Detecting COVID-19 From Chest X-Rays Using Deep Learning Techniques

Date	01-11-2023
Team ID	Team-592660
Project Name	Detecting COVID-19 From Chest X-Rays Using Deep Learning Techniques

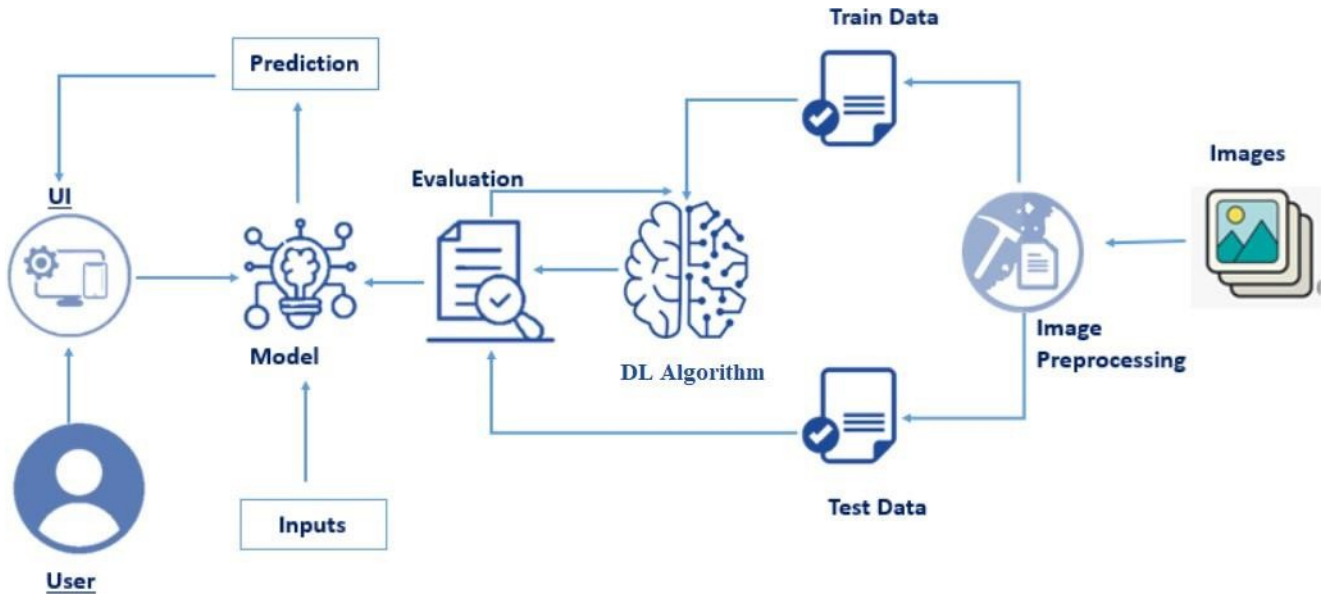
Project Description:

COVID-19 (coronavirus disease 2019) is an infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), which is a strain of coronavirus. The disease was officially announced as a pandemic by the World Health Organization (WHO) on 11 March 2020. Given spikes in new COVID-19 cases and the re-opening of daily activities around the world, the demand for curbing the pandemic is to be more emphasized. Medical images and artificial intelligence (AI) have been found useful for rapid assessment to provide treatment of COVID-19 infected patients. The PCR test may take several hours to become available, information revealed from the chest X-ray plays an important role for a rapid clinical assessment. This means if the clinical condition and the chest X-ray are normal, the patient is sent home while awaiting the results of the etiological test. But if the X-ray shows pathological findings, the suspected patient will be admitted to the hospital for close monitoring. Chest X-ray data have been found to be very promising for assessing COVID-19 patients, especially for resolving emergency-department and urgent-care-center overcapacity. Deeplearning (DL) methods in artificial intelligence (AI) play a dominant role as high performance classifiers in the detection of the disease using chest X-rays.

One of the biggest challenges following the Covid-19 pandemic is the detection of the disease in patients. To address this challenge we have been using the Deep Learning Algorithm to build an image recognition model that can detect the presence of Covid-19 from an X-Ray or CT-Scan image of a patient's lungs.

Transfer learning has become one of the most common techniques that has achieved better performance in many areas, especially in medical image analysis and classification. We used Transfer Learning techniques like Inception V3,Resnet50,Xception V3 that are more widely used as a transfer learning method in medical image analysis and they are highly effective.

Technical Architecture:



Prerequisites:

To complete this project, you must require the following software's, concepts, and packages

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, crossplatform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

Link: Click here to watch the video

1. **To build Machine learning models you must require the following packages**

- **Numpy**:
  - It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations

- **Scikit-learn:**

  - It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy

- **Flask:**
  Web framework used for building Web applications

- **Python packages:**
  - open anaconda prompt as administrator
  - Type “pip install numpy” and click enter.
  - Type “pip install pandas” and click enter.
  - Type “pip install scikit-learn” and click enter.
  - Type “pip install tensorflow==2.3.2” and click enter.
  - Type “pip install keras==2.3.1” and click enter.
  - Type “pip install Flask” and click enter.

- **Deep Learning Concepts**

  - **CNN:** a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNN Basic

  - **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

Flask Basics

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques of Convolutional Neural Network.
- Gain a broad understanding of image data.
- Know how to pre-process/clean the data using different data preprocessing techniques.
- know how to build a web application using the Flask framework.

Project Flow:

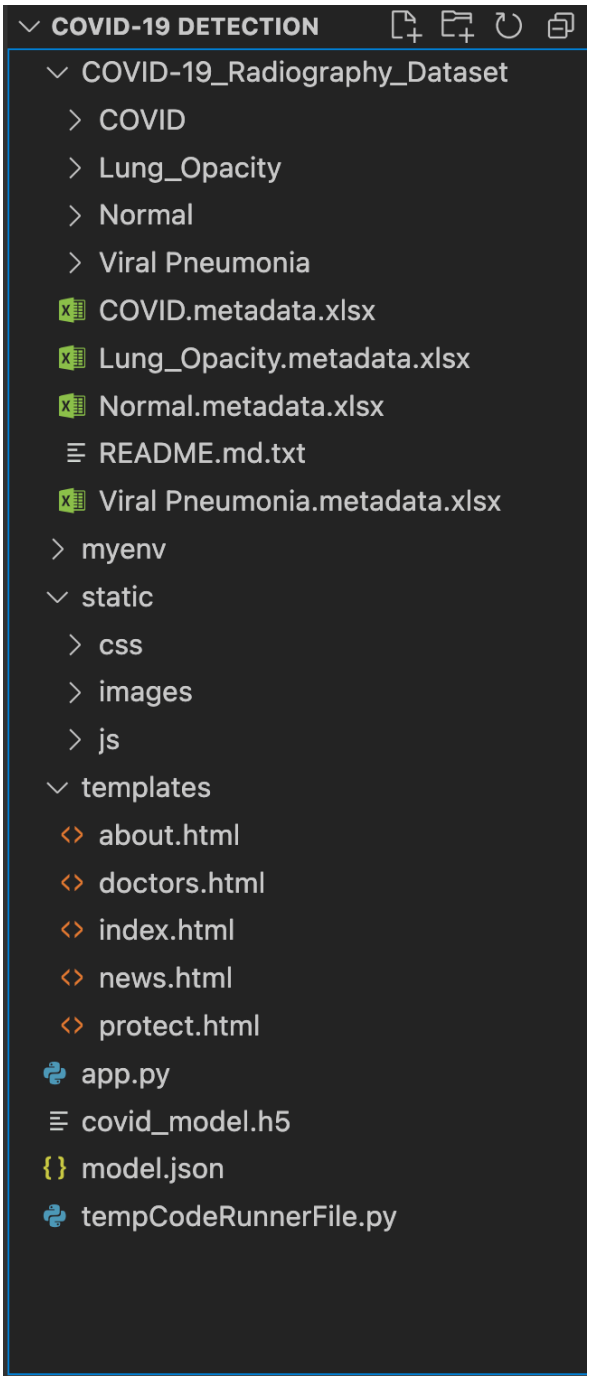
- The user interacts with the UI (User Interface) to choose the image.
- The chosen image analyzed by the model which is integrated with flask application.
- CNN Models analyze the image, then prediction is showcased on the Flask UI.

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
- Create Train and Test Folders.
- Data Preprocessing.
- Import the Image Data Generator library and Configure the ImageDataGenerator class.
- Apply ImageDataGenerator functionality to Trainset and Testset.
- Model Building.
- Import the model building Libraries.
- Initializing the model.
- Adding Input Layer.
- Adding Hidden Layer.
- Adding Output Layer.
- Configure the Learning Process.
- Training and testing the model.
- Save the Model.
- Application Building.
- Create an HTML file.
- Build Python Code.

Project Structure:

Create a Project folder which contains files as shown below:



- The Dataset folder contains the training and testing images for training our model.
- We are building a Flask Application that needs HTML pages stored in the **templates**
  folder and a python script **app.py** for server side scripting
- we need the model which is saved and the saved model in this content is a **Covid.h5**

- templates folder contains base.html,index.html pages.

Import the required Libraries for training_testing and adding image layers

```
import os
import pathlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
import cv2
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Activation, Conv2D, MaxPool2D, Flatten, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping
import tensorflow as tf
from google.colab import files
from sklearn.metrics import classification_report,confusion_matrix
```

Milestone 1: Data Collection and Preparation

Data is obtained from the Kaggle dataset "sid321axn/covid-cxr-image-dataset-research." The dataset is downloaded and unzipped.

