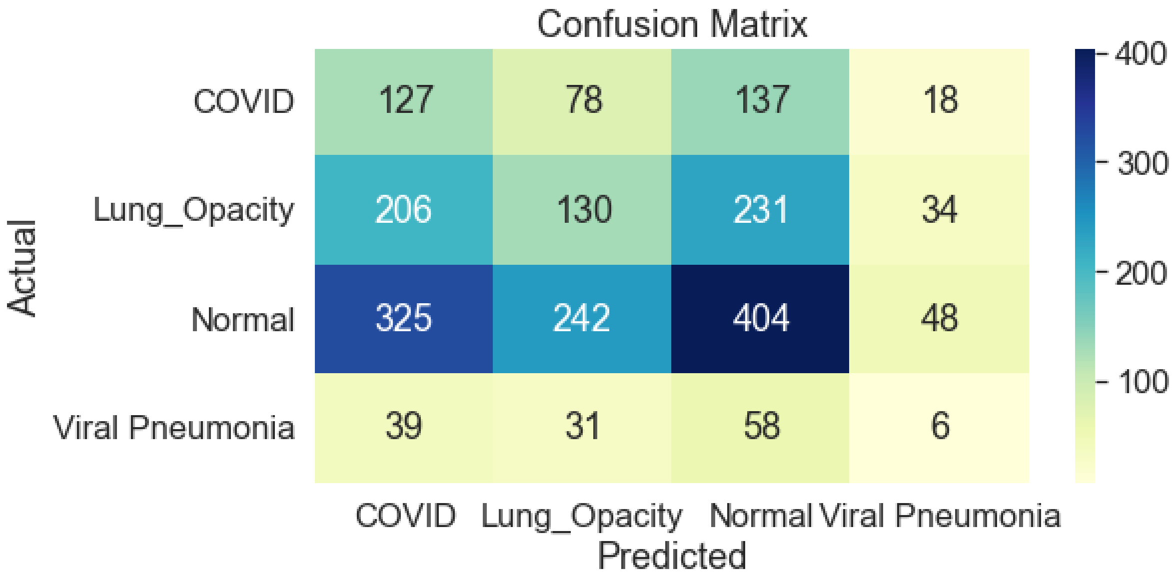
sns.heatmap(conf_mat_df, annot=True, fmt='d', cmap="YlGnBu")

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()



