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

Date	01-11-2023
Team ID	Team-592696
Project Name	Detecting COVID-19 From Chest X-Rays Using Deep Learning Techniques

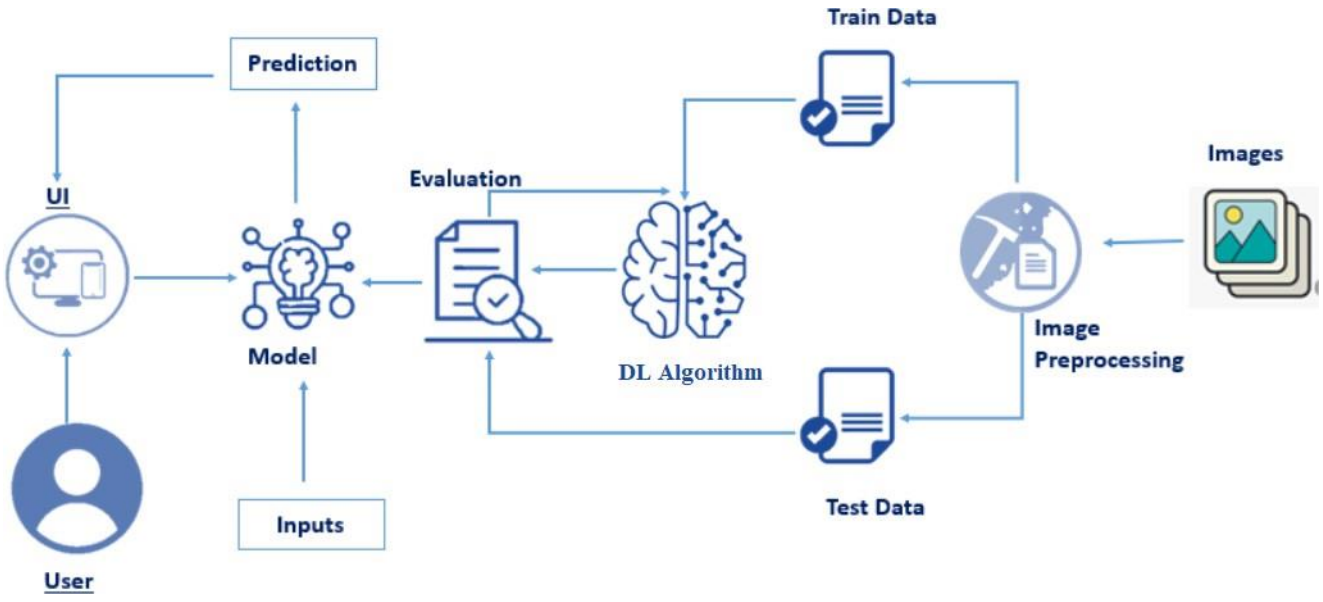
Project Description:

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 theresults of the etiological test. But if the X-ray shows pathological findings, the suspected patientwill 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. Deep-learning (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-Scanimage 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 usedas 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, cross-platform, 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, mostcommonly applied to analyzing visual imagery.
    CNN Basic

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

    Flask Basics

If you are using Pycharm IDE, you can install the packages through the command prompt andfollow 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.

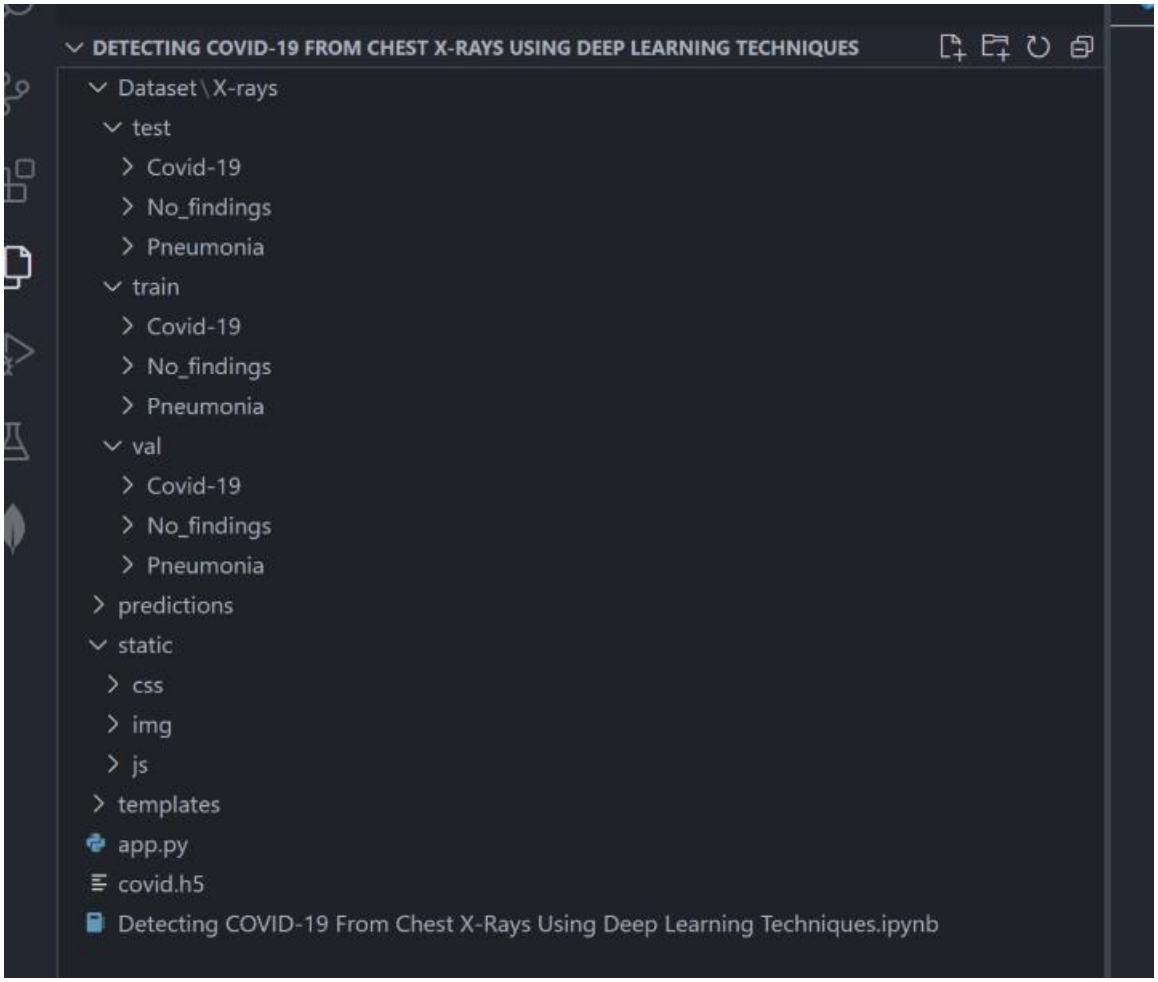
To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
  - Create Train and Test Folders.
- Data Preprocessing.
  - Import the ImageDataGenerator library
  - Configure ImageDataGenerator class
  - ApplyImageDataGenerator 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.

