

Covid-19 Detection Using Chest X-Rays



Team Incognito

Harika Reddy Vundyala-16327664

Nishitha Reddy Dyasani-12588531

Abstract:

According to the World Health Organization, the coronavirus epidemic threatens the world's health system every day. Disease detection is an important factor in preventing the epidemic. Whether the person has a virus or not is usually done by the PCR test. In addition to the PCR method, chest x-ray images can be classified with deep learning methods. This popularity reflected positively on limited health datasets. In this study, it was aimed to detect the disease of people whose x-rays were taken for suspected COVID-19. The data set includes chest x-rays of patients with COVID-19, viral pneumonia, and healthy patients. These three groups have been classified through multi-class classification deep learning models.

Introduction:

In this study, it is aimed to Classify Healthy patients, COVID-19, and viral pneumonia cases. Deep learning allows us to train artificial intelligence to predict outputs with a given data set. Also it uses many nonlinear layers for feature extraction and feature modification. The data set used in the study was created by a few researchers from Qatar and Dhaka universities. The data set includes COVID-19 positive cases, healthy patients, and chest radiographs of viral pneumonia patients .In this study, it is aimed to solve the multi-class classification problem.

Objectives:

- In this project, We are building a Convolutional Neural Network model for this dataset where it detects the particular image type.
- In the real-time world, it helps people to get their covid test results in the quickest way possible.
- This application takes the image data as an input and returns its label(covid,normal,viral pneumonia)

Scope:

- This model helps people to get their covid results ahead of time with their chest X-Rays.
- In future studies, the success ratio can be increased by strengthening the data set.
- Lung tomography can be used in addition to chest radiographs. By developing different deep learning models, success ratio and performance can be increased.

Problem Statement:

Currently, many countries across the world are suffering from COVID-19. Existing diagnosis procedures to identify disease are time consuming. Therefore, to overcome this we proposed a CNN model, which can efficiently classify the COVID-19 cases well advanced in time. So, the task here is to classify X-rays of people as COVID-19 positive, normal and viral pneumonia by using a suitable deep learning model called CNN.

Related Work:

- There are various approaches to this problem which are using linear regression, least absolute shrinkage in order to detect the covid.
- There are other approaches which are using Support Vector Machine but not with a good accuracy rate, instead exponential smoothing gave the best result.

Methodology:

The proposed system aims to predict Covid-19 from chest X-ray images using Deep Learning and Convolutional Neural Networks. The available dataset was collected from Kaggle which includes normal, pneumonia, and covid chest x-ray images.

We have designed and developed a web application where the user can upload an image of a chest X-Ray and click on the predict button which will classify the image from a group of 3 different images and display the output.

For developing the Covid-19 detection using chest X-rays application, we have used CNN model for classification.

Technologies Used

- Python
- Flask
- Convolution Neural Network

Design of Interface:

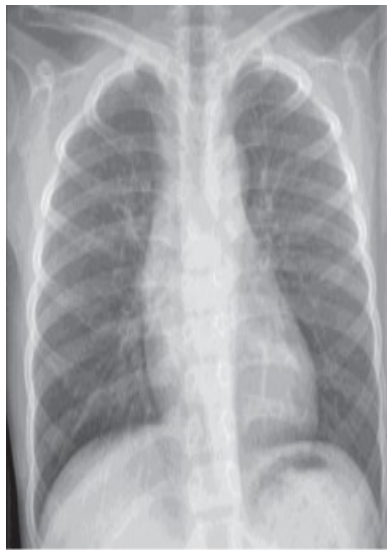
This proposed work's most significant benefit is that it is very user-friendly, and the system benefits all sections of societies. Everyone can use it if anyone has a minimum knowledge of browsing and selecting images. This whole system is mainly done in the Python language, we trained and validated our data in Google Colab. We use Flask, which helps us to deploy the model. For the CNN base model, we use Keras for image processing, we also use Tensorflow, which resize images and zoom. We used scikit-learn to maintain our algorithm-like epochs. To highlight training loss, training accuracy, validation accuracy, and validation loss, we use Matplotlib, from which the library was born, as well as Sklearn. In the backend part, deep learning is used.