The structure of the dataset is explored to understand the directory and file organization.

Download the Dataset: https://www.kaggle.com/datasets/sid321axn/covid-cxr-image-dataset-research

```
import os
os.environ["KAGGLE_CONFIG_DIR"] = "/content"
```

```
!kaggle datasets download -d sid321axn/covid-cxr-image-dataset-research
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /content/kaggle.json'
Downloading covid-cxr-image-dataset-research.zip to /content
 99% 568M/572M [00:04<00:00, 156MB/s]
100% 572M/572M [00:04<00:00, 124MB/s]
```

Downloading and unzipping Dataset :

```
!unzip \*.zip
```

```
Archive:  covid-cxr-image-dataset-research.zip
  inflating: COVID_IEEE/covid/01E392EE-69F9-4E33-BFCE-E5C968654078.jpeg
  inflating: COVID_IEEE/covid/03BF7561-A9BA-4C3C-B8A0-D3E585F73F3C.jpeg
  inflating: COVID_IEEE/covid/1-s2.0-S0140673620303706-fx1_lrg.jpg
  inflating: COVID_IEEE/covid/1-s2.0-S0929664620300449-gr2_lrg-a.jpg
  inflating: COVID_IEEE/covid/1-s2.0-S0929664620300449-gr2_lrg-b.jpg
  inflating: COVID_IEEE/covid/1-s2.0-S0929664620300449-gr2_lrg-c.jpg
  inflating: COVID_IEEE/covid/1-s2.0-S0929664620300449-gr2_lrg-d.jpg
  inflating: COVID_IEEE/covid/1-s2.0-S1684118220300608-main.pdf-001.jpg
  inflating: COVID_IEEE/covid/1-s2.0-S1684118220300608-main.pdf-002.jpg
  inflating: COVID_IEEE/covid/1-s2.0-S1684118220300682-main.pdf-002-a1.png
  inflating: COVID_IEEE/covid/1-s2.0-S1684118220300682-main.pdf-002-a2.png
  inflating: COVID_IEEE/covid/1-s2.0-S1684118220300682-main.pdf-003-b1.png
  inflating: COVID_IEEE/covid/1-s2.0-S1684118220300682-main.pdf-003-b2.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-000-fig1a.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-000-fig1b.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-001-fig2a.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-001-fig2b.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-002-fig3a.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-002-fig3b.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-003-fig4a.png
  inflating: COVID_IEEE/covid/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-003-fig4b.png
  inflating: COVID_IEEE/covid/1312A392-67A3-4EBF-9319-810CF6DA5EF6.jpeg
  inflating: COVID_IEEE/covid/16654_1_1.png
  inflating: COVID_IEEE/covid/16654_2_1.jpg
```

Contents of Dataset :

```
for dirpath,dirnames,filenames in os.walk("/content/COVID_IEEE"):
    print(f"there are {len(dirnames)} directories and {len(filenames)} images in '{dirpath}'.")
```

```
there are 3 directories and 0 images in '/content/COVID_IEEE'.
there are 0 directories and 536 images in '/content/COVID_IEEE/covid'.
there are 0 directories and 619 images in '/content/COVID_IEEE/virus'.
there are 0 directories and 668 images in '/content/COVID_IEEE/normal'.
```

Milestone 2: Image Preprocessing

In this milestone we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

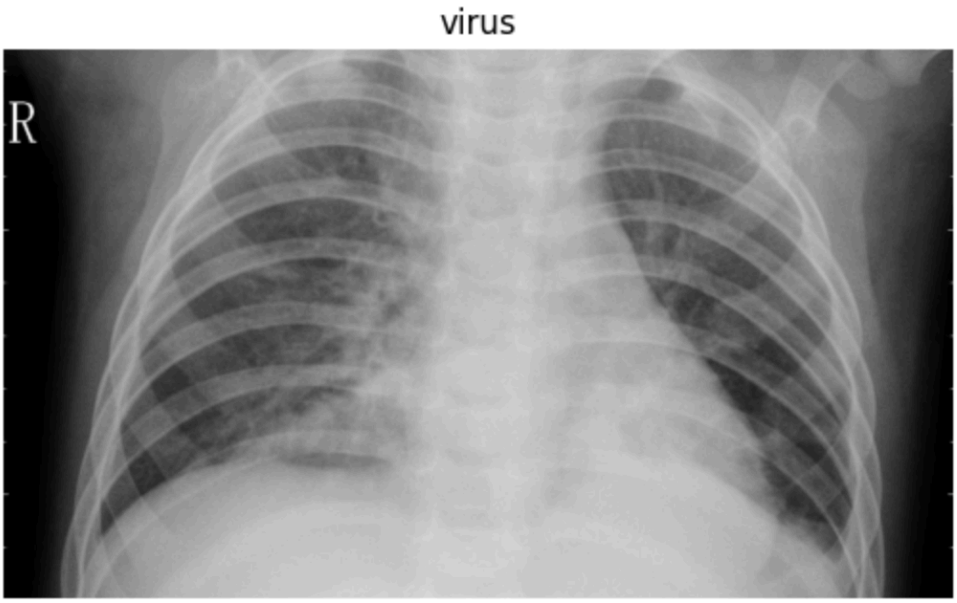
Data Visualization:
Random images from each class (COVID-19, normal, virus) are visualized to gain insights into the data.
The **view_image** function is used to display random images from each class.

```python
def view_image(target_dir, target_class):
    target_folder = target_dir+target_class
    random_image = random.sample(os.listdir(target_folder),1)
    print(random_image)
    img = mpimg.imread(target_folder+"/"+ random_image[0])
    plt.imshow(img, cmap ="gray")
    plt.title(target_class)
    plt.axis("off")
    print(f"image shape {img.shape}")

    return img
```

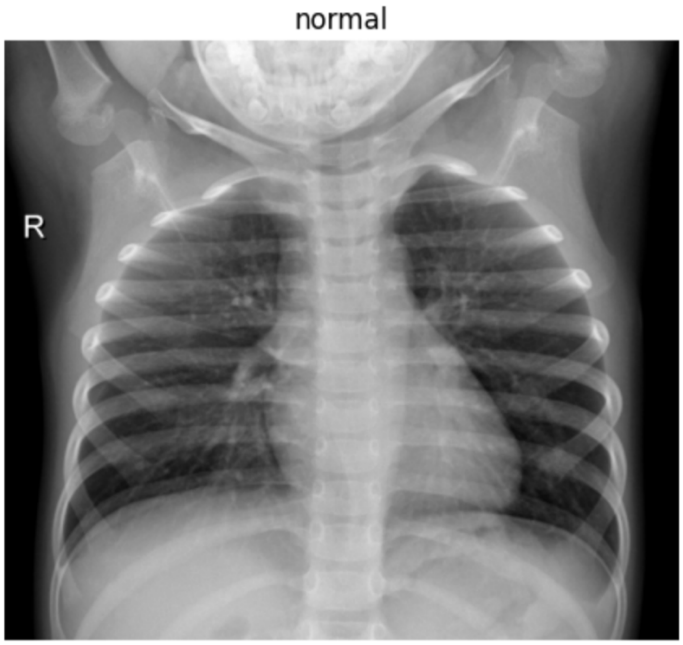
```python
img = view_image("/content/COVID_IEEE/","virus")
```

```
['person516_virus_1033.jpeg']
image shape (648, 1120)
```



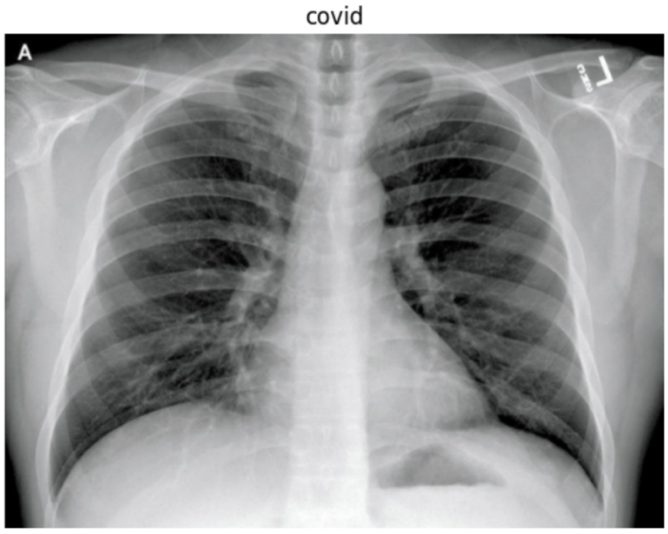
```
img = view_image("/content/COVID_IEEE/","normal")
```

```
['NORMAL2-IM-0692-0001.jpeg']
image shape (1534, 1724)
```




```
img = view_image("/content/COVID_IEEE/","covid")
```

```
['covid190026.png']
image shape (480, 640, 3)
```



Step 3: Data Pre-processing:

Images are normalized by dividing pixel values by 255.0.
Labels are one-hot encoded to represent the three classes.

```
data=[]
labels=[]
covid=os.listdir("/content/COVID_IEEE/covid/")
for a in covid:

    image = cv2.imread("/content/COVID_IEEE/covid/"+a,)
    image = cv2.resize(image, (224, 224))


    data.append(image)
    labels.append(0)
```

For Covid cases we append label as '0' .

```
normal=os.listdir("/content/COVID_IEEE/normal/")
for a in normal:

    image = cv2.imread("/content/COVID_IEEE/normal/"+a,)
    image = cv2.resize(image, (224, 224))



    data.append(image)
    labels.append(1)
```

For Normal cases we append label as '1' .

```
virus=os.listdir("/content/COVID_IEEE/virus/")
for a in virus:

    image = cv2.imread("/content/COVID_IEEE/virus/"+a,)
    image = cv2.resize(image, (224, 224))


    data.append(image)
    labels.append(2)
```

For Virus cases we append label as '2' .

