## Introduction:

During pandemic COVID-19, WHO has made wearing masks compulsory to protect against this deadly virus. So, we have developed a machine learning project – Real-time Face Mask Detector with Python.
We have built a real-time system to detect whether the person on the webcam is wearing a mask or not. We have trained the face mask detector model using Keras and OpenCV.

## Implementation:

We have divided this project in two parts. In the first part, we have created a basic CNN model using Keras and used our dataset to train the face mask detector model. In the second part, we test the results using a real-time webcam using OpenCV.

First, train.py file which is mainly used for training the neural network on our dataset.

**I. Import all the libraries and modules required:** we are using the following libraries and packages numpy, keras, tensorflow, sklearn, openCV.

**II. Build the neural network**: This convolution network consists of two pairs of Conv and MaxPool layers to extract features from the dataset and. Which is then followed by a Flatten and Dropout layer to convert the data in 1D and ensure overfitting, and lastly two Dense layers are used for classification. In all the layers, a RELU activation function is used except the output layer in which we used SOFTMAX. And we use ADAM optimizer to train the neural network in less time and more efficiently.

**III. Image Data Generation/Augmentation:** Here, we are using data augmentation for training the model based on various parameters such as rotation, sheer, shifting and zooming so that the model gets itself trained dynamically and thus will be more efficient in feature extraction and classification.

**IV. Initialize a callback checkpoint to keep saving best model after each epoch while training:** This function of keras callback is used to save the best model after every epoch. We just need to define a few of the parameters like where we want to store, what we want to monitor and etc. We have first defined the path and then assigned val_loss to be monitored, if it lowers down we will save it.

**V. Train the model:** Finally, we will be training the model based on the training and testing images as well as the augmented images.

Second, test.py file which is used for testing the model that we have trained.

**I. Accessing the webcam:** Here we are using the videocapture() function for accessing the camera.

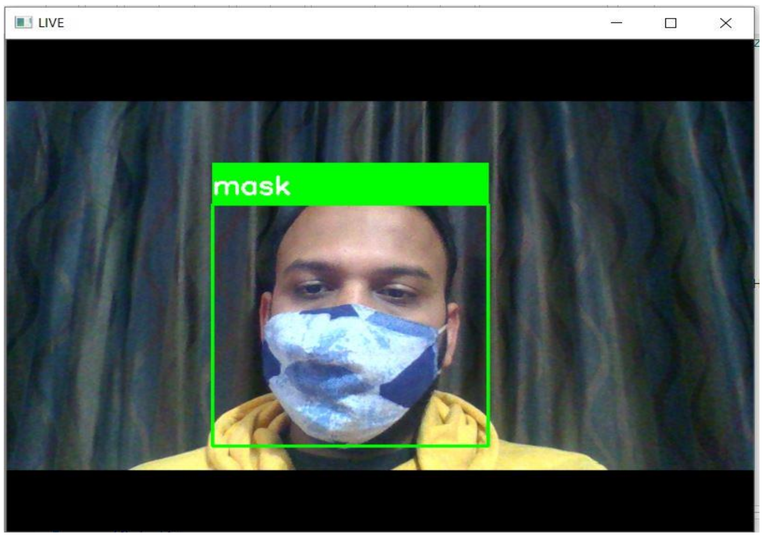
**II. Loading Classifier:** Here we are using Haar Cascade classifier for object detection. Haar Cascade classifier is an effective object detection approach. This is basically a machine learning based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images.

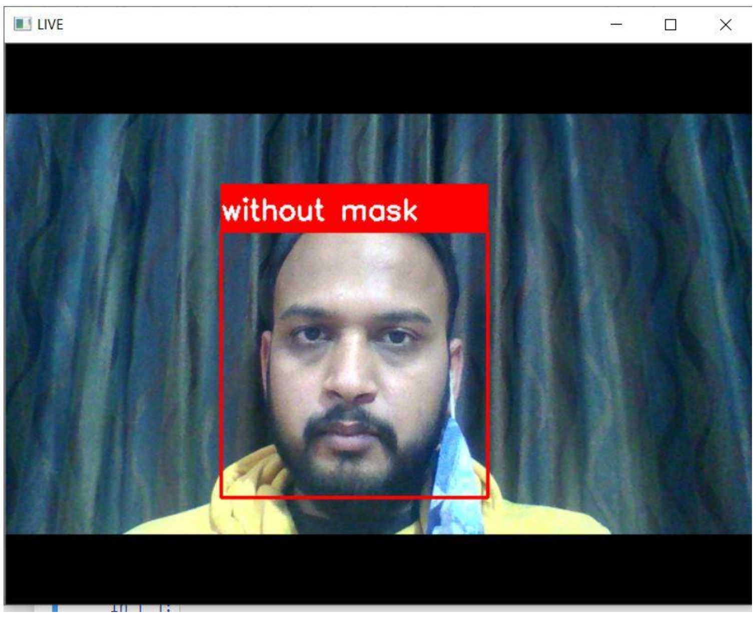
So how this works is they are huge individual .xml files with a lot of feature sets and each xml corresponds to a very specific type of use case.

**III. Detecting faces in the image and creating the region of interest:** We are flipping the image followed by resizing so that the computation can be made easy. Here, we will be determining the region of interest that will be provided to the model.

**IV. Displaying predicted class in the image and highlighting it:** Here, we will provide the region of interest as input to the model and accordingly our model will predict the class to which the input belongs. Following this, we will be drawing rectangles over the face, which if predicted as 'with mask' will be Green in color and if predicted as 'without mask', then will be Red.

## Output:




LIVE
without mask