Model Building:

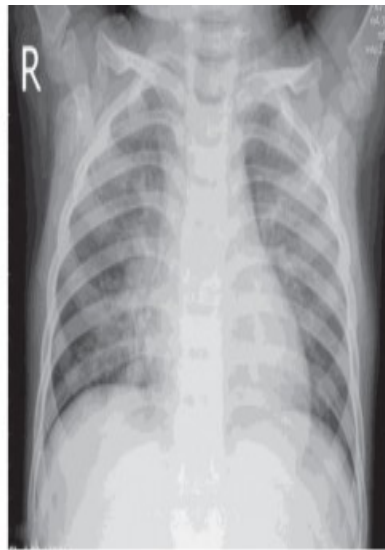
Dataset:

The dataset we are using in our project is Covid-19 Detection from kaggle.

- The data set includes chest X-ray images of 219 COVID-19 patients, 219 healthy patients and 219 viral pneumonia patients. This means a total of 657 images.
- The dataset has 3 classes which include Normal, Pneumonia and Covid-19 images.
- This is used in multi-class classification problems.



(a) Normal



(b) Pneumonia



(c) COVID-19

Preprocessing:

Preprocessing data is a common first step in the deep learning workflow to prepare raw data in a format that the network can accept. We have normalized the data and also performed image augmentation as we have a small dataset.

Data Normalization: Normalization gives equal weights/importance to each variable so that no single variable steers model performance in one direction just because they are bigger numbers. Here we have rescaled the data.

Image Augmentation: Image augmentation is a very powerful technique used to artificially create variations in existing images to expand an existing image data set. This creates new and different images from the existing image data set that represents a comprehensive set of possible images. This is done by applying different transformation techniques like zooming the existing image, rotating the existing image by a few degrees, shearing or cropping the existing set of images etc. **This helps to increase the performance of the model by generalizing better and thereby reducing overfitting.**

CNN Model:

In our Covid-19 Detection using chest X-Rays, we have run the CNN model for better accuracy results. CNNs are structures designed to take images as input and are used effectively in computer vision. CNN consists of one or more convolutional layers and one or more fully connected layers, Convolution, Relu, Pooling, Flattening, and Fully Connected layers.

We created a **Sequential CNN model** to classify the images. Since the sequential model can be accurate for 80% of the use cases we have trained this model on our dataset. The model built consists of **three convolution blocks** with a **max pool layer** in each of them. There's a **fully connected layer** with 128 units on top of it that is activated by the relu activation function. These layers are stacked to the Sequential model.

Methods and Libraries used in CNN Model:

Sequential() - We have to create a Sequential model and stack the layers to the model

compile() - We have used a compile method to create learning for the model with loss function as **spasrecategorical_crossentropy** and optimizer as **adam** and metrics as accuracy.

summary() - gives a useful summary of the model, which includes: Name and type of all layers in the model, output shape for each layer, number of weight parameters of each layer.

fit() - We have used the fit method to train the model

predict() - We have used this method to predict the result for test data.