Code for roc curve

from sklearn.metrics import roc_auc_score, roc_curve

from sklearn.preprocessing import LabelBinarizer

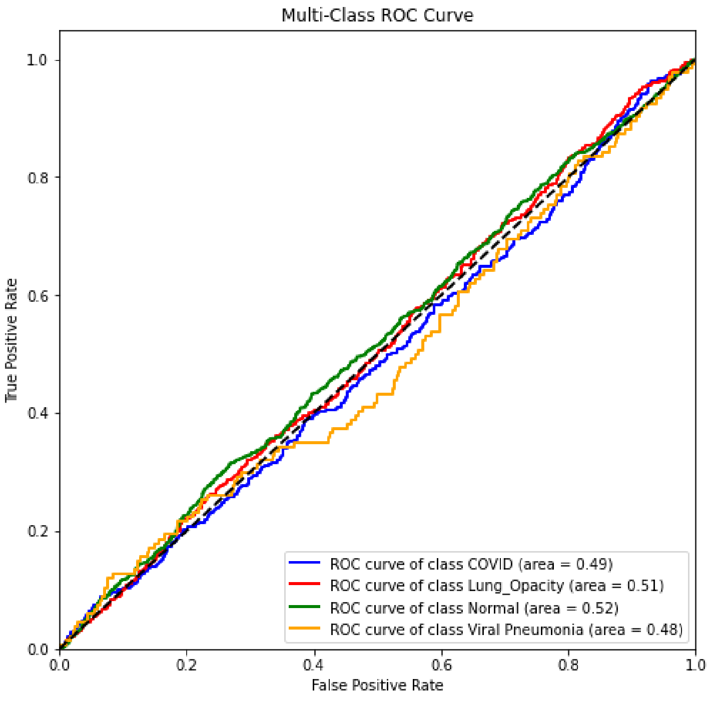
import matplotlib.pyplot as plt

import keras

Load the saved model

```
model = keras.models.load_model('demo1/InceptionResnetv2.h5')


class_names = list(test_generator.class_indices.keys())
# Make predictions on the test data
y_pred_proba = model.predict(test_generator)


# Calculate the AUC for each class
lb = LabelBinarizer()
lb.fit(test_generator.classes)
y_true = lb.transform(test_generator.classes)
aucs = []
for i in range(test_generator.num_classes):
    auc = roc_auc_score(y_true[:, i], y_pred_proba[:, i])
    aucs.append(auc)


# Plot the ROC curve
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(test_generator.num_classes):
    fpr[i], tpr[i], _ = roc_curve(y_true[:, i], y_pred_proba[:, i])
    roc_auc[i] = aucs[i]


plt.figure(figsize=(8, 8))
```

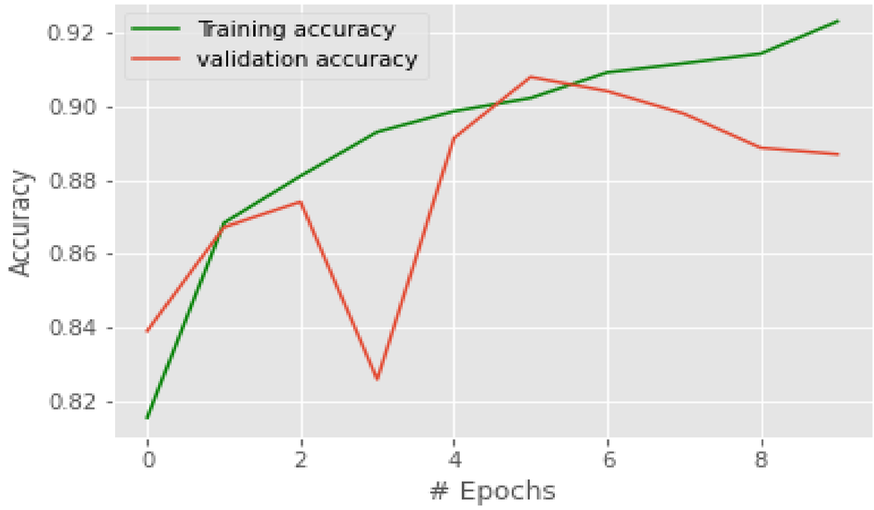
```
colors = ['blue', 'red', 'green', 'orange']
for i, color in zip(range(test_generator.num_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(class_names[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Multi-Class ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

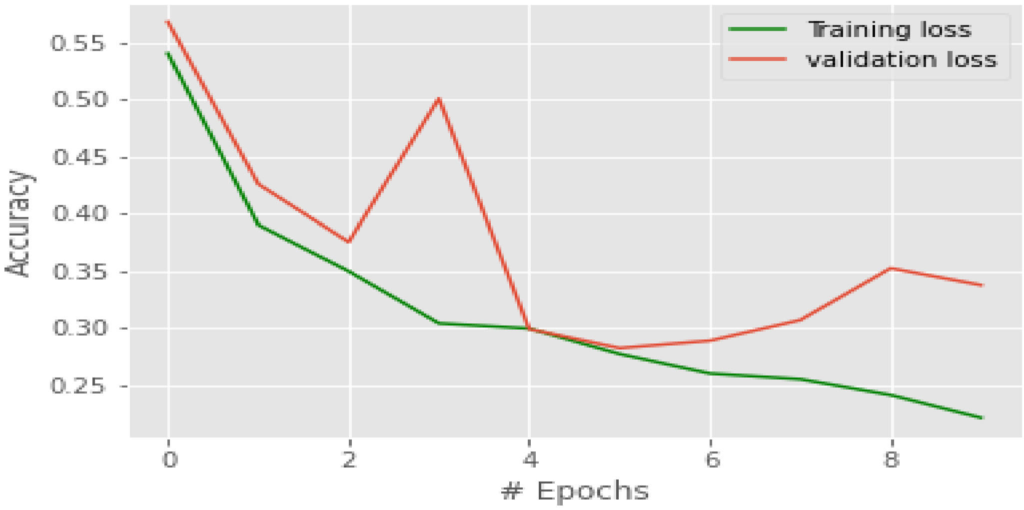


Mobilenet

Code for training accuracy