Milestone 1: Data Collection

Collect images of chest X-rays then organized into subdirectories based on their respective names as shown in the project structure.Create folders of Covid-19 that need to be recognized.

In this project, we have collected images of 6 types of X-rays like Covid positive, Covid negative and Pneumonia they are saved in the respective sub directories with their respectivenames.

Download the Dataset: https://www.kaggle.com/datasets/pranjallk1995/covid-xrays

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 somegeometric transformations of images like rotation, scaling, translation, etc.

Activity 1: Import the ImageDataGenerator library

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using imagedata augmentation via the ImageDataGenerator class.

Let us import the ImageDataGenerator class from tensorflow Keras

```
#import the datagenerator library
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Activity 2: Configure ImageDataGenerator class

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width_shift_range and height_shift_range arguments.
- The image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

Image Data Augmentation

```
[ ]  #image data augmentation to the training data
     train_datagen=ImageDataGenerator(rescale=1./255,
                                      shear_range=0.1,
                                      zoom_range=0.1,
                                      horizontal_flip=True)
```

```
[ ]  #image data augmentation to the testing data.
     val_datagen = ImageDataGenerator(rescale=1./ 255)
```

Activity 3:Apply ImageDataGenerator functionality to Trainset and Testset

Let us apply ImageDataGenerator functionality to Trainset and Testset by using the followingcode.For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories Covid-positive, Covid-negative and pneumonia together with labels 0 to 3 {Covid-positive: 0, Covid-negative: 1, Pneumonia: 2}

Arguments:

- directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data which is 32.
- target_size: Size to resize images after they are read from disk.
- class_mode:
  - 'int': means that the labels are encoded as integers (e.g. forsparse_categorical_crossentropy loss).
  - 'categorical' means that the labels are encoded as a categorical vector (e.g. forcategorical_crossentropy loss).
  - 'binary' means that the labels (there can be only 2) are encoded as float32 scalars withvalues 0 or 1 (e.g. for binary_crossentropy).
  - None (no labels).

Loading our data and performing data Augmentation

```
train_transform = train_datagen.flow_from_directory('/content/X-rays/train',
                                                    target_size=(128,128),
                                                    batch_size=32,
                                                    class_mode='categorical')
val_transform = val_datagen.flow_from_directory('/content/X-rays/val',
                                                target_size=(128,128),
                                                batch_size=32,
                                                class_mode='categorical')
test_transform = test_datagen.flow_from_directory('/content/X-rays/test',
                                                  target_size=(128,128),
                                                  batch_size=32,
                                                  class_mode='categorical')
```

```
Found 760 images belonging to 3 classes.
Found 95 images belonging to 3 classes.
Found 95 images belonging to 3 classes.
```

We notice that 760 images belong to 3 classes for training and 95 images belong to 3 classes
for testing purposes.

Milestone 3: Model Building
Now it's time to build our Convolutional Neural Networking which contains an input layer alongwith the convolution, max-pooling, and finally an output layer.

Activity 1: Importing the Model Building Libraries

Importing the necessary libraries

Import the Necessary Libraries

```
#to define linear initializations import Sequential
from tensorflow.keras.models import Sequential

#to add layers import Dense
from tensorflow.keras.layers import Dense

#to create a convolution kernel import Conv2D
from tensorflow.keras.layers import Conv2D

#Adding max pooling layer
from tensorflow.keras.layers import MaxPool2D

#Adding Flatten layer
from tensorflow.keras.layers import Flatten

#Adding other layers
from tensorflow.keras.layers import InputLayer
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dropout

#Adding optimizers
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import Adam
```

Activity 2: 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 3: Adding CNN Layers

- For information regarding CNN Layers refer to the linkLink: 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-poolinglayer.
- Max pool layer is used to down sample the input. (Max pooling is a pooling operation thatselects 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.

```
model.add(InputLayer(input_shape=(256, 256, 3)))

# 1st convolution layer and pooling
model.add(Conv2D(8, (3, 3), activation='relu', strides=(1, 1), padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization())

# 2nd convolution layer and pooling
model.add(Conv2D(16, (3, 3), activation='relu', strides=(1, 1), padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization())

# 3rd convolution layer and pooling
model.add(Conv2D(32, (3, 3), activation='relu', strides=(1, 1), padding='same'))
model.add(BatchNormalization())

# 4th convolution layer and pooling
model.add(Conv2D(16, (3, 3), activation='relu', strides=(1,1), padding='same'))
model.add(BatchNormalization())

# 5th convolution layer and pooling
model.add(Conv2D(32, (3, 3), activation='relu', strides=(1,1), padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization())

# 6th convolution layer and pooling
model.add(Conv2D(64, (3, 3), activation='relu', strides=(1,1), padding='same'))
model.add(BatchNormalization())

# 7th cconvolution layer and pooling
model.add(Conv2D(32, (3, 3), activation='relu', strides=(1,1), padding='same'))
model.add(BatchNormalization())
```

```
# 8th convolution layer and pooling
model.add(Conv2D(64, (3, 3), activation='relu', strides=(1,1), padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization())

# 9th convolution layer and pooling
model.add(Conv2D(128, (3, 3), activation='relu', strides=(1,1), padding='same'))
model.add(BatchNormalization())

# 10th convolution layer and pooling
model.add(Conv2D(64, (3, 3), activation='relu', strides=(1,1), padding='same'))
model.add(BatchNormalization())

# Flattening the layers
model.add(Flatten())
```

Activity 5: Adding Dense Layer

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

Adding fully connected layers