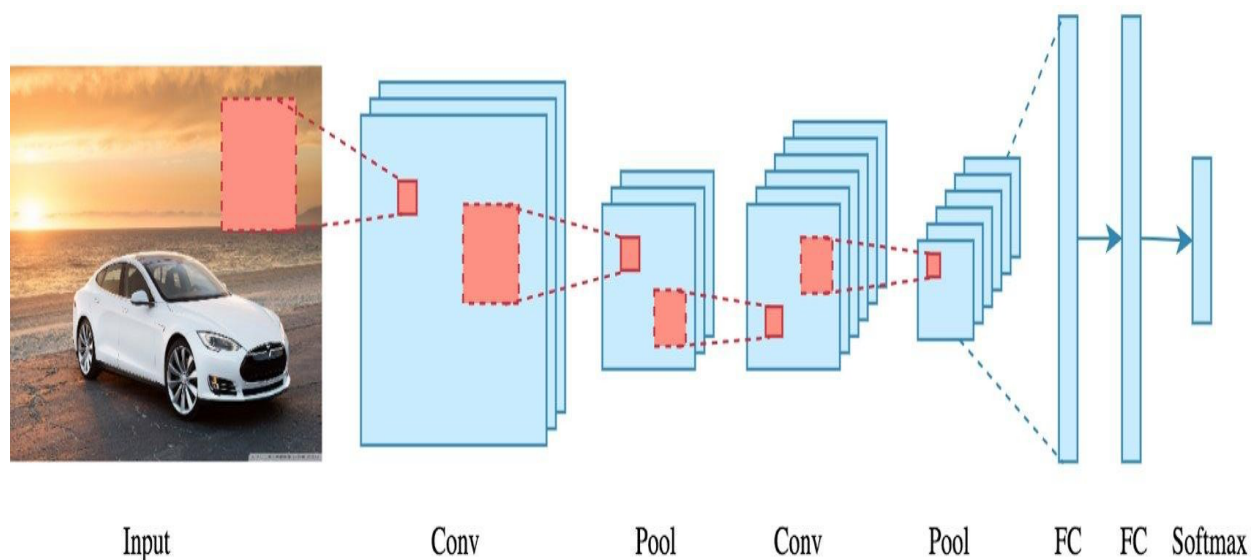


Figure: Architecture of CNN Model

Model: "sequential_2"

| Layer (type) | Output Shape | Param # |
|--------------------------------|----------------------|---------|
| sequential_1 (Sequential) | (None, 180, 180, 3) | 0 |
| rescaling_1 (Rescaling) | (None, 180, 180, 3) | 0 |
| conv2d_3 (Conv2D) | (None, 180, 180, 16) | 448 |
| max_pooling2d_3 (MaxPooling2D) | (None, 90, 90, 16) | 0 |
| conv2d_4 (Conv2D) | (None, 90, 90, 32) | 4640 |
| max_pooling2d_4 (MaxPooling2D) | (None, 45, 45, 32) | 0 |
| conv2d_5 (Conv2D) | (None, 45, 45, 64) | 18496 |
| max_pooling2d_5 (MaxPooling2D) | (None, 22, 22, 64) | 0 |
| dropout (Dropout) | (None, 22, 22, 64) | 0 |
| flatten_1 (Flatten) | (None, 30976) | 0 |
| dense_2 (Dense) | (None, 128) | 3965056 |
| dense_3 (Dense) | (None, 5) | 645 |
| Total params: 3,989,285 | | |
| Trainable params: 3,989,285 | | |
| Non-trainable params: 0 | | |

Table 1: Architecture of the model used for deep feature extraction

Accuracy:

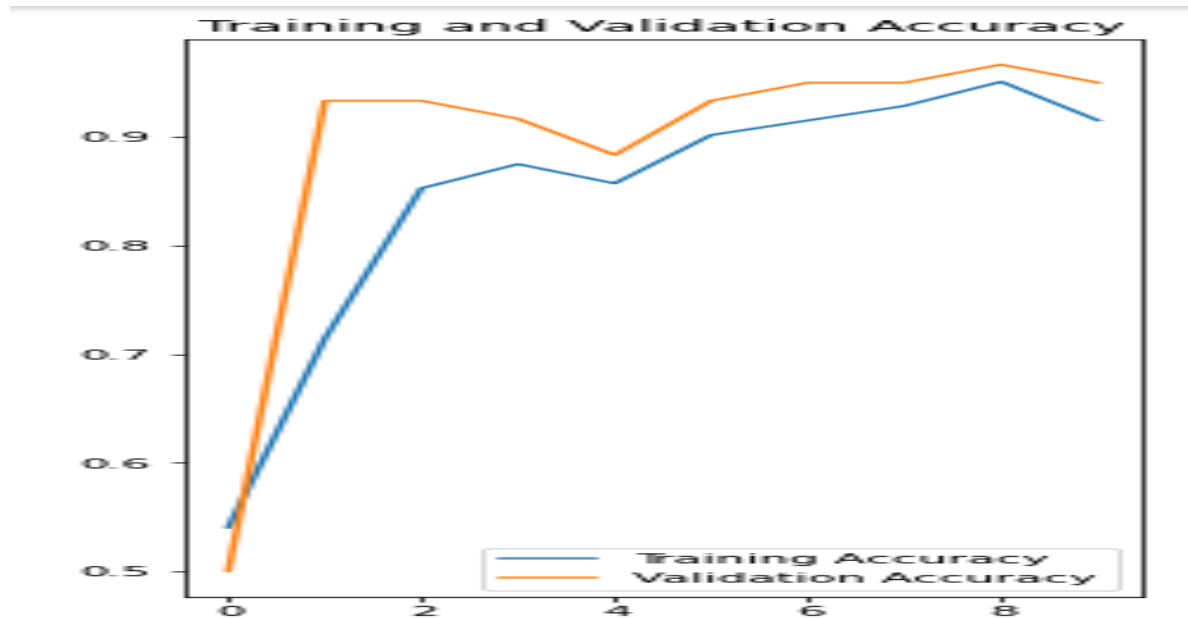


Figure : Training and Validation Accuracy

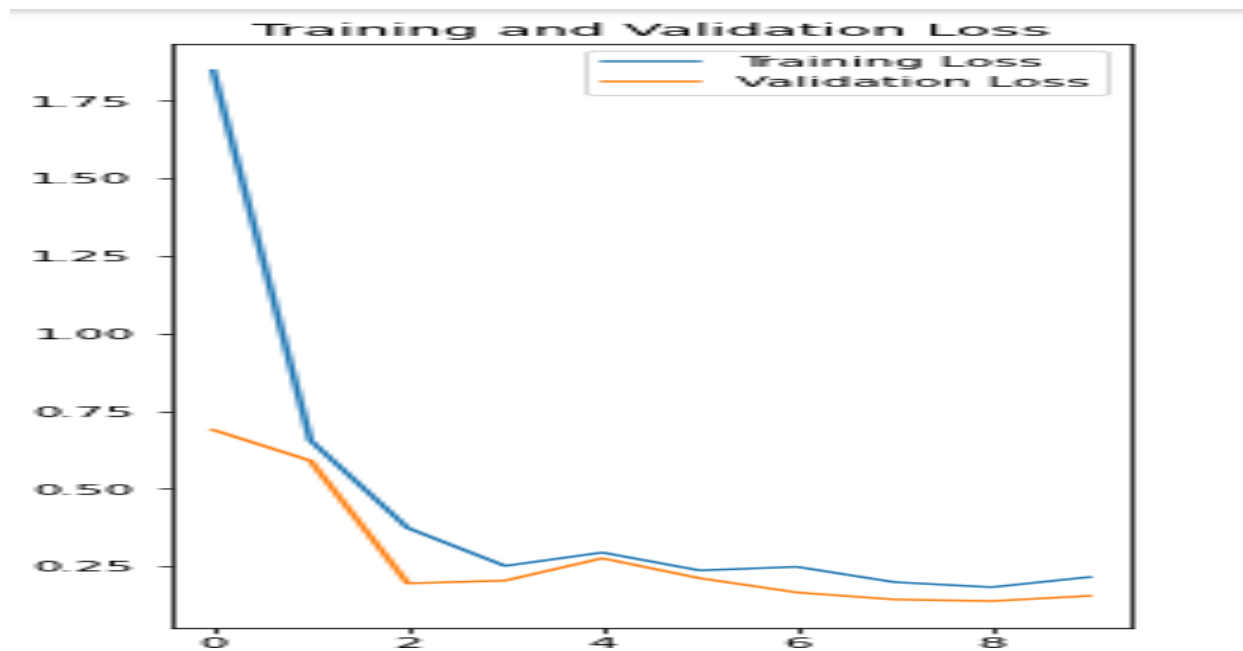


Figure: Training and Validation Loss

Frontend Tool

Application:

We have used **Flask** for developing the web application for our Covid-19 detection. In our web application, the user has to upload an image of the chest X-Ray for the classification of the image.

Firstly, the user has to upload an image and when clicked on the **Predict Image** button, the result will be displayed ahead of time. The result is displayed based on the classification whether the user is infected to Covid or not.

Flask environment:

For building the web application for our model, we have used Flask with python. Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier.

In the **Visual Studio terminal**, we have to install the flask for developing web design for our model which is in the python language. We have created a **virtual environment** in our workspace. Activated the environment and installed the flask using pip.

Command used for installing the flask: **pip install flask**

For our Flask web application, we made a separate directory called DEMO FLASK and created the following:

- 1.A '**templates**' folder, which contains the **home.html** file
- 2.A '**model.h5**' folder, which contains our **trained CNN model**.
- 3.An '**app.py**'(python) file which contains the flask code for our web app.

app.py file:

Here, we have created a flask app. For that, we need to load the **model.h5** file which is our deep learning trained model. We have used the routing from the **index.html** to this python file for the loading of the web page in the application. For the web application to load to the landing page, we have used the **GET** routing to the index page. So when the application is loaded, the index.html page is viewed. We use **render_template** library to grab the HTML file and render that as our webpage.