```
plt.style.use("ggplot")
plt.figure()
plt.plot(history.history['accuracy'],'r',label='Training accuracy',color='green')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("demo1/Mobilenet_acc.png")
plt.show()
plt.style.use("ggplot")
plt.figure()
plt.plot(history.history['loss'],'r',label='Training loss',color='green')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig("demo1/Mobilenet_loss.png")
plt.show()
```





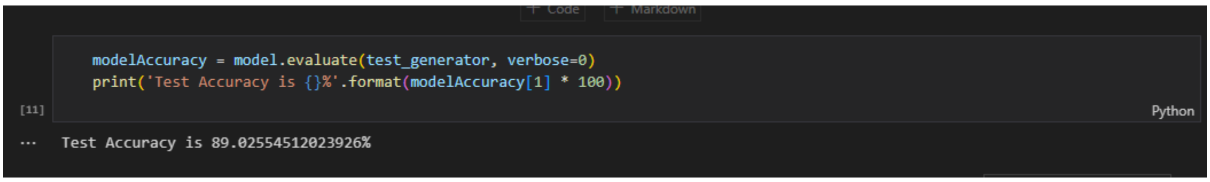
Code for test accuracy

modelAccuracy = model.evaluate(test_generator, verbose=0)

print('Test Accuracy is {}%'.format(modelAccuracy[1] * 100))

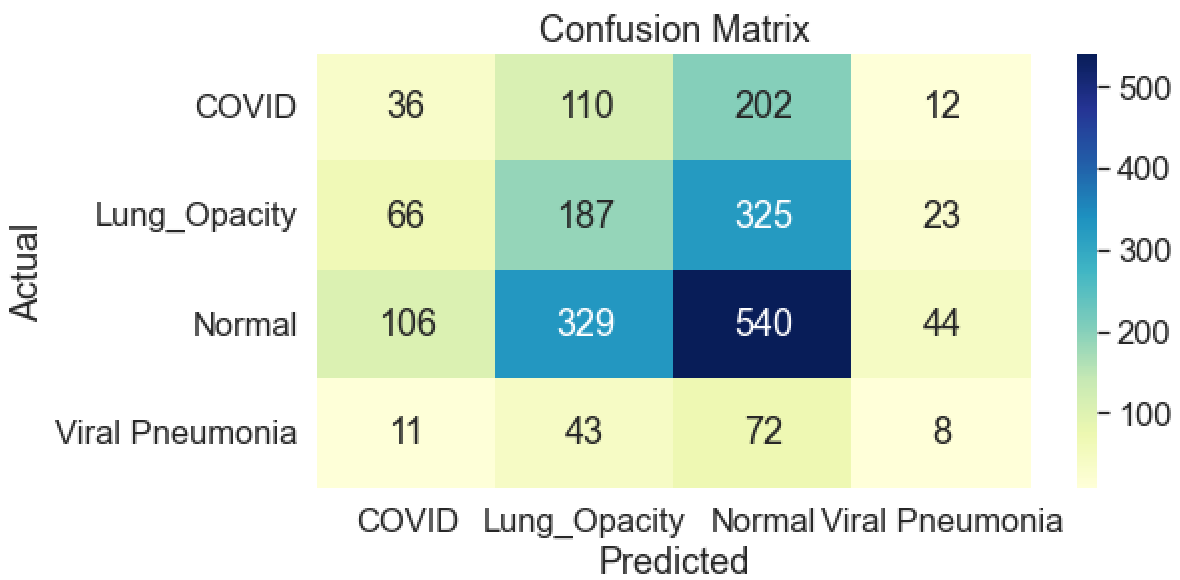
y_pred = model.predict(test_generator)

y_pred = np.argmax(y_pred, axis=1)



Code for confusion matrix