```
#Adding the 1st hidden layer
model.add(Dense(units=100, activation='relu'))
#Adding the 2nd hidden layer
model.add(Dense(units=100, activation='relu'))
#Adding the 3rd hidden layer
model.add(Dropout(0.25))
# output layer
model.add(Dense(units=3, activation='softmax'))
```

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 the model

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 8)       224

 max_pooling2d (MaxPooling2  (None, 128, 128, 8)       0
 D)

 batch_normalization (Batch  (None, 128, 128, 8)       32
 Normalization)

 conv2d_1 (Conv2D)           (None, 128, 128, 16)      1168

 max_pooling2d_1 (MaxPoolin  (None, 64, 64, 16)        0
 g2D)

 batch_normalization_1 (Bat  (None, 64, 64, 16)        64
 chNormalization)

 conv2d_2 (Conv2D)           (None, 64, 64, 32)        4640

 batch_normalization_2 (Bat  (None, 64, 64, 32)        128
 chNormalization)

 conv2d_3 (Conv2D)           (None, 64, 64, 16)        4624

 batch_normalization_3 (Bat  (None, 64, 64, 16)        64
```

```
conv2d_4 (Conv2D)           (None, 64, 64, 32)        4640

max_pooling2d_2 (MaxPoolin  (None, 32, 32, 32)        0
g2D)

batch_normalization_4 (Bat  (None, 32, 32, 32)        128
chNormalization)

conv2d_5 (Conv2D)           (None, 32, 32, 64)        18496

batch_normalization_5 (Bat  (None, 32, 32, 64)        256
chNormalization)

conv2d_6 (Conv2D)           (None, 32, 32, 32)        18464

batch_normalization_6 (Bat  (None, 32, 32, 32)        128
chNormalization)

conv2d_7 (Conv2D)           (None, 32, 32, 64)        18496

max_pooling2d_3 (MaxPoolin  (None, 16, 16, 64)        0
g2D)

batch_normalization_7 (Bat  (None, 16, 16, 64)        256
chNormalization)

conv2d_8 (Conv2D)           (None, 16, 16, 128)       73856

batch_normalization_8 (Bat  (None, 16, 16, 128)       512
chNormalization)

conv2d_9 (Conv2D)           (None, 16, 16, 64)        73792

batch_normalization_9 (Bat  (None, 16, 16, 64)        256

flatten (Flatten)           (None, 16384)             0

dense (Dense)               (None, 100)               1638500

dense_1 (Dense)             (None, 100)               10100

dropout (Dropout)           (None, 100)               0

dense_2 (Dense)             (None, 3)                 303

=================================================================
Total params: 1869127 (7.13 MB)
Trainable params: 1868215 (7.13 MB)
Non-trainable params: 912 (3.56 KB)
```

Activity 6: 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

compliling the model

```
#compiling the model
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

Activity 7: 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 30 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 evaluatethe loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argumentcan 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 pointsin your dataset divided by the validation batch size.

Fit the model

```
#fit on data for 20 epochs
history = model.fit_generator(train_transform, epochs=20, validation_data=val_transform)
```

```
<ipython-input-18-3db5584d355b>:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which
  history = model.fit_generator(train_transform, epochs=20, validation_data=val_transform)
Epoch 1/20
24/24 [==============================] - 66s 3s/step - loss: 0.8833 - accuracy: 0.5829 - val_loss: 1.1047 - val_accuracy: 0.2632
Epoch 2/20
24/24 [==============================] - 61s 3s/step - loss: 0.5071 - accuracy: 0.7895 - val_loss: 1.1082 - val_accuracy: 0.2632
Epoch 3/20
24/24 [==============================] - 61s 3s/step - loss: 0.4661 - accuracy: 0.7934 - val_loss: 1.1492 - val_accuracy: 0.2632
Epoch 4/20
24/24 [==============================] - 60s 2s/step - loss: 0.2813 - accuracy: 0.9026 - val_loss: 1.1600 - val_accuracy: 0.4421
Epoch 5/20
24/24 [==============================] - 61s 3s/step - loss: 0.1844 - accuracy: 0.9303 - val_loss: 1.1801 - val_accuracy: 0.4316
Epoch 6/20
24/24 [==============================] - 62s 2s/step - loss: 0.1420 - accuracy: 0.9461 - val_loss: 1.1120 - val_accuracy: 0.4737
Epoch 7/20
24/24 [==============================] - 61s 3s/step - loss: 0.1228 - accuracy: 0.9645 - val_loss: 1.0564 - val_accuracy: 0.5474
Epoch 8/20
24/24 [==============================] - 62s 2s/step - loss: 0.0844 - accuracy: 0.9711 - val_loss: 1.0463 - val_accuracy: 0.5368
Epoch 9/20
24/24 [==============================] - 60s 2s/step - loss: 0.0637 - accuracy: 0.9816 - val_loss: 0.8587 - val_accuracy: 0.6105
Epoch 10/20
24/24 [==============================] - 60s 2s/step - loss: 0.0421 - accuracy: 0.9908 - val_loss: 0.7741 - val_accuracy: 0.6737
Epoch 11/20
24/24 [==============================] - 61s 2s/step - loss: 0.0227 - accuracy: 0.9974 - val_loss: 0.7186 - val_accuracy: 0.6947
Epoch 12/20
24/24 [==============================] - 59s 2s/step - loss: 0.0215 - accuracy: 0.9974 - val_loss: 0.6907 - val_accuracy: 0.7053
Epoch 13/20
24/24 [==============================] - 61s 3s/step - loss: 0.0222 - accuracy: 0.9961 - val_loss: 0.6616 - val_accuracy: 0.7263
Epoch 14/20
24/24 [==============================] - 62s 3s/step - loss: 0.0273 - accuracy: 0.9934 - val_loss: 0.5729 - val_accuracy: 0.7684
Epoch 15/20
24/24 [==============================] - 60s 2s/step - loss: 0.0159 - accuracy: 0.9974 - val_loss: 0.5758 - val_accuracy: 0.7579
Epoch 16/20
24/24 [==============================] - 58s 2s/step - loss: 0.0194 - accuracy: 0.9974 - val_loss: 0.6260 - val_accuracy: 0.7474
Epoch 17/20
24/24 [==============================] - 61s 2s/step - loss: 0.0100 - accuracy: 1.0000 - val_loss: 0.5305 - val_accuracy: 0.8211
Epoch 18/20
24/24 [==============================] - 61s 3s/step - loss: 0.0118 - accuracy: 0.9987 - val_loss: 0.4964 - val_accuracy: 0.8000
Epoch 19/20
24/24 [==============================] - 59s 2s/step - loss: 0.0076 - accuracy: 1.0000 - val_loss: 0.5146 - val_accuracy: 0.8105
Epoch 20/20
24/24 [==============================] - 61s 3s/step - loss: 0.0053 - accuracy: 1.0000 - val_loss: 0.4713 - val_accuracy: 0.8526
```

Activity 8: 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