We are using routing for form handling on the index page. We have used **GET, POST** routing for the form where we get the results of the predicted Image. For each action in the form, we need to get the image file user selected by using the request library and reading the image.

For the Classify of an image, we have used the **POST** routing for the form and getting the image user selected. When the Predict Image button is clicked, it checks the label of the image and gives the output in the text format by reloading the webpage using the **render_template** library.

- **Load the image selected by the user using the request library**
- **Predict the label of the image**

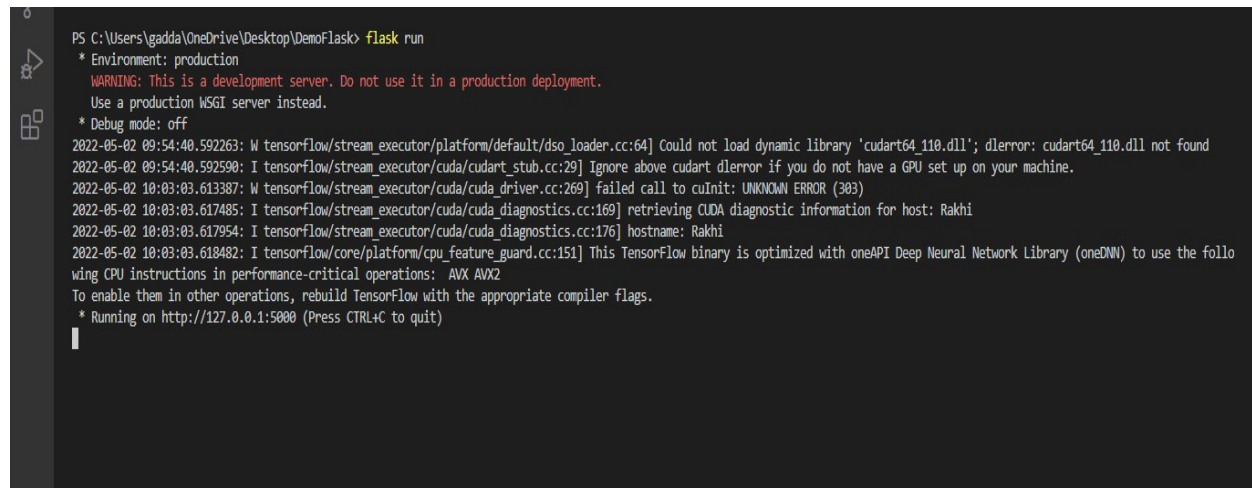
Index.html file:

In this file, we have designed the GUI of the application. We have used **bootstrap** for the **CSS** styling and font of the text on the web page. In the scripting, we have defined the function for the load image when the uploading of an image is done by the user.

We have designed the form for the web page. This form contains the choose file, classify, play and Translate buttons.