```
y_pred = model.predict(test_generator) # predict on test_generator

y_pred_classes = np.argmax(y_pred, axis=1) # obtain predicted class labels

conf_mat = confusion_matrix(test_generator.classes, y_pred_classes)

class_names = list(test_generator.class_indices.keys())

conf_mat_df = pd.DataFrame(conf_mat, index=class_names,
columns=class_names)

plt.figure(figsize=(8,4))

sns.set(font_scale=1.5, color_codes=True, palette='deep')

sns.heatmap(conf_mat_df, annot=True, fmt='d', cmap="YlGnBu")

plt.title('Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()
```

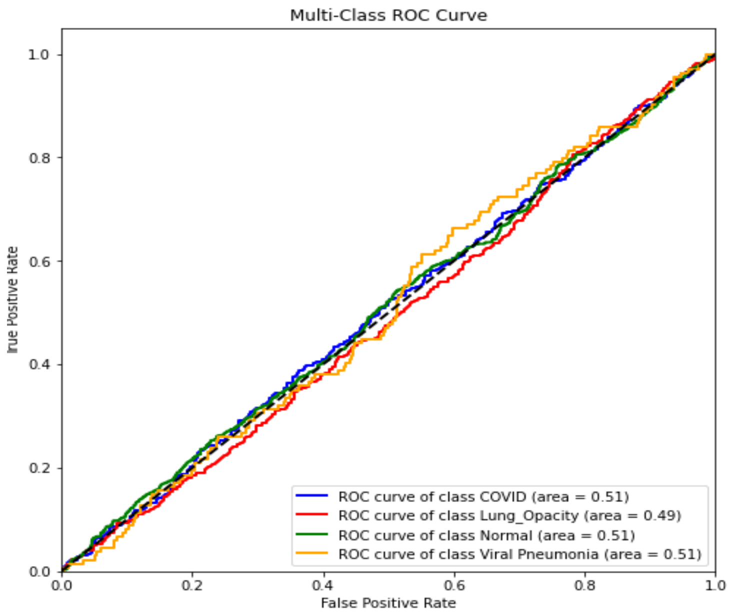


code for roc curve

```
from sklearn.metrics import roc_auc_score, roc_curve

from sklearn.preprocessing import LabelBinarizer
```

```
import matplotlib.pyplot as plt
import keras

# Load the saved model
model = keras.models.load_model('demo1/mobilenet.h5')

class_names = list(test_generator.class_indices.keys())
# Make predictions on the test data
y_pred_proba = model.predict(test_generator)

# Calculate the AUC for each class
lb = LabelBinarizer()
lb.fit(test_generator.classes)
y_true = lb.transform(test_generator.classes)
aucs = []
for i in range(test_generator.num_classes):
    auc = roc_auc_score(y_true[:, i], y_pred_proba[:, i])
    aucs.append(auc)

# Plot the ROC curve
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(test_generator.num_classes):
```

```
    fpr[i], tpr[i], _ = roc_curve(y_true[:, i], y_pred_proba[:, i])
    roc_auc[i] = aucs[i]

plt.figure(figsize=(8, 8))
colors = ['blue', 'red', 'green', 'orange']
for i, color in zip(range(test_generator.num_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(class_names[i], roc_auc[i]))

plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Multi-Class ROC Curve')
plt.legend(loc="lower right")
plt.show()
```



Final model comparison table

Model name	Training accuracy	Validation accuracy	Testing accuracy
Alexnet	77.58	58.03	64.75
Resnet50	66.42	66.73	66.46
Inceptionresnetv2	89.84	73.30	79.80
ann	71.05	75.42	77.625
mobilenet	92.32	88.71	89.025

CONCLUSION

In this project, we have successfully created an application which takes in an X-ray image and classify the disease accordingly. This application saves a lot of time and is very cheap to operate. This application can also be easily scaled up to handle large amount of tests which generally happens during a pandemic. Faster prediction also results in faster treatment and low chances of contagion to the public and also provided the detailed experimental analysis for each model by the test metrics like test accuracy,confusion matrix,roc curve

REFERENCES

1. https://www.sciencedirect.com/science/article/pii/S1746809422002257
2. https://www.sciencedirect.com/science/article/pii/S0378437122003521
3. https://www.ijser.org/researchpaper/COVID-19-Trend-Analysis-using-Machine-Learning-Techniques.pdf
4. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8138040/
5. https://ieeexplore.ieee.org/document/9623065
6. https://ieeexplore.ieee.org/document/9651804
7. https://www.sciencedirect.com/science/article/pii/S2214250921000500
8. https://ieeexplore.ieee.org/document/9215356/references#references
9. https://www.sciencedirect.com/science/article/pii/S0010482522008319
10. https://www.sciencedirect.com/science/article/pii/S0010482522008009
11. https://www.sciencedirect.com/science/article/pii/S1568494622007050
12. https://www.sciencedirect.com/science/article/pii/S2049080122002795