Images are loaded and resized to a common size (224x224 pixels).

```
data = np.array(data) / 255.0
img_labels = np.array(labels)
```

Milestone 3: Model Building

Now it's time to build our Convolutional Neural Networking which contains an input layer along with the convolution, max-pooling, and finally an output layer.

```
X_train, X_test, y_train, y_test = train_test_split(data, img_labels, test_size=0.20, random_state=42)
y_train = tf.keras.utils.to_categorical(y_train, num_classes=3)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=3)
```

Activity 1: Initializing the model

Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add() method.

```
#Initialzing model
model = Sequential()
```

Activity 2: Adding CNN Layers

- For information regarding CNN Layers refer to the link Link: https://victorzhou.com/blog/intro-to-cnns-part-1/
- As the input image contains three channels, we are specifying the input shape as (256,256,3).

- We are adding a convolution layer with activation function as “relu” and with a small filter size (3,3) and the number of filters (32) followed by a max-pooling layer.
- Max pool layer is used to down sample the input. (Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter)
- Flatten layer flattens the input. Does not affect the batch size.

```
#Block Number 1
model.add(Conv2D(input_shape = (224,224,3), filters=32,padding="same", kernel_size= (3,3)))
model.add(Activation("relu"))

model.add(Conv2D(filters=32,padding="same", kernel_size= (3,3)))
model.add(Activation("relu"))

model.add(MaxPool2D((2,2)))

#Block Number 2
model.add(Conv2D(filters=64,padding="same", kernel_size= (3,3)))
model.add(Activation("relu"))


model.add(Conv2D(filters=64,padding="same", kernel_size= (3,3)))
model.add(Activation("relu"))

model.add(MaxPool2D((2,2)))

#Block Number 3
model.add(Conv2D(filters=128,padding="same", kernel_size= (3,3)))
model.add(Activation("relu"))

model.add(Conv2D(filters=128,padding="same", kernel_size= (3,3)))
model.add(Activation("relu"))

model.add(MaxPool2D((2,2)))

model.add(MaxPool2D((2,2)))

# Fully Connected layer
model.add(Flatten())

model.add(Dense(units=1024, activation="relu"))

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


model.add(Dense(units=3, activation="softmax"))

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

Activity 3: Adding Dense Layer

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.

```
model = Flatten()(base_model.output)

model = Dense(units=1024, activation="relu")(model)
model = Dense(units=512, activation="relu")(model)
model = Dense(units=256, activation="relu")(model)

prediction_layer = Dense(units=3, activation="softmax")(model)
```

```
model = Model(inputs = base_model.input, outputs = prediction_layer)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities.

Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

Summary Of Model:

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizati  (None, 112, 112, 32)      128
 on)

 conv1_relu (ReLU)           (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D  (None, 112, 112, 32)      288
 )

 conv_dw_1_bn (BatchNormali  (None, 112, 112, 32)      128
 zation)

 conv_dw_1_relu (ReLU)       (None, 112, 112, 32)      0

 conv_pw_1 (Conv2D)          (None, 112, 112, 64)      2048
```

```
conv_pw_1_bn (BatchNormali  (None, 112, 112, 64)      256
zation)

conv_pw_1_relu (ReLU)       (None, 112, 112, 64)      0

conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 64)      0

conv_dw_2 (DepthwiseConv2D  (None, 56, 56, 64)        576
)

conv_dw_2_bn (BatchNormali  (None, 56, 56, 64)        256
zation)

conv_dw_2_relu (ReLU)       (None, 56, 56, 64)        0

conv_pw_2 (Conv2D)          (None, 56, 56, 128)       8192

conv_pw_2_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

conv_dw_3 (DepthwiseConv2D  (None, 56, 56, 128)       1152
)

conv_dw_3_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_dw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pw_3 (Conv2D)          (None, 56, 56, 128)       16384

conv_pw_3_bn (BatchNormali  (None, 56, 56, 128)       512
zation)

conv_pw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pad_4 (ZeroPadding2D)  (None, 57, 57, 128)       0

conv_dw_4 (DepthwiseConv2D  (None, 28, 28, 128)       1152
)

conv_dw_4_bn (BatchNormali  (None, 28, 28, 128)       512
zation)

conv_pw_4_relu (ReLU)       (None, 28, 28, 256)       0

conv_dw_5 (DepthwiseConv2D  (None, 28, 28, 256)       2304
)

conv_dw_5_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)

conv_dw_5_relu (ReLU)       (None, 28, 28, 256)       0
```

```
conv_pw_5 (Conv2D)           (None, 28, 28, 256)       65536

conv_pw_5_bn (BatchNormali  (None, 28, 28, 256)       1024
zation)

conv_pw_5_relu (ReLU)        (None, 28, 28, 256)       0

conv_pad_6 (ZeroPadding2D)  (None, 29, 29, 256)       0

conv_dw_6 (DepthwiseConv2D  (None, 14, 14, 256)       2304
)

conv_dw_6_bn (BatchNormali  (None, 14, 14, 256)       1024
zation)

conv_dw_6_relu (ReLU)        (None, 14, 14, 256)       0

conv_pw_6 (Conv2D)           (None, 14, 14, 512)       131072

conv_pw_6_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_6_relu (ReLU)        (None, 14, 14, 512)       0

conv_dw_7 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_7_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_7_relu (ReLU)        (None, 14, 14, 512)       0
conv_dw_7 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_7_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_7_relu (ReLU)        (None, 14, 14, 512)       0

conv_pw_7 (Conv2D)           (None, 14, 14, 512)       262144

conv_pw_7_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_7_relu (ReLU)        (None, 14, 14, 512)       0

conv_dw_8 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_8_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_8_relu (ReLU)        (None, 14, 14, 512)       0

conv_pw_8 (Conv2D)           (None, 14, 14, 512)       262144
```

```
conv_pw_8_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_8_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_9 (DepthwiseConv2D  (None, 14, 14, 512)       4608
)

conv_dw_9_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_dw_9_relu (ReLU)       (None, 14, 14, 512)       0

conv_pw_9 (Conv2D)          (None, 14, 14, 512)       262144

conv_pw_9_bn (BatchNormali  (None, 14, 14, 512)       2048
zation)

conv_pw_9_relu (ReLU)       (None, 14, 14, 512)       0

conv_dw_10 (DepthwiseConv2  (None, 14, 14, 512)       4608
D)

conv_dw_10_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_dw_10_relu (ReLU)      (None, 14, 14, 512)       0

conv_pw_10 (Conv2D)         (None, 14, 14, 512)       262144

conv_pw_10_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_pw_10_relu (ReLU)      (None, 14, 14, 512)       0

conv_dw_11 (DepthwiseConv2  (None, 14, 14, 512)       4608
D)

conv_dw_11_bn (BatchNormal  (None, 14, 14, 512)       2048
ization)

conv_pw_11_relu (ReLU)      (None, 14, 14, 512)       0

conv_pad_12 (ZeroPadding2D  (None, 15, 15, 512)       0
)

conv_dw_12 (DepthwiseConv2  (None, 7, 7, 512)         4608
D)

conv_dw_12_bn (BatchNormal  (None, 7, 7, 512)         2048
ization)
```

```
 conv_dw_12_relu (ReLU)      (None, 7, 7, 512)         0

 conv_pw_12 (Conv2D)         (None, 7, 7, 1024)        524288

 conv_pw_12_bn (BatchNormal  (None, 7, 7, 1024)        4096
 ization)

 conv_pw_12_relu (ReLU)      (None, 7, 7, 1024)        0

 conv_dw_13 (DepthwiseConv2  (None, 7, 7, 1024)        9216
 D)

 conv_dw_13_bn (BatchNormal  (None, 7, 7, 1024)        4096
 ization)

 conv_dw_13_relu (ReLU)      (None, 7, 7, 1024)        0

 conv_pw_13 (Conv2D)         (None, 7, 7, 1024)        1048576

 conv_pw_13_bn (BatchNormal  (None, 7, 7, 1024)        4096
 ization)

 conv_pw_13_relu (ReLU)      (None, 7, 7, 1024)        0

 flatten_1 (Flatten)         (None, 50176)             0

 dense_3 (Dense)             (None, 1024)              51381248

 dense_4 (Dense)             (None, 512)               524800

 dense_5 (Dense)             (None, 256)               131328

 dense_6 (Dense)             (None, 3)                 771

=================================================================
Total params: 55267011 (210.83 MB)
Trainable params: 52038147 (198.51 MB)
Non-trainable params: 3228864 (12.32 MB)
_________________________________________________________________
```

Activity 4: Configure The Learning Process

- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.
- Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
- Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process

Compiling the Model:

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

Activity 5: Train The model

Now, let us train our model with our image dataset. The model is trained for 30 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 20 epochs and probably there is further scope to improve the model.

fit_generator functions used to train a deep learning neural network **Arguments:**

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.

- Epochs: an integer and number of epochs we want to train our model for.

- validation_data can be either:
  - an inputs and targets list
  - a generator
  - an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

```
model.fit(X_train, y_train, validation_split=0.3, epochs=20, batch_size=32)
```