Run a Flask app

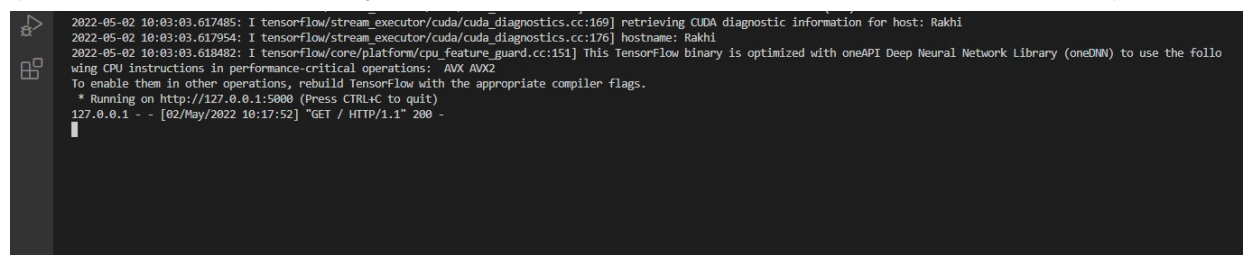
In the Visual Studio Terminal, we run the app by entering flask run, which runs the development server. The development server looks for app.py by default. When we run flask, we get the following output.



```
PS C:\Users\gadda\OneDrive\Desktop\DemoFlask> flask run
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
2022-05-02 09:54:40.592263: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-05-02 09:54:40.592590: I tensorflow/stream_executor/cuda/cuda_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-05-02 10:03:03.613387: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-05-02 10:03:03.617485: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: Rakhi
2022-05-02 10:03:03.617954: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: Rakhi
2022-05-02 10:03:03.618482: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

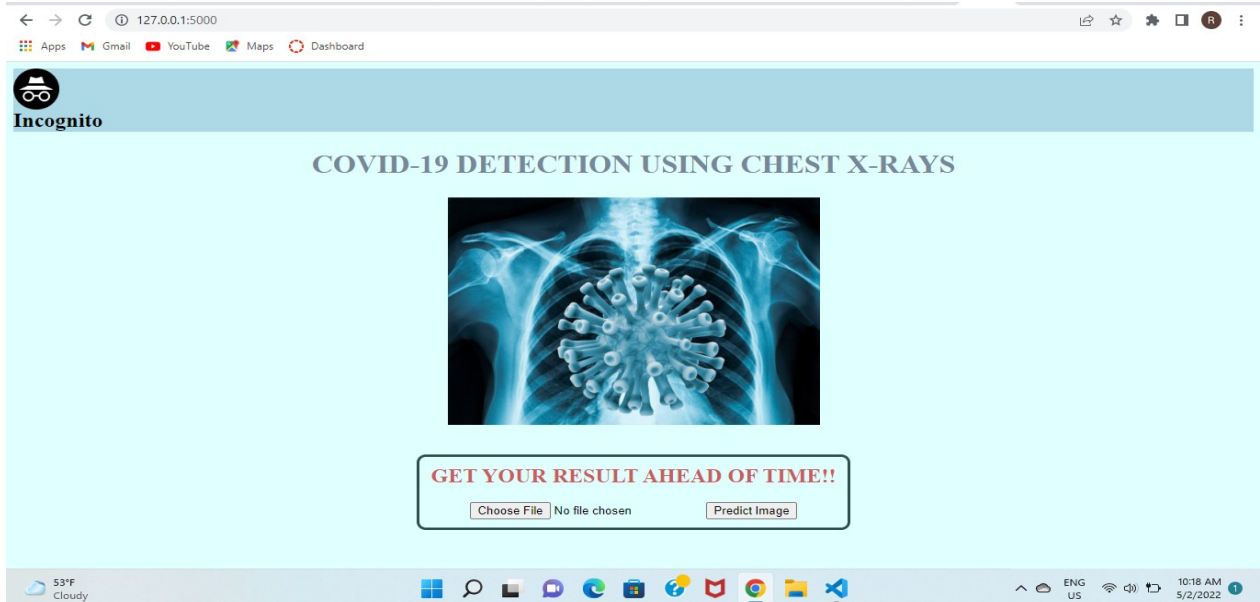
To open the default browser to the rendered page, paste the <http://127.0.0.1:5000/> URL in the browser which is shown at the end of the line in the above figure. Now, the below screenshot will be displayed which is GUI covid-19 prediction desktop tool.