```
#save the model
model.save('covid.h5')
```

Activity 9: Test The model

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

```
#import numpy library
import numpy as np

#import load_model method to load our saved model
from keras.models import load_model

#import image from keras.preprocessing
from keras.preprocessing import image

#loading our saved model file
model= load_model("covid.h5")
```

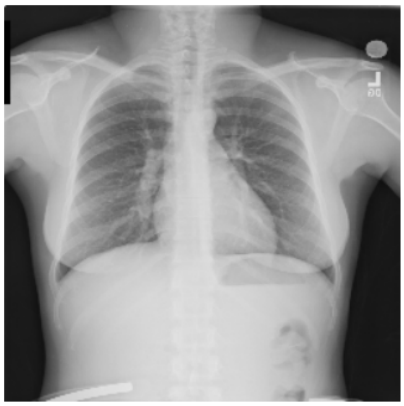
Taking an image as input and checking the results

```
img = image.load_img('/content/X-rays/test/No_findings/00002007_000.png',target_size=(256,256))

#converting in to array format
x=image.img_to_array(img)

#changing its dimensions as per our requirement
x=np.expand_dims(x,axis=0)

#printing the image
img
```



```
pred =np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 172ms/step
```

```
[214] index=['0','1','2']
      result=str(index[pred[0]])
      result
```

```
'1'
```

```
[209] train_transform.class_indices
```

```
{'Covid-19': 0, 'No_findings': 1, 'Pneumonia': 2}
```

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

```
index1=['Covid-19','No_findings','Pneumonia']
result1=str(index1[pred[0]])
result1
```