```
Epoch 1/20
32/32 [==============================] - 9s 155ms/step - loss: 7.7835 - accuracy: 0.7892 - val_loss: 1.3070 - val_accuracy: 0.8950
Epoch 2/20
32/32 [==============================] - 2s 73ms/step - loss: 0.6017 - accuracy: 0.9461 - val_loss: 1.0233 - val_accuracy: 0.9315
Epoch 3/20
32/32 [==============================] - 2s 76ms/step - loss: 0.3514 - accuracy: 0.9765 - val_loss: 0.5383 - val_accuracy: 0.9566
Epoch 4/20
32/32 [==============================] - 2s 75ms/step - loss: 0.1303 - accuracy: 0.9863 - val_loss: 1.0034 - val_accuracy: 0.9269
Epoch 5/20
32/32 [==============================] - 2s 77ms/step - loss: 0.1797 - accuracy: 0.9843 - val_loss: 0.9130 - val_accuracy: 0.9429
Epoch 6/20
32/32 [==============================] - 2s 77ms/step - loss: 0.3066 - accuracy: 0.9735 - val_loss: 1.4392 - val_accuracy: 0.9269
Epoch 7/20
32/32 [==============================] - 2s 74ms/step - loss: 0.0689 - accuracy: 0.9951 - val_loss: 0.8624 - val_accuracy: 0.9475
Epoch 8/20
32/32 [==============================] - 2s 74ms/step - loss: 0.0814 - accuracy: 0.9941 - val_loss: 0.9533 - val_accuracy: 0.9338
Epoch 9/20
32/32 [==============================] - 2s 74ms/step - loss: 0.0226 - accuracy: 0.9961 - val_loss: 0.8741 - val_accuracy: 0.9224
Epoch 10/20
32/32 [==============================] - 2s 76ms/step - loss: 0.0313 - accuracy: 0.9941 - val_loss: 0.8525 - val_accuracy: 0.9521
Epoch 11/20
32/32 [==============================] - 3s 83ms/step - loss: 0.0684 - accuracy: 0.9941 - val_loss: 0.4911 - val_accuracy: 0.9612
Epoch 12/20
32/32 [==============================] - 2s 76ms/step - loss: 0.0428 - accuracy: 0.9951 - val_loss: 0.8066 - val_accuracy: 0.9498
Epoch 13/20
32/32 [==============================] - 2s 76ms/step - loss: 0.0508 - accuracy: 0.9931 - val_loss: 1.2184 - val_accuracy: 0.9178
Epoch 14/20
32/32 [==============================] - 2s 75ms/step - loss: 0.0958 - accuracy: 0.9853 - val_loss: 0.7644 - val_accuracy: 0.9543
Epoch 15/20
32/32 [==============================] - 2s 77ms/step - loss: 0.1371 - accuracy: 0.9882 - val_loss: 0.7807 - val_accuracy: 0.9589
Epoch 16/20
32/32 [==============================] - 3s 80ms/step - loss: 0.1116 - accuracy: 0.9922 - val_loss: 2.9036 - val_accuracy: 0.8973
Epoch 17/20
32/32 [==============================] - 2s 75ms/step - loss: 0.0733 - accuracy: 0.9922 - val_loss: 1.1088 - val_accuracy: 0.9452
Epoch 18/20
32/32 [==============================] - 2s 76ms/step - loss: 0.0399 - accuracy: 0.9941 - val_loss: 0.9273 - val_accuracy: 0.9361
Epoch 19/20
32/32 [==============================] - 2s 74ms/step - loss: 0.0214 - accuracy: 0.9971 - val_loss: 0.8087 - val_accuracy: 0.9452
Epoch 20/20
32/32 [==============================] - 2s 76ms/step - loss: 0.0161 - accuracy: 0.9951 - val_loss: 0.5642 - val_accuracy: 0.9566
<keras.src.callbacks.History at 0x780bc007e710>
```

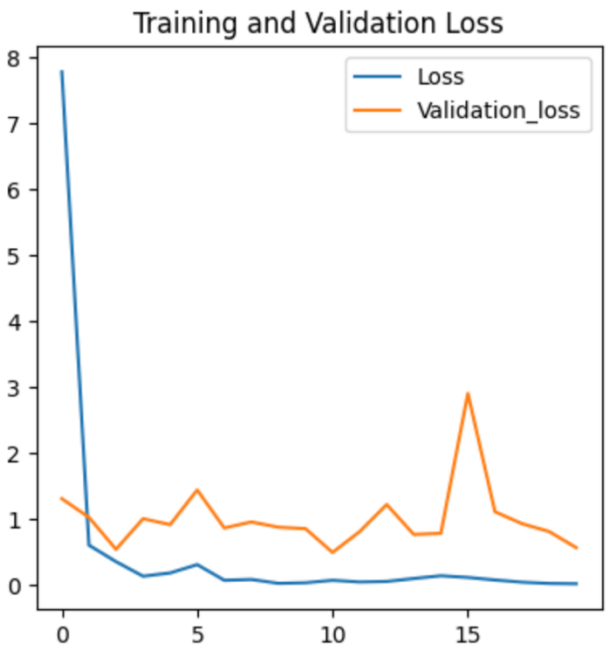
Plotting the data metrics:

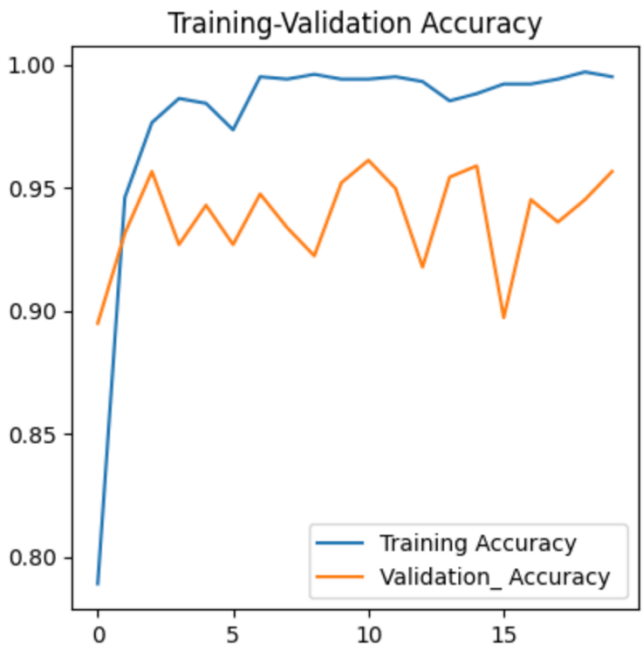
```
#saving the model history
loss = pd.DataFrame(model.history.history)

#plotting the loss and accuracy
plt.figure(figsize=(10,10))

plt.subplot(2,2,1)
plt.plot(loss["loss"], label ="Loss")
plt.plot(loss["val_loss"], label = "Validation_loss")
plt.legend()
plt.title("Training and Validation Loss")

plt.subplot(2,2,2)
plt.plot(loss['accuracy'],label = "Training Accuracy")
plt.plot(loss['val_accuracy'], label ="Validation_ Accuracy ")
plt.legend()
plt.title("Training-Validation Accuracy")
```





Classification Report:

```
predictions = model.predict(X_test)

y_pred = np.argmax(predictions, axis = 1)
y_test_new = np.argmax(y_test, axis = 1)
```

```
12/12 [==============================] - 1s 69ms/step
```

```
print(classification_report(y_test_new, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       108
           1       0.96      0.99      0.98       129
           2       0.98      0.96      0.97       128

    accuracy                           0.98       365
   macro avg       0.98      0.98      0.98       365
weighted avg       0.98      0.98      0.98       365
```

Data Matrix :

```
pd.DataFrame(confusion_matrix(y_test_new, y_pred),
             columns= ["covid", "normal", "virus"],
             index = ["covid", "normal", "virus"])
```

	covid	normal	virus
covid	107	0	1
normal	0	128	1
virus	0	5	123

Activity 6: Save the Model

The model is saved with .h5 extension as follows
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

Save the Model:

```
model.save("covid_model.h5")  # Save the model as an HDF5 file
```

Activity 7: Test The model

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data. Load the saved model using load_model

Taking an image as input and checking the results

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model

# Load the trained model
model = load_model("covid_model.h5")  # Replace with the path to your model file

# Load and preprocess the image you want to predict
image_path = "/content/COVID_IEEE/covid/covid1900294.png"  # Replace with the path to your image
img = cv2.imread(image_path)
img = cv2.resize(img, (224, 224))
img = img / 255.0  # Normalize the image

# Reshape the image to match the model's input shape
img = np.reshape(img, (1, 224, 224, 3))

# Make predictions
predictions = model.predict(img)

# Interpret the results
class_names = ["covid", "normal", "virus"]
predicted_class_index = np.argmax(predictions)
predicted_class = class_names[predicted_class_index]

print(f"The image is predicted as: {predicted_class}")
```

```
1/1 [==============================] - 0s 415ms/step
The image is predicted as: covid
```

By using the model we are predicting the output for the given input image as COVID