Also, when we visit the URL like /, a message appears in the debug terminal showing the HTTP request which is shown in the below figure.
(127.0.0.1 - - [02/May/2022 10:17:52] "GET / HELP/1.1" 200-)

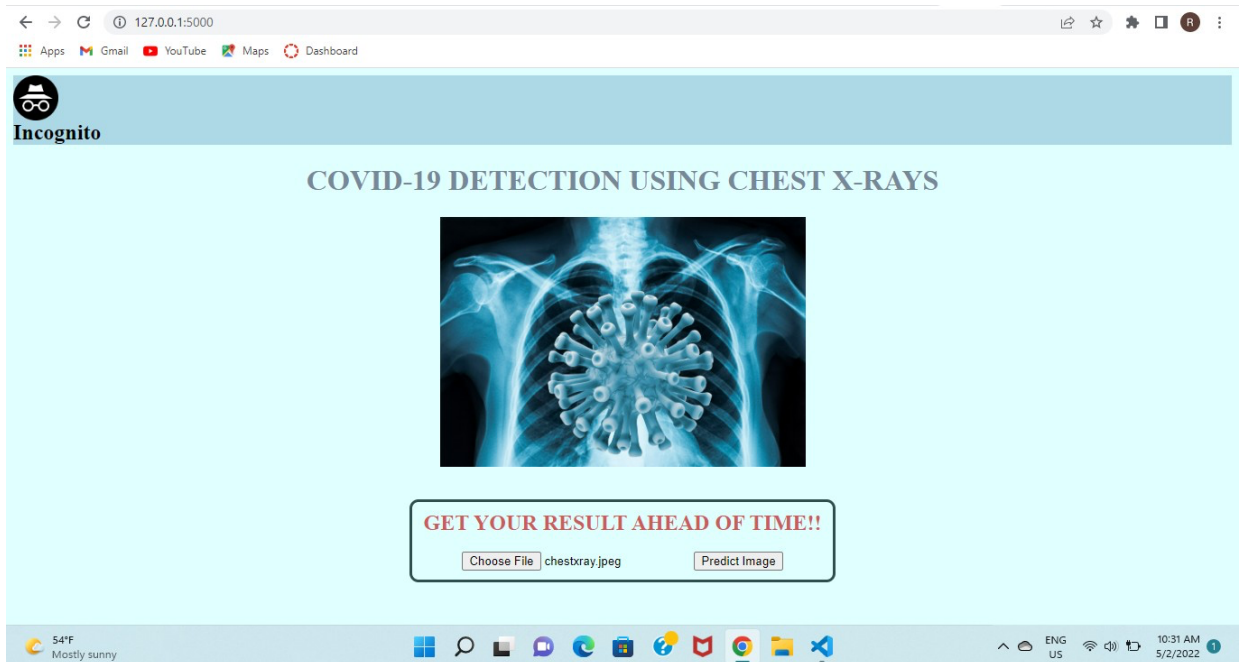


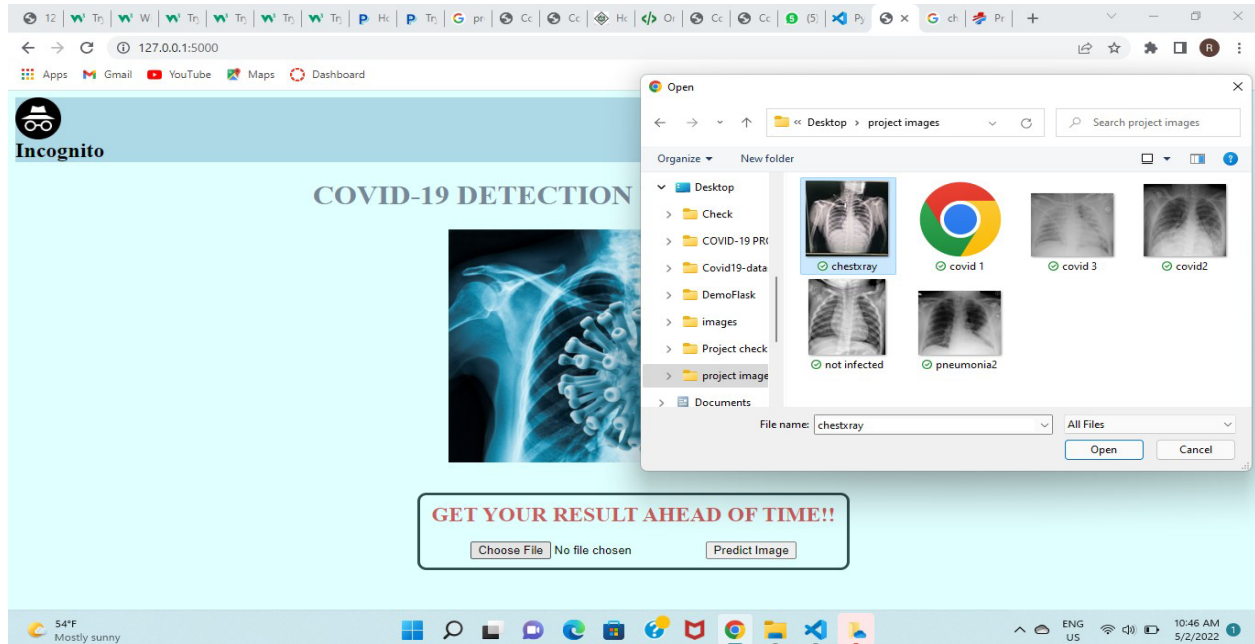
```
2022-05-02 10:03:03.617485: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: Rakhi
2022-05-02 10:03:03.617954: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: Rakhi
2022-05-02 10:03:03.618482: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
127.0.0.1 - - [02/May/2022 10:17:52] "GET / HELP/1.1" 200 -
```

When we paste the <http://127.0.0.1:5000/> URL in the browser, we get the below figure displayed.

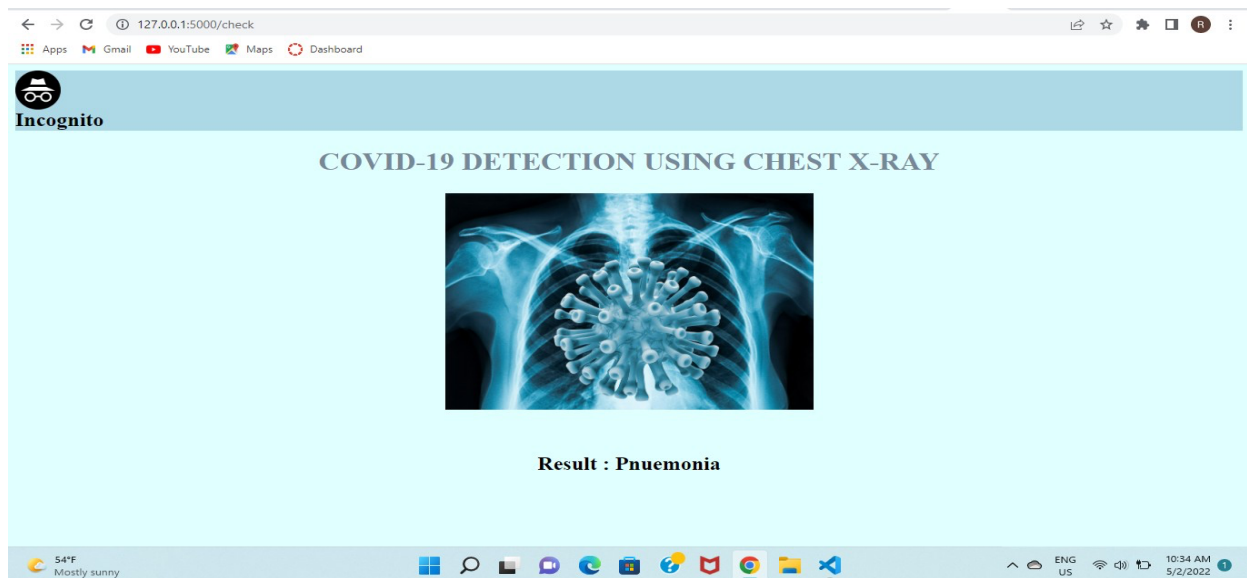


After the selection of images under the **Choose File** button, click on the **Predict Image** to get the results.





When we click on the Predict Image button, the page will be redirected to <http://127.0.0.1:5000/check> page where we get the results of our prediction.



Conclusion:

This project is all about how the system can know their Covid results as early as possible at a low cost. The main motive is to decrease the cost of the corona test and get the results as early as possible using neural networks and artificial intelligence. The project is done to detect chest X-ray images using detection. This will help the patient to get the Covid result from home at no cost. That will be helpful for Covid patients and also for healthy people who are not affected by Covid. In this way, we can save people from Covid-19. Lastly, this project shows a Graphical User Interface (GUI) application that has been operated for the Covid Prediction framework, which introduces the design and implementation of technology. It is believed that the proposal is very helpful for all people and greatly impacts on decreasing the Covid rate.

References:

- Worldometers, “Coronavirus,” 2021. Available: https://www.worldometers.info/coronavirus/?utm_campaign=homeAdvegas1?%22
- L. Wang, Z. Q. Linc and A. Wong, “COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images,” Scientific Reports, vol. 10, no. 19549, pp. 1–9, 2020.
- P. Patel, Normal CXR Dataset. [Online]. 2020. Available: <https://www.kaggle.com/prashant268/chest-xraycovid19-pneumonia>.