```
'No_findings'
```

The predicted class index name will be printed here.

Milestone 4: Application Building

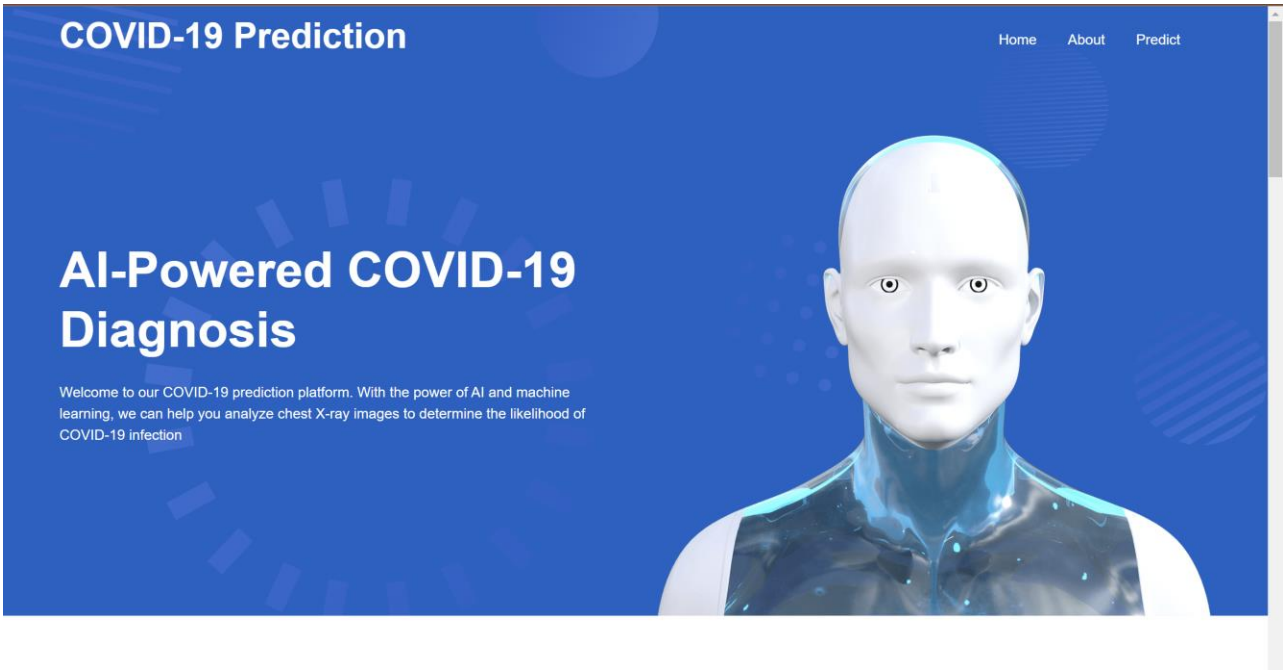
Now that we have trained our model, let us build our flask application which will be running inour local browser with a user interface.

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

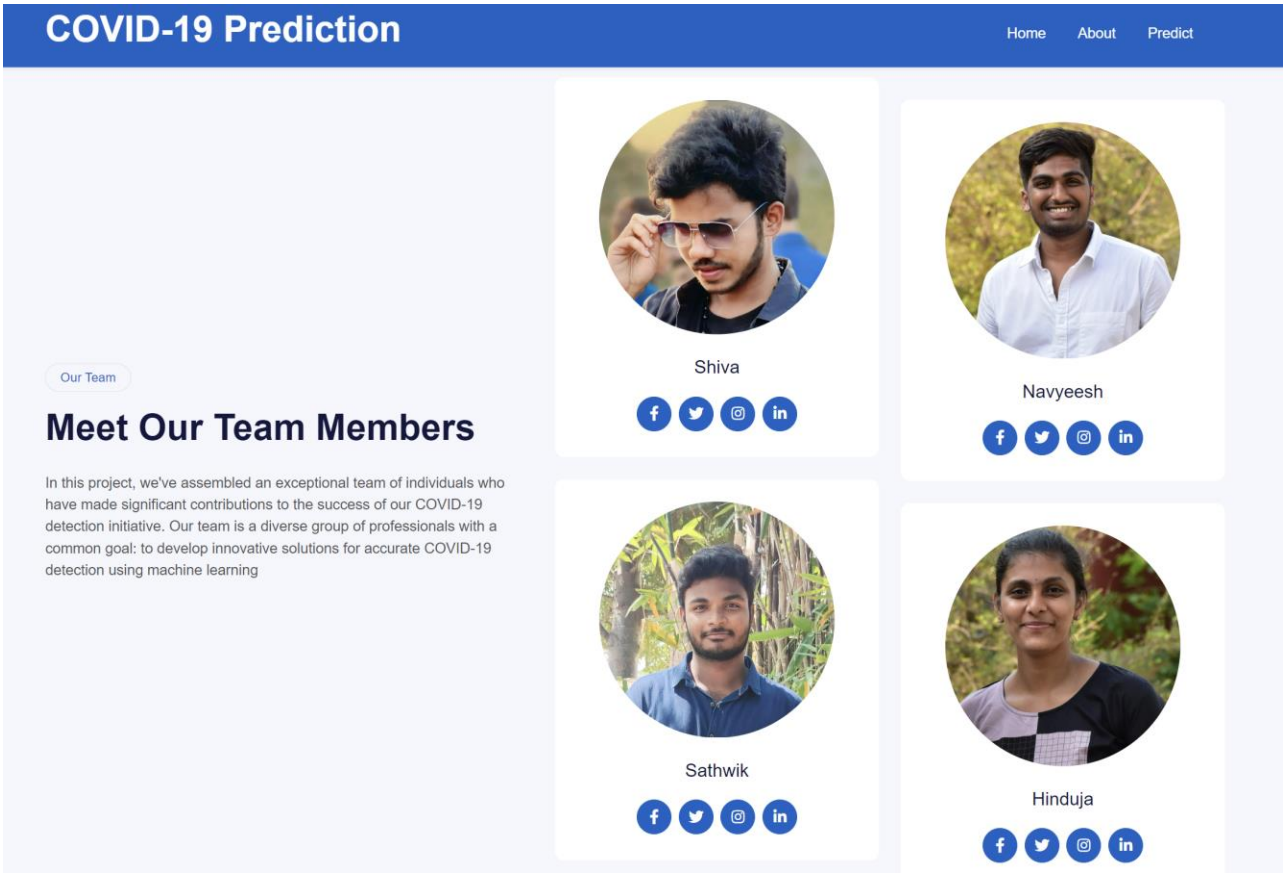
- We use HTML to create the front end part of the web page.
- Here, we have created 3 HTML pages- home.html, intro.html, and upload.html
- home.html displays the home page.
- Intro.html displays an introduction about the project
- upload.html gives the emergency alert For more information https://www.w3schools.com/html/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

index.html looks like this

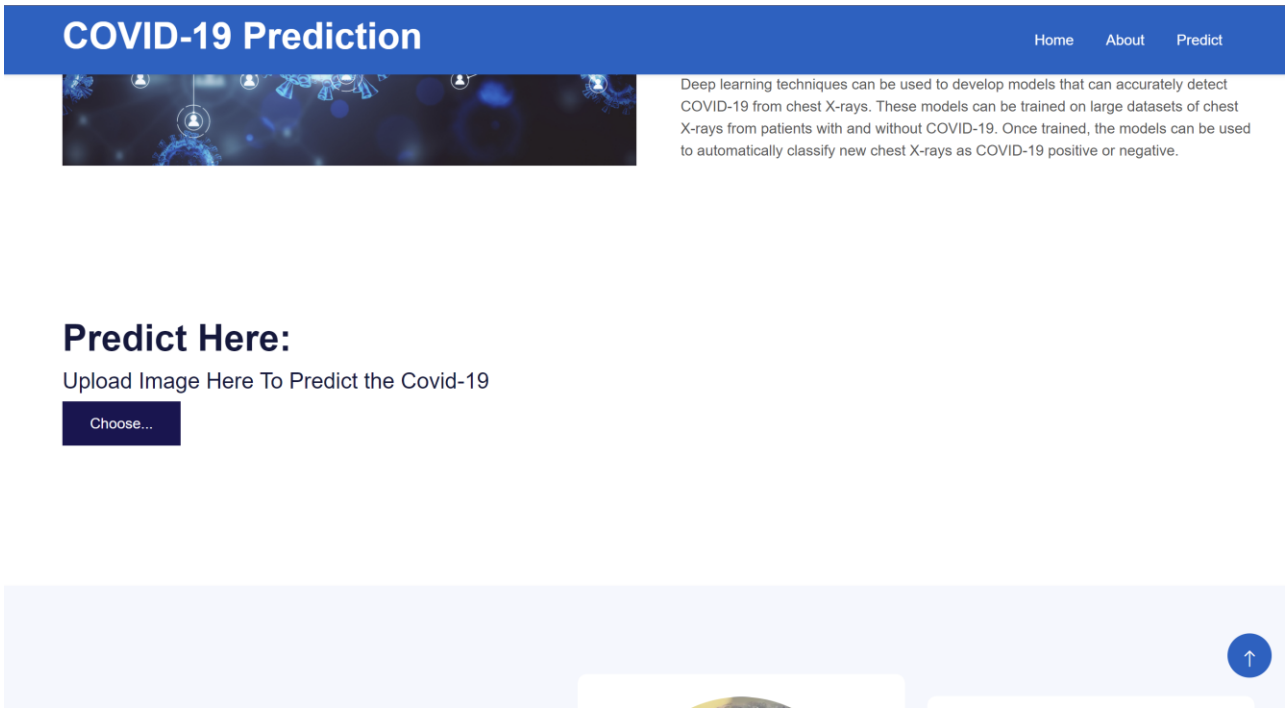


About Section:

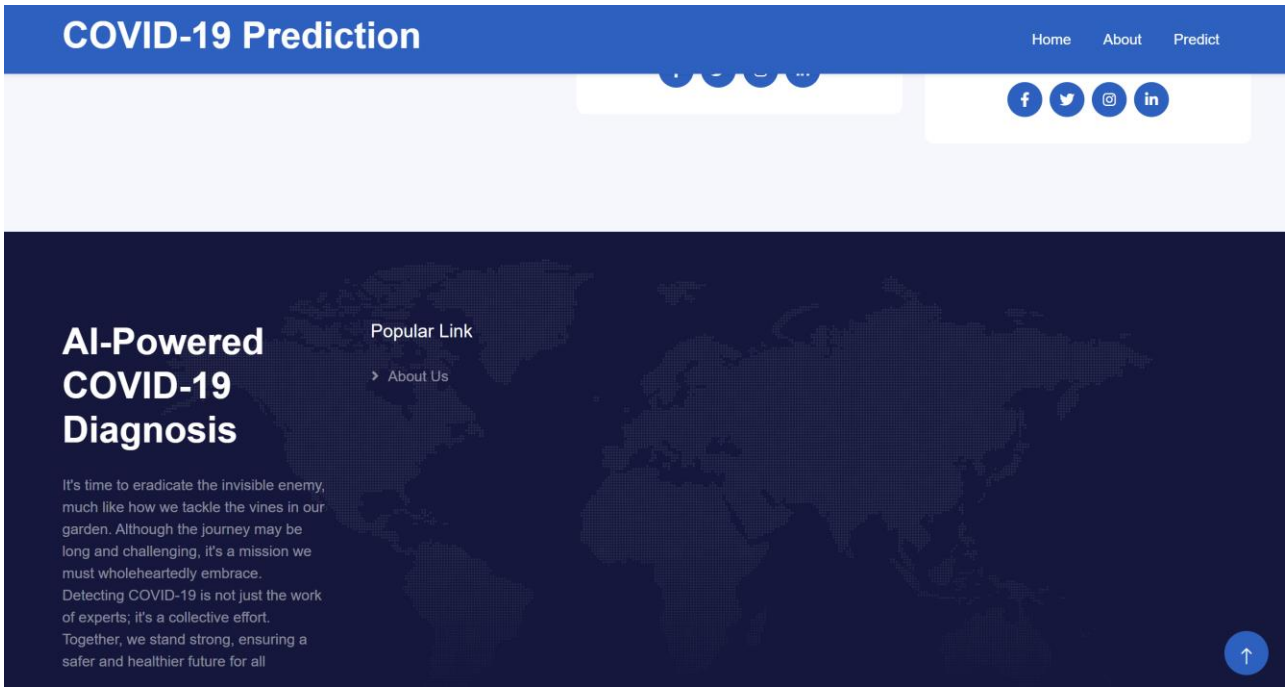
Contact Us:



Predict section:



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 > upload

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask,render_template,request
import os
import numpy as np
```