Milestone 4: Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

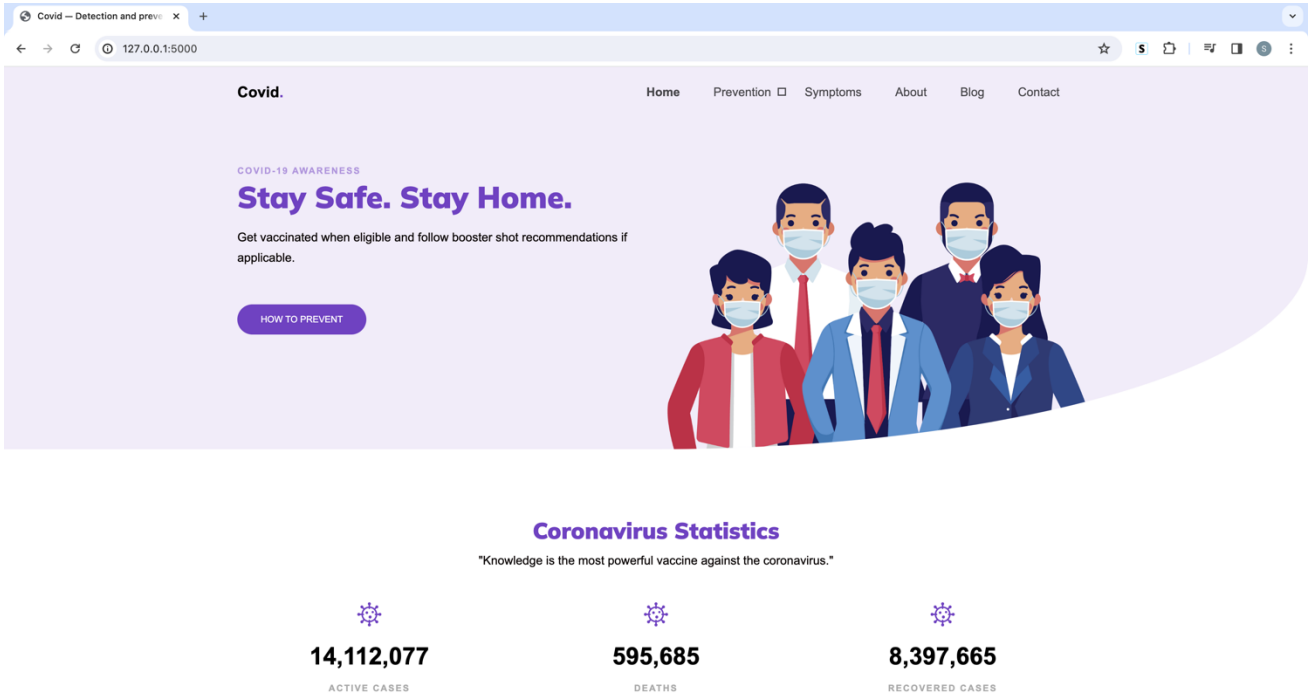
In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Garbage and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the “Image” button, the next page is opened where the user chooses the image and predicts the output.

Activity 1: Create HTML Pages

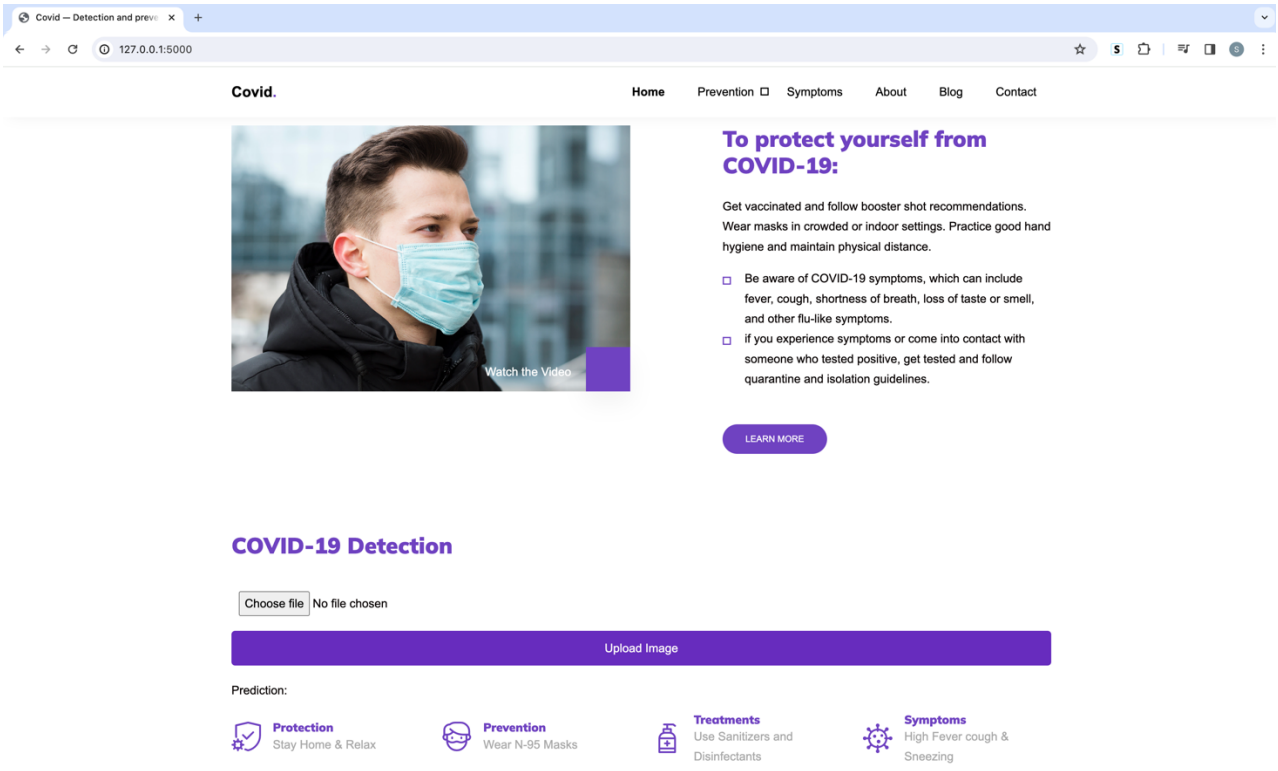
- We use HTML to create the front end part of the web page.
- Here, we have created 3 HTML pages- home.html, intro.html, and upload.html
- home.html displays the home page. o Intro.html displays an introduction about the project
- upload.html gives the emergency alert For more information https://www.w3schools.com/html/
- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

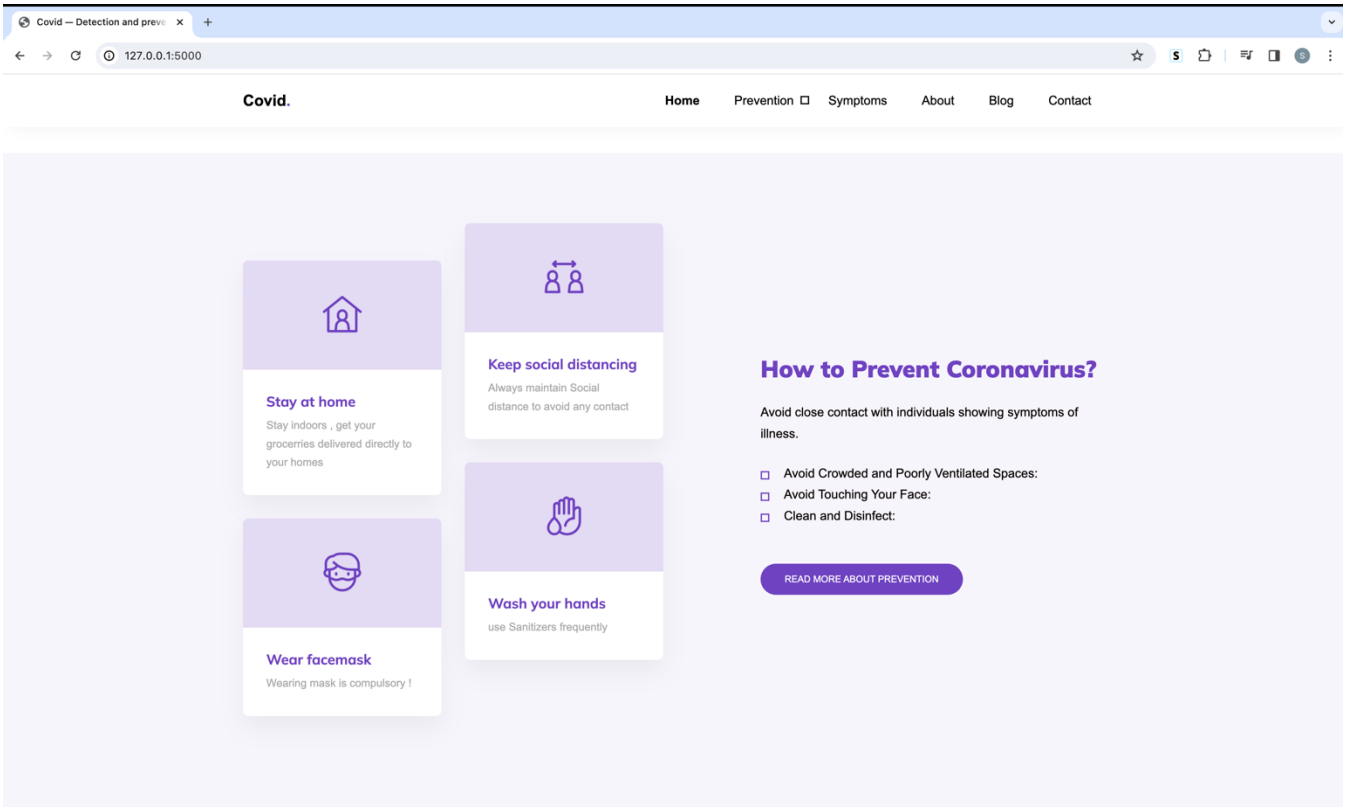
- **Link :**CSS , JS

index.html looks like this

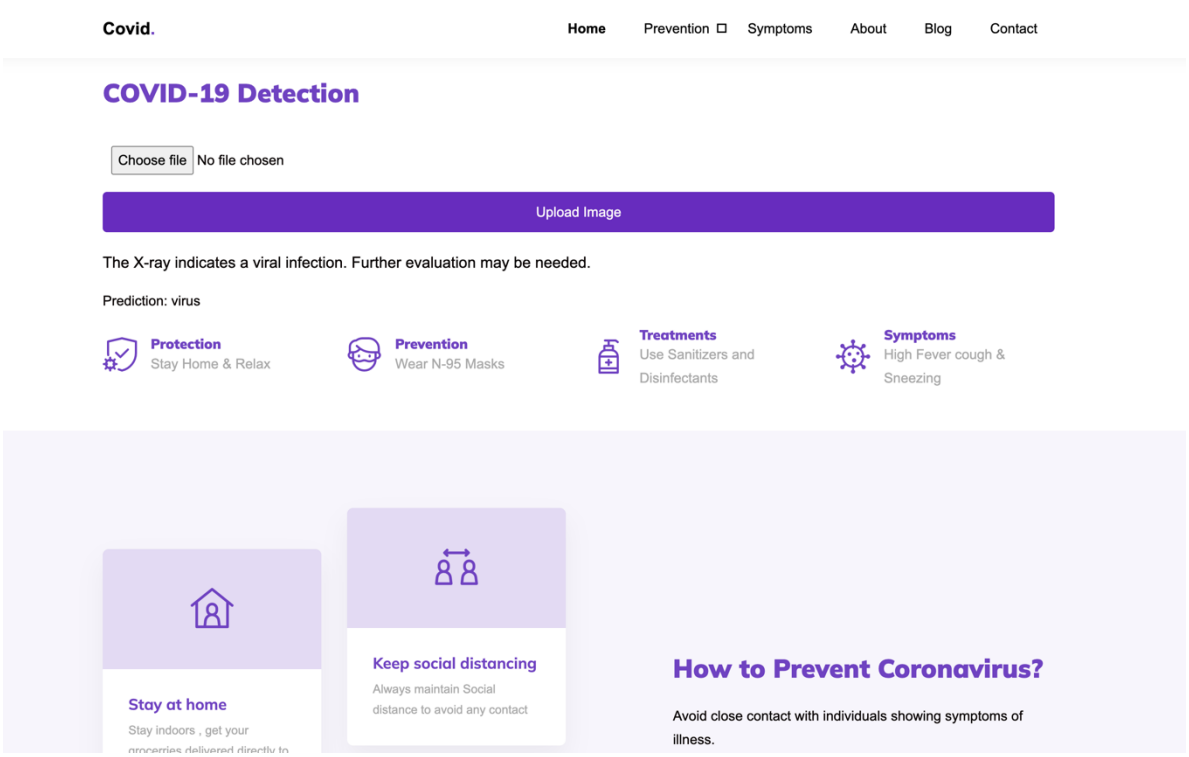


Prevention :

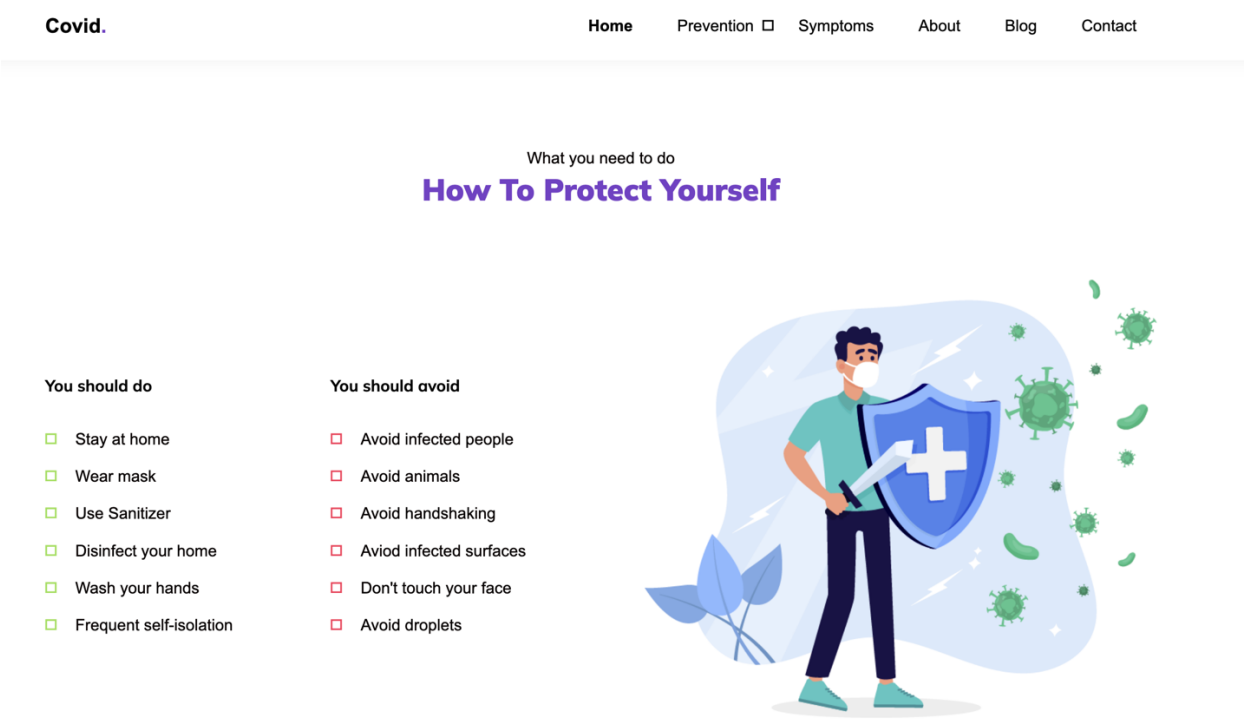

Covid — Detection and preve
127.0.0.1:5000
Covid.
Home
Prevention
Symptoms
About
Blog
Contact
Watch the Video
To protect yourself from COVID-19:
Get vaccinated and follow booster shot recommendations. Wear masks in crowded or indoor settings. Practice good hand hygiene and maintain physical distance.
Be aware of COVID-19 symptoms, which can include fever, cough, shortness of breath, loss of taste or smell, and other flu-like symptoms.
if you experience symptoms or come into contact with someone who tested positive, get tested and follow quarantine and isolation guidelines.
LEARN MORE
COVID-19 Detection
Choose file
No file chosen
Upload Image
Prediction:
Protection
Stay Home & Relax
Prevention
Wear N-95 Masks
Treatments
Use Sanitizers and Disinfectants
Symptoms
High Fever cough & Sneezing


Covid — Detection and preve
127.0.0.1:5000
Covid.
Home
Prevention
Symptoms
About
Blog
Contact
Stay at home
Stay indoors , get your grocerries delivered directly to your homes
Keep social distancing
Always maintain Social distance to avoid any contact
Wear facemask
Wearing mask is compulsory !
Wash your hands
use Sanitizers frequently
How to Prevent Coronavirus?
Avoid close contact with individuals showing symptoms of illness.
Avoid Crowded and Poorly Ventilated Spaces:
Avoid Touching Your Face:
Clean and Disinfect:
READ MORE ABOUT PREVENTION

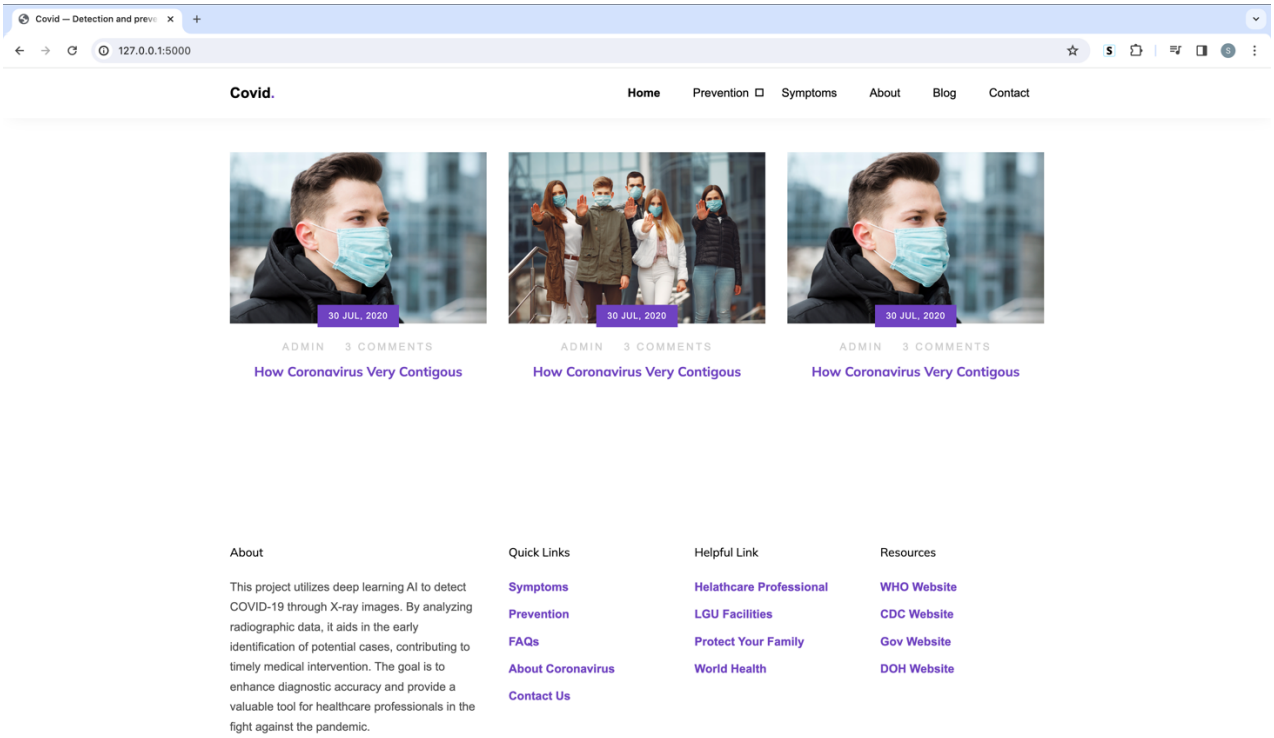
Predict section:



Protect Yourself:



Footer:



Activity 2: Build python code

Task 1: Importing Libraries

The first step is usually importing the libraries that will be needed in the program.

```
app.py > ...
1   from flask import Flask, render_template, request, jsonify
2   import tensorflow as tf
3   import numpy as np
```

Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (__name) as argument Pickle library to load the model file.

Task 2: Creating our flask application and loading our model by using load_model method

```
5    app = Flask(__name__)
6
7    # Load your Keras model
8    model = tf.keras.models.load_model("covid_model.h5")
9
```

Task 3: Routing to the html Page

Here, the declared constructor is used to route to the HTML page created earlier.
In the above example, '/' URL is bound with index.html function. Hence, when the home page of a web server is opened in the browser, the html page will be rendered.
Whenever you browse an image from the html page this photo can be accessed through POST or GET Method.

```
10   @app.route('/')
11   def index():
12       return render_template('index.html')
13
14   @app.route('/predict', methods=['POST'])
15   def predict():
```

Showcasing prediction on UI:

```
def predict():
    if 'image' not in request.files:
        return jsonify({'error': 'No file part'})

    image = request.files['image']
    image_np = tf.image.decode_image(image.read(), channels=3)
    image_np = tf.image.resize(image_np, (224, 224)) / 255.0

    # Perform model prediction
    prediction = model.predict(np.expand_dims(image_np, axis=0))
    class_names = ["covid", "normal", "virus"]
    predicted_class_index = np.argmax(prediction)
    predicted_class = class_names[predicted_class_index]
    messages = {
        "covid": "Based on the analysis, The X-ray shows signs of a COVID-19 infection.",
        "normal": "Great news! The X-ray appears to be normal with no signs of infection.",
        "virus": "The X-ray indicates a viral infection. Further evaluation may be needed."
    }
    return render_template('index.html', prediction=predicted_class, message=messages[predicted_class])
```

Here we are defining a function which requests the browsed file from the html page using the post method. The requested picture file is then saved to the uploads folder in this same directory using OS library. Using the load image class from Keras library we are retrieving the saved picture from the path declared. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class. This returns the name of the class. This name is rendered to the predict variable used in the html page.

Predicting the results

We then proceed to detect all type of Garbage in the input image using model.predict function and the result is stored in the result variable.

Final Run the application

This is used to run the application in a local host.

```
if __name__ == '__main__':
    app.run(debug=True)
```

Activity 3: Run the application

- Open the Vscode prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type “python app.py” command.
- It will show the local host where your app is running on **http://127.0.0.1.5000/**
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.

```
(base) s.hruthik@Hruthik-Macpro covid-master % python app.py
```

Then it will run on localhost:5000

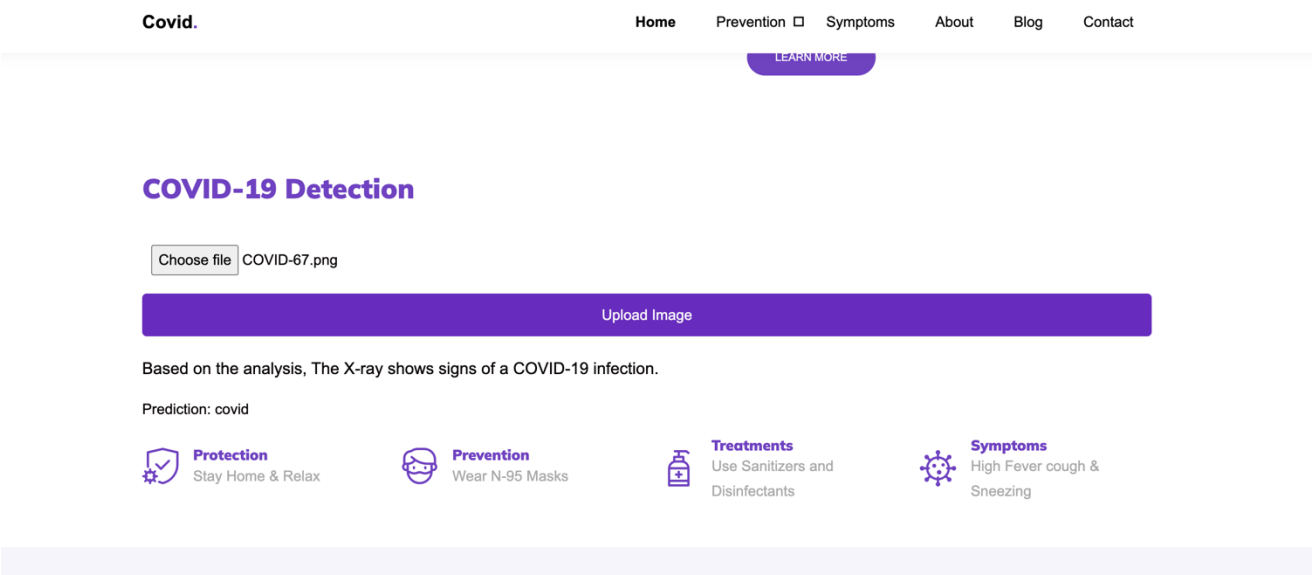
```
(base) s.hruthik@Hruthik-Macpro covid-master % python app.py
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use the legacy Keras optim
izer instead, located at `tf.keras.optimizers.legacy.Adam`.
WARNING:absl:There is a known slowdown when using v2.11+ Keras optimizers on M1/M2 Macs. Falling back to the legacy Keras optimizer, i.
e., `tf.keras.optimizers.legacy.Adam`.
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized optimizer
.
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized optimizer
.
 * Serving Flask app 'app'
 * Debug mode:
INFO:werkzeug:                                    lopment server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (fsevents)
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use the legacy Keras optim
izer instead, located at `tf.keras.optimizers.legacy.Adam`.
WARNING:absl:There is a known slowdown when using v2.11+ Keras optimizers on M1/M2 Macs. Falling back to the legacy Keras optimizer, i.
e., `tf.keras.optimizers.legacy.Adam`.
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized optimizer
.
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized optimizer
.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 187-490-681
```

Follow link (cmd + click)

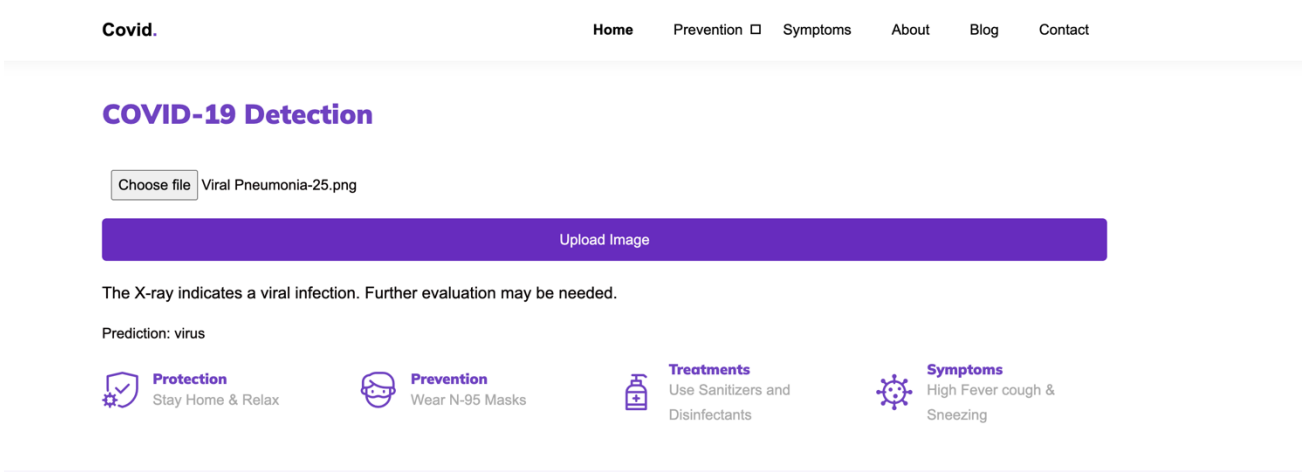
Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.