Importing the flask module in the project is mandatory. An object of the Flask class is our WSGIapplication. 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

```
app = Flask(__name__)
model = load_model(r"covid.h5",compile = False)
@app.route('/')
def index():
    return render_template("index.html")
```

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 ofa 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.

```
app = Flask(__name__)
model = load_model(r"covid.h5",compile = False)
@app.route('/')
def index():
    return render_template("index.html")

@app.route('/predict',methods = ['GET','POST'])
def upload():
    
    return text
if __name__=='__main__':
```

Showcasing prediction on UI:

```
@app.route('/predict',methods = ['GET','POST'])
def upload():
    if request.method=='POST':
        f = request.files['images']
        basepath=os.path.dirname(__file__)
        filepath = os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)

        img = image.load_img(filepath,target_size =(256,256))
        x = image.img_to_array(img)
        x = np.expand_dims(x,axis = 0)
        pred =np.argmax(model.predict(x),axis=1)
        index=['Based on the analysis of your chest X-ray image, our model has determined the likelihood of COVID-19 i
        text=""+str(index[pred[0]])
    return text
if __name__=='__main__':
    app.run(debug=True)
```

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 directoryusing 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 numericalvalue of a class (like 0,1 ,2 etc.) which lies in the 0th index of the variable preds. This numericalvalue is passed to the index variable declared. 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 functionand the result is stored in the result variable.

```
[224]  pred =np.argmax(model.predict(x),axis=1)

...    1/1 [==============================] - 0s 47ms/step

[225]  index=['0','1','2']
       result=str(index[pred[0]])
       result

...    '1'

[226]  train_transform.class_indices

...    {'Covid-19': 0, 'No_findings': 1, 'Pneumonia': 2}
```

Final Run the application

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

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

Activity 3: Run the application

- Open the anaconda 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 towhere you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.

```
PS C:\ai_ml> python app.py
```

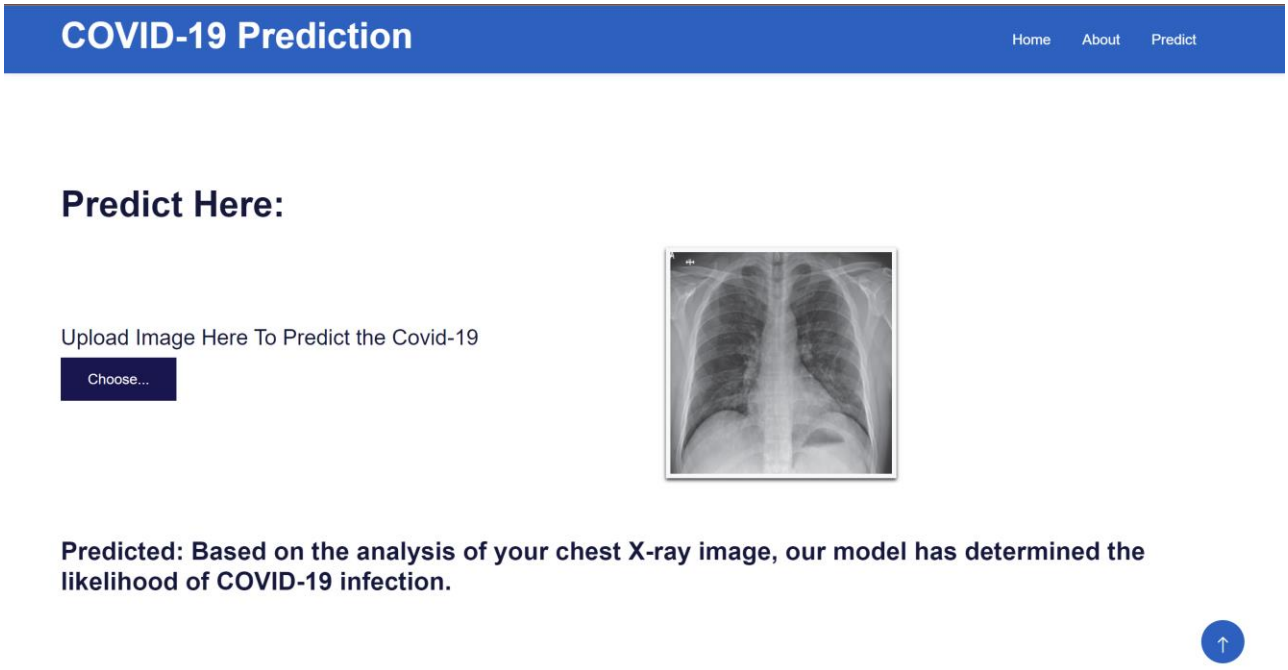
Then it will run on localhost:5000

```
PS C:\ai_ml> python app.py
2023-11-01 20:08:09.963680: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in p
formance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compi
r flags.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
2023-11-01 20:08:15.948972: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in p
formance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compi
r flags.
 * Debugger is active!
 * Debugger PIN: 144-408-769
```