FINAL OUTPUTS:

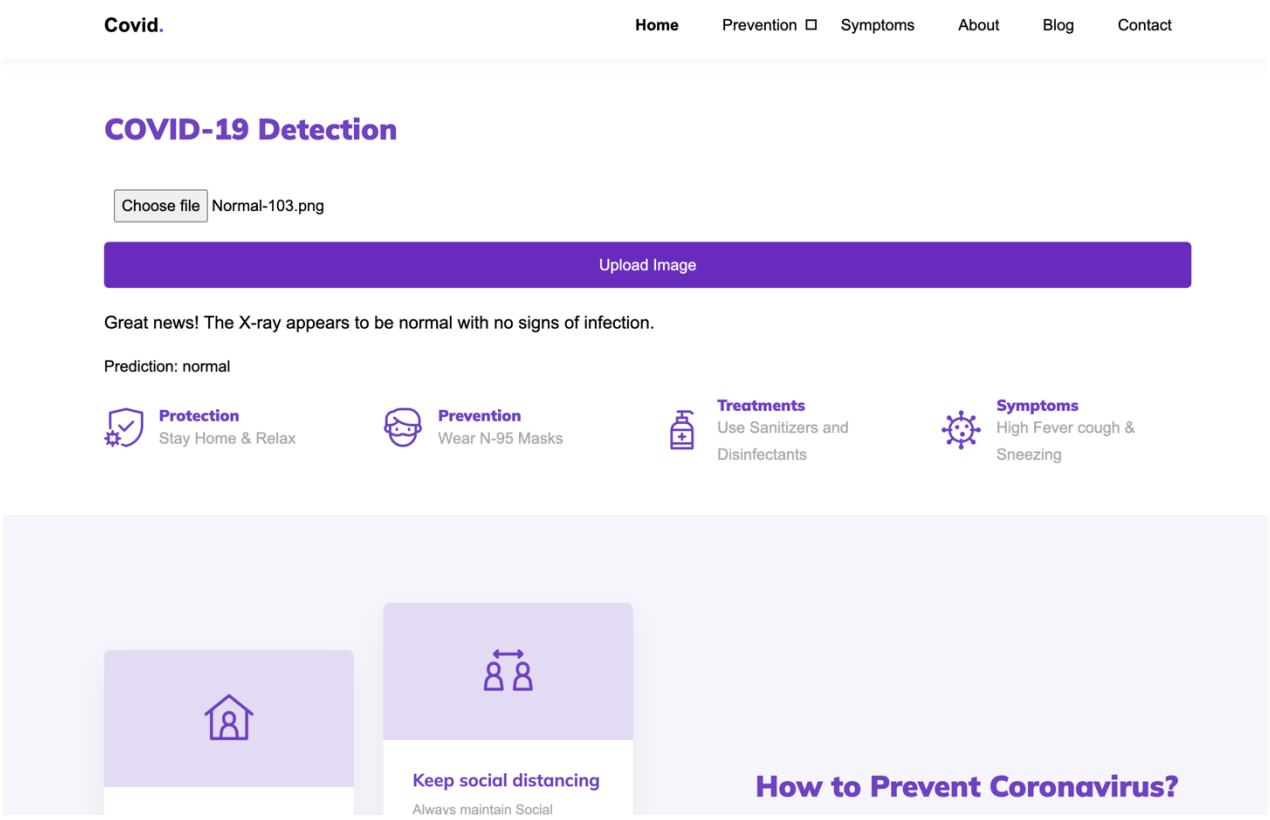
Output-1:



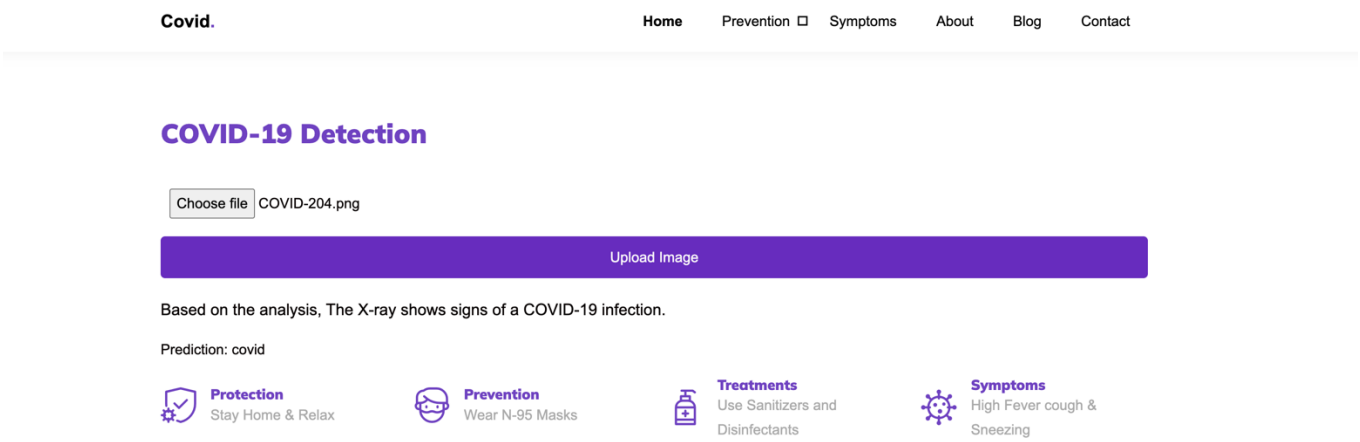
Output-2:



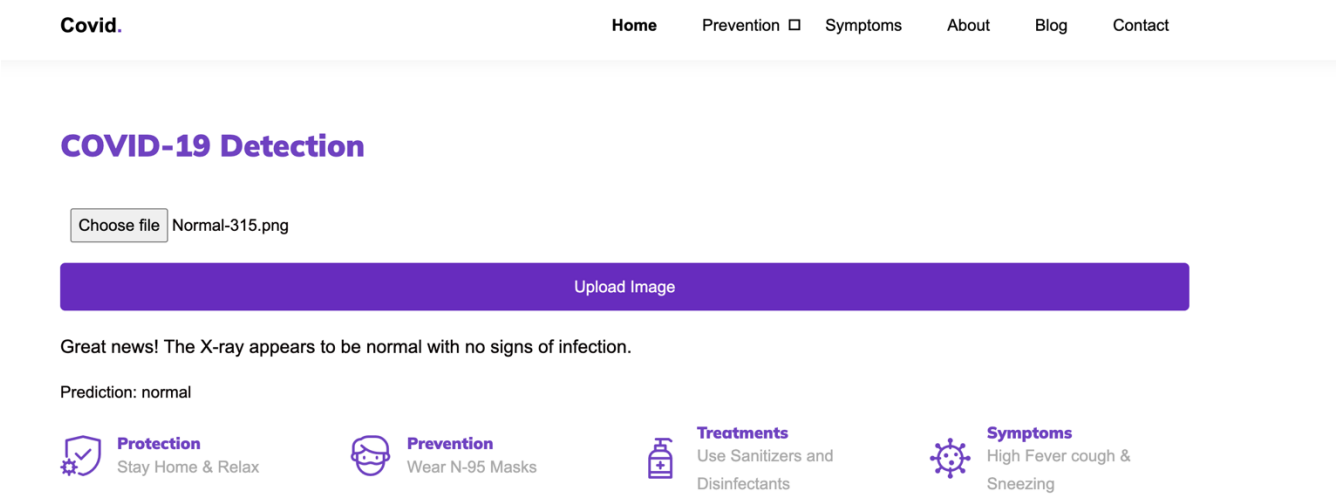
Output-3:



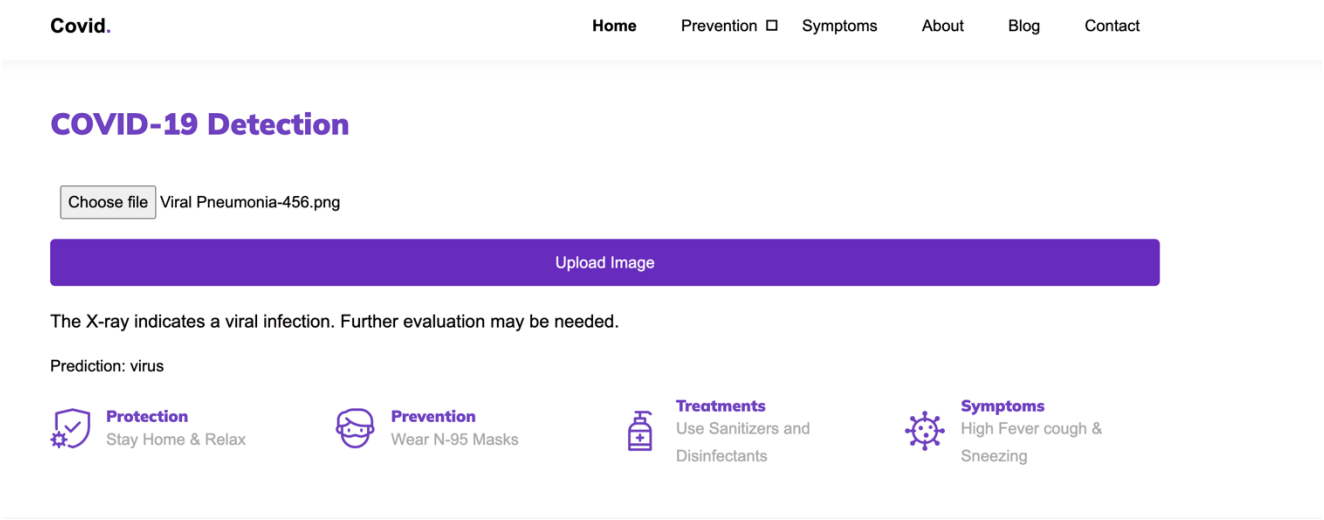
Output-4:



Output-5:



Output-6:


Covid.
Home
Prevention
Symptoms
About
Blog
Contact
COVID-19 Detection
Choose file
Viral Pneumonia-456.png
Upload Image
The X-ray indicates a viral infection. Further evaluation may be needed.
Prediction: virus
Protection
Stay Home & Relax
Prevention
Wear N-95 Masks
Treatments
Use Sanitizers and Disinfectants
Symptoms
High Fever cough & Sneezing