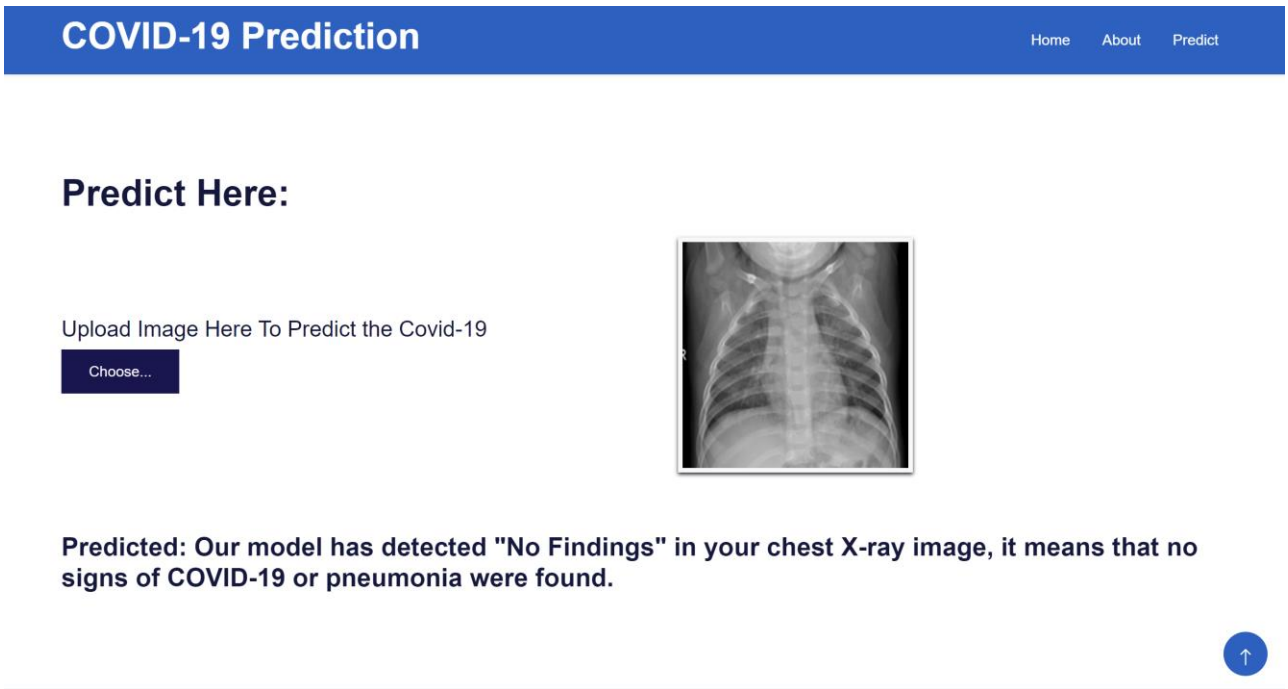
Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.

FINAL OUTPUTS:

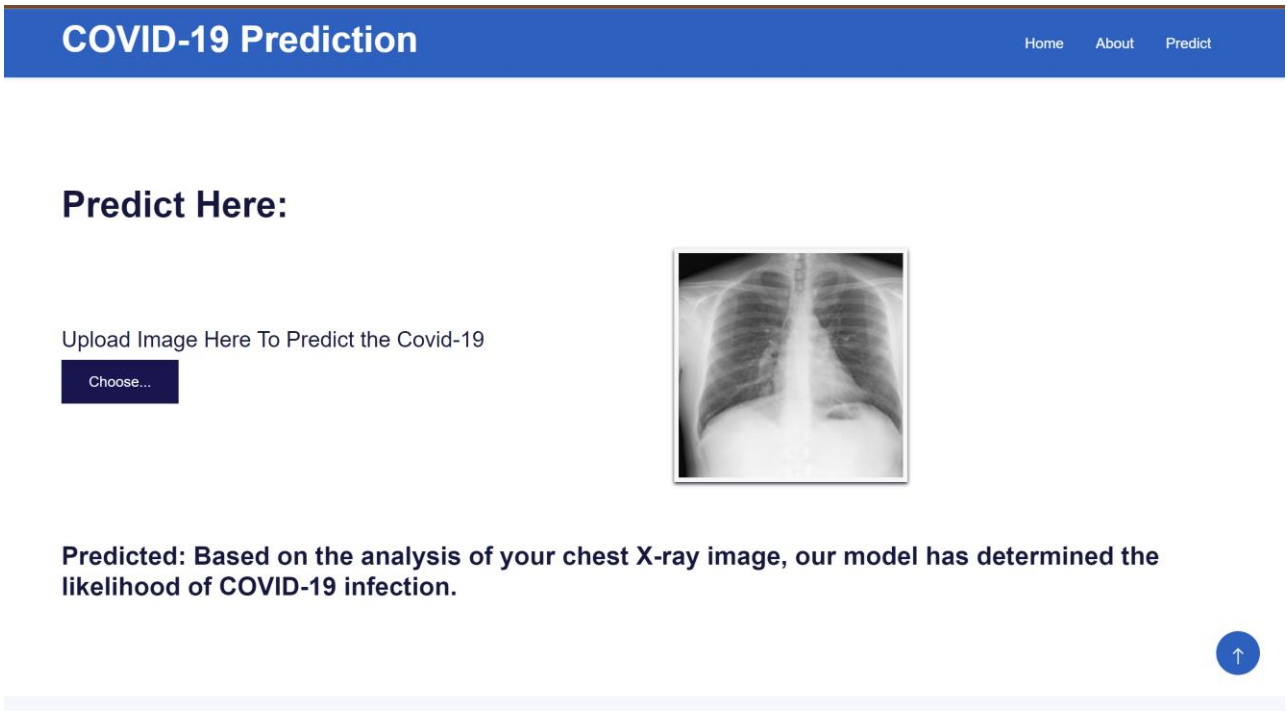
Output-1:



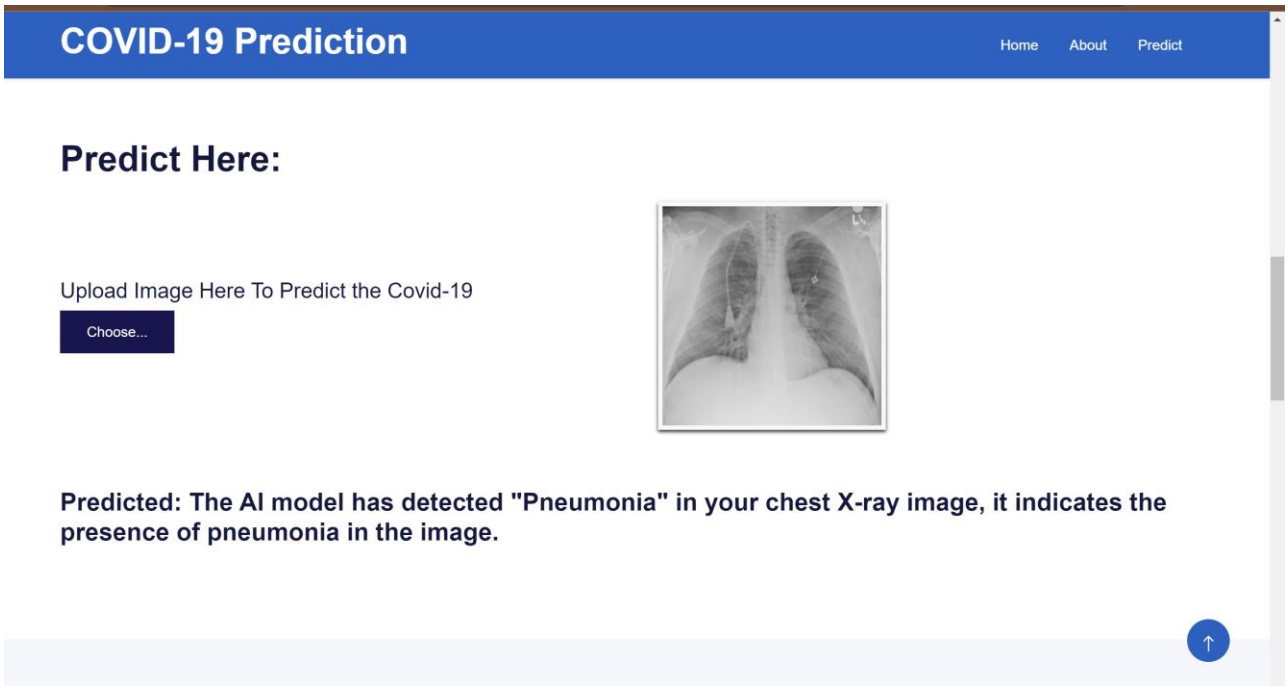
Output-2:



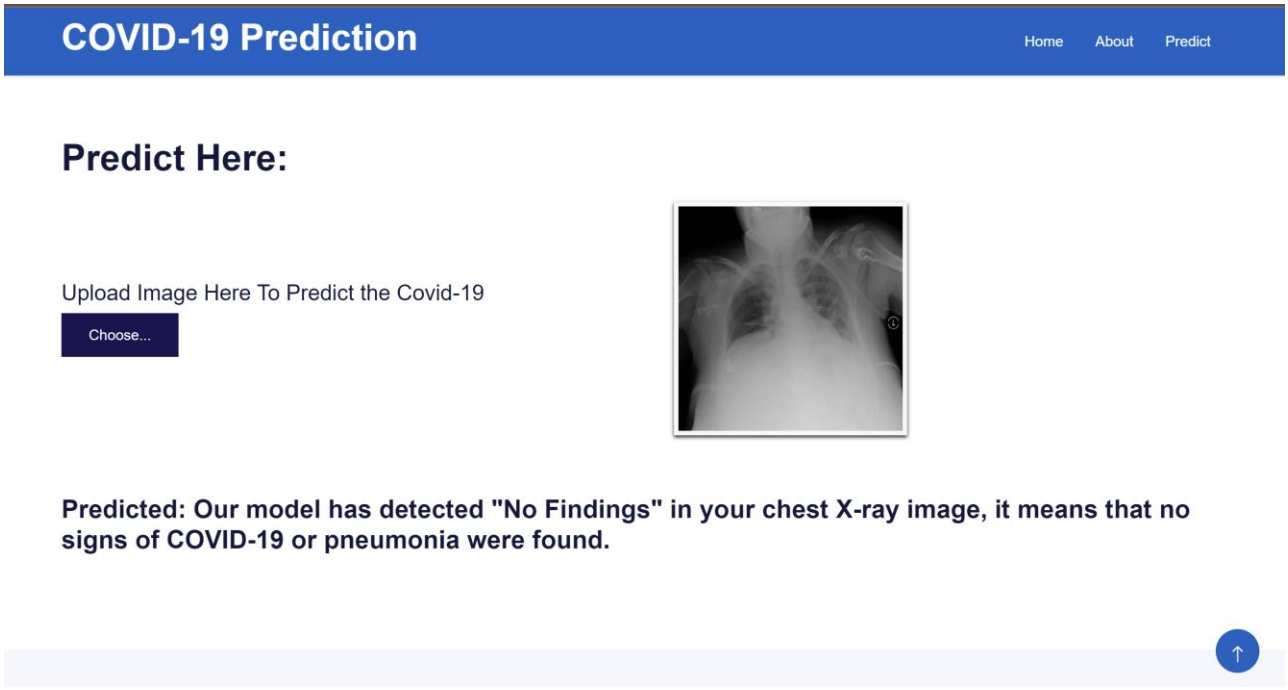
Output-3:



Output-4:



Output-5:



Output-6: