VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

Project Report

Project Title	**: Face Mask Detection**
Course Name	**: Soft Computing**
Course Code	**: SWE1011**
Faculty	**: Prof. Subhashini R**
SLOT	**: B1+TB1**

Submitted By:

Jaswanth Kumar M – 19MIS0364

Rambabu Katari – 19MIS0382

Introduction:

We know that after the breakout of the worldwide pandemic COVID-19, has resulted in a global pandemic.Due to the rapid worldwide spread of Coronavirus Disease 2019 (COVID-19) has resulted not only in a global pandemic and also for Strict healthy measures.There arises a severe need of protection mechanisms, face mask being the primary one.

Problem Definition:

There is a restructure in our daily routine in daily life with the COVID-19. People are avoiding staying outside the home and a lot of changes occur in their day to day life.

Rule of usage of wearing a mask is made compulsory to those who come outside their homes. But, face mask detection has become a crucial task to help global society. With the rapid growth of the machine learning techniques, automatic detection of face mask can be implemented by using image processing techniques

Solution Domain:

So, we can overcome this problem by using machine learning techniques like image processing.

The basic aim of the project is to detect the presence of a face mask on human faces on live streaming video as well as on images..The main step involved in recognizing the presence of a mask is to detect the face.

This again will be divided into two parts:

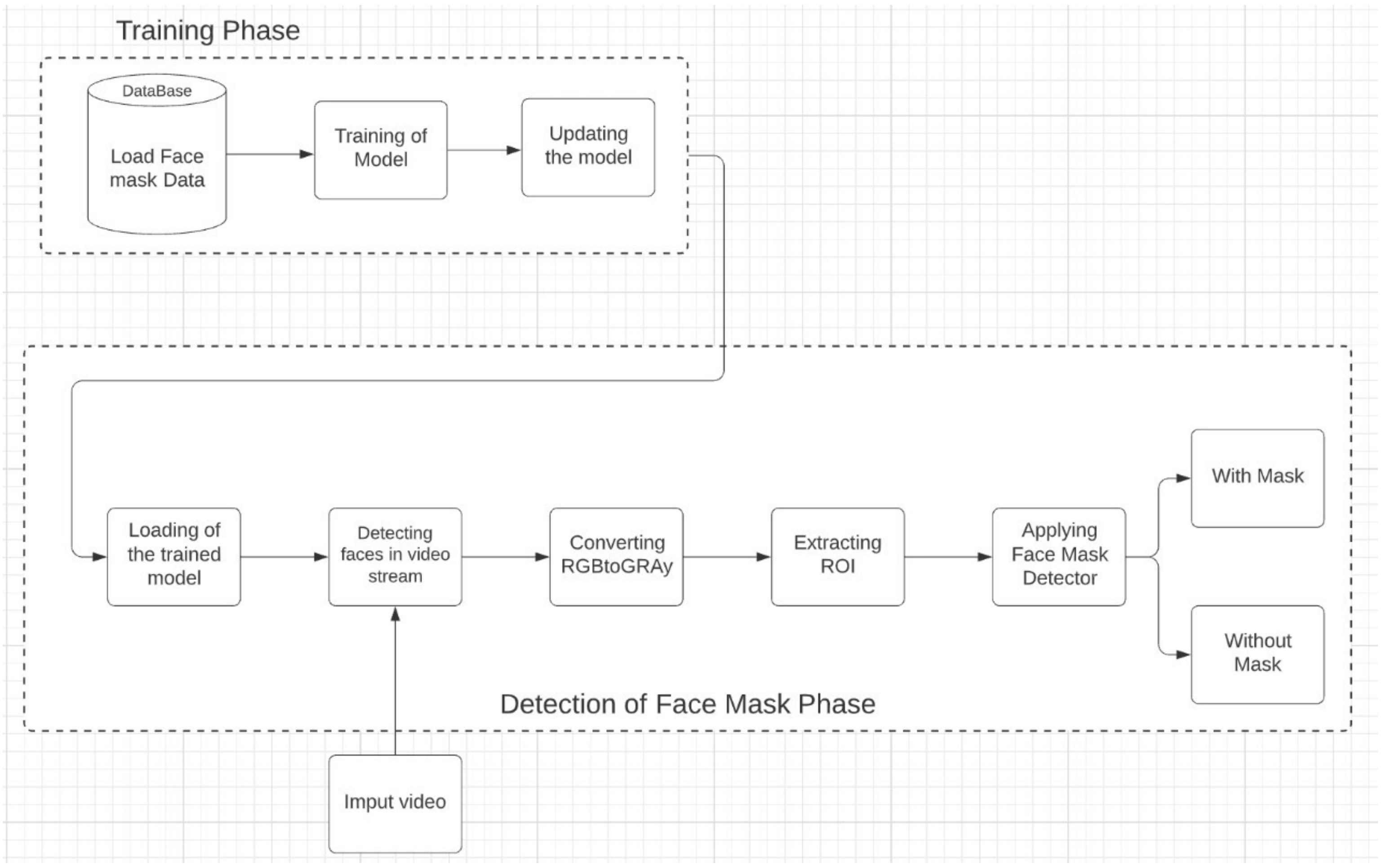
- To detect Faces
- To detect masks on the faces

Our project presents a simplified approach to serve the above purpose using Convolutional neural networks

and the basic Machine Learning (ML) packages such as

- TensorFlow,
- Keras and
- OpenCV through python.

II) Complete Design:



III) Module Description:

Data Pre-Processing:

Pre-processing refers to the transformations applied to your data before feeding it to the algorithm.
It is important for data pre-processing to be done correctly as not to negatively affect the end product, or data output.

In Data pre-Processing, We need to convert the datasets which contain the folders with mask and without mask into dictionaries by labeling with respective keys.

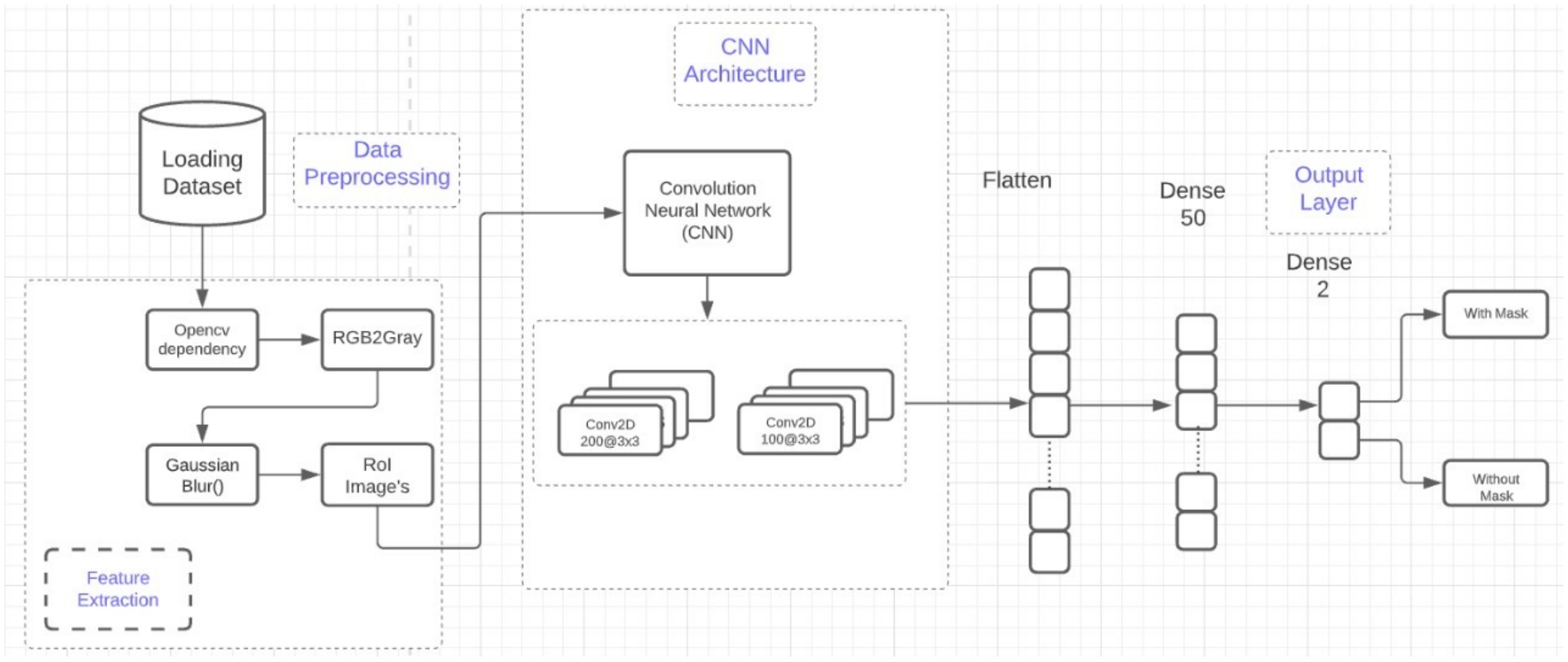
Convolutional Neural Networks training:

Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk

Detector Training:

Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with_mask or without_mask

IV) Detailed workflow Design:



V) Literature Survey:

Author	Dataset	Year	Summary of Used Method	Performance in men- tioned literature	Ref No
Shimming et al.	MAFA	2017	A novel dataset and face occlusion detection technique using LLE-CNN and its three separate modules	Average precision 74.6%	[1]
Wei et al.	1. WiderFace (Pre-trained) 2.MASKED FACES		Three CNN classifiers were combined to detect a masked face and named cascade framework of CNN	Accuracy is 86.6% and recall is 87.8%	[2]
Qin et al.	MMD	2020	Image Super-Resolution network (SR) and a classification network were combined to identify facemask-wearing conditions. Images with low resolution improved with an SR network, and MobileNetV2 was used for the classification network	Accuracy is 98.70%	[3]

Madhura et al.	Customized		An 8 layer CNN was used to detect faces, extract RoI and detect face masks. The detector was applied both on input images and video streams.	Accuracy is 98.6%	[4]
Amit et al.	1.RMFRD 2.FMDD 3.FFHQ		A two-stage detection algorithm consisted of face detection and facemask detection. It first detected a face (RetinaFace) and then detected a facemask (NASNetMobile)	Precision- 98.28%	[5]
Shashi Yadav	Customized dataset		A real-time social distance maintenance and a facemask detector was deployed in raspberry pi4 using a combination of SSD and MobileNetV2. Violating instructions resulted in alarm	Precision of 91.7% with a confidence score of 0.7	[6]
Jiang et al.	FMD		A facemask detector consisted of CNNs, FPN, and a context attention module. The models used were MobileNet and ResNet	-	[7]
Nieto-Rodr0ıguez et al.	Mask Augsburg Speech Corpus (MASC)	2015	This method used color filters for classification of face and facemask in operating room with the help of skin texture in HSV color space	74.6% accuracy	[8]

Ristea et al.	1.BAO 2.LFW	2020	Detects facemask from speech using data augmentation technique with multiple ResNet that have been trained by GANs	Recall above 95%, False positive rate below 5%	[9]
Loey et al.	1.RMFRD 2.SMFD 3.LFW		Facemask classification algorithm whose feature extraction was done ResNet-50 and classification is done using SVM, DT and ensemble classifiers	RMFD, SMFD, and LFW datasets showed 99.64%, 99.49% and 100% testing accuracy respectively	[10]

[1] Shimming et al. has designed an algorithm that aims to detect covered faces, in which " mask " refers to any facial mask and visible masks.They divided their model into three main parts: (a) proposal module, (b) embedded module, and (c) verification module. The first module integrates two CNNs and removes features from face images. The second module is dedicated to detecting non-surface facial features that occur with closure.The average reported accuracy was 74.6%.

[2] Wei et al. proposed a cascaded CNN structure to identify the masked face, which is composed of three complete CNNs. Their data set contains only 200 images, which is not enough to be included in any DL-based framework. DL-based structures require a large number of databases to be trained to achieve maximum performance. To overcome this problem, they used a pre-trained model with the Wider Face database, and then tuned.Their model has 86.6% accuracy and 87.8% memory in their MASKED FACE test database.

[3] Qin et al had worked on mask-wearing condition identification algorithm using image super resolution and classification network.They Merge and titled it as SRCNet along three categories

no face mask, incorrect face mask appropriate face mask conditions.The CNN algorithm used here was adopted from mobilenetThe classifier predicted the image condition and classified them into the respective categories and their model showed 98.70% accuracy

[4] In this paper, the author proposed Madhura et al. implemented a model called Facemask Net to identify if a person is wearing a facemask properly or not.His dataset contains 35 images by the combination of masked and unmasked faces.he prepropsed and trained into desired value he want.After getting roi, the facemask net mode is applied to roi images for classification. His facenet mask model got accuracy of 98.6%.

[5] In this paper, irrespective of other author amit et al developed a two stage based detecto using two stage based detectors. The authors used the RetinaFacemodel as the first stage detector and NASNetMobile for the classifier model for comparison. So, in between these two detectors the intermediate phase begin and collects the images from model 1 and passed it ive video stream upon classifying.His model got 98.28% accuracy

[6] Shashi yadav developed a real time social distance maintenance and a face mask detector using raspberry pi. He developed this by using a ssd and mobilenetv2 together.The study offers three important criteria in the postCovid world: (i) finding people who use MobileNetV2 released by OpenCV and TensorFlow, (ii) Euclidean rating distances between found persons, and (iii) found that people wear a mask or not. In (iii), they continued training pictures of covered and uncovered faces to separate the mask as well there is no effective mask category.The model achieved accuracy of 91.7%

[7] This paper consists of a significant face detector model named RetinalFaceMask developed using backbone network by Jiang et al.The backbone network refers to the extraction component in DL. RetinaFacemask adopts ResNet as the primary backbone, and MobileNet as the comparative backbone. It is a type of DL, where information of the model is transferred from one

frame to another in order to measure objectives. RetinaFaceMask model built of a very powerful network, sometimes leading to high computation

[8] Ristea et al. fostered a technique for facemask discovery from speech. This model comprised of two sections, I) preparing Generative Adversarial Networks (GANs) with cycle in the middle of two classes (with veil and without cover), and ii) Assigning inverse marks to each changed elocution, creating new preparation emphasizes utilizing cycle-consistent GANs. The underlying and changed accents were changed over to spectra that were utilized as contributions to ResNet networks with various profundities. The networks were gathered by grouping the SVMs. In this cycle, augmented spectrograms were used to prepare the model.

[9] Nieto-Rodr0 ıguez et al. presented a constant facemask detection system that sets off an alert when medical services staff don't wear careful masks in the clinical or working room. They involved two detectors and two color filters for each identifier. One of them was a face identifier, and one more was a clinical mask identifier. The face discovery was developed utilizing the customary Viola-Jones face detection algorithm

[10] Loey et al. proposed a hybrid infrastructure by which feature extraction is developed using ResNet-50, and classification by dividing into three categories (SVM, decision tree, and ensemble classifier). The database is divided into three parts: 70% training, 10% validation, and 20% for testing. The model was trained, tested, and validated Real-World Mask Face Dataset (RMFD)

VII) S/W tools used:

Tools used for the development are as follows:

- Jupyter Notebook - (Anaconda Navigator)
- Python - Language

VIII) Dataset name:

Kaggle - Face_mask_detection_dataset - Customized Dataset - [Prajna Bhandary]

Dataset was developed by Prajna Bhandary which contains two datasets named as:

- With Mask
- Without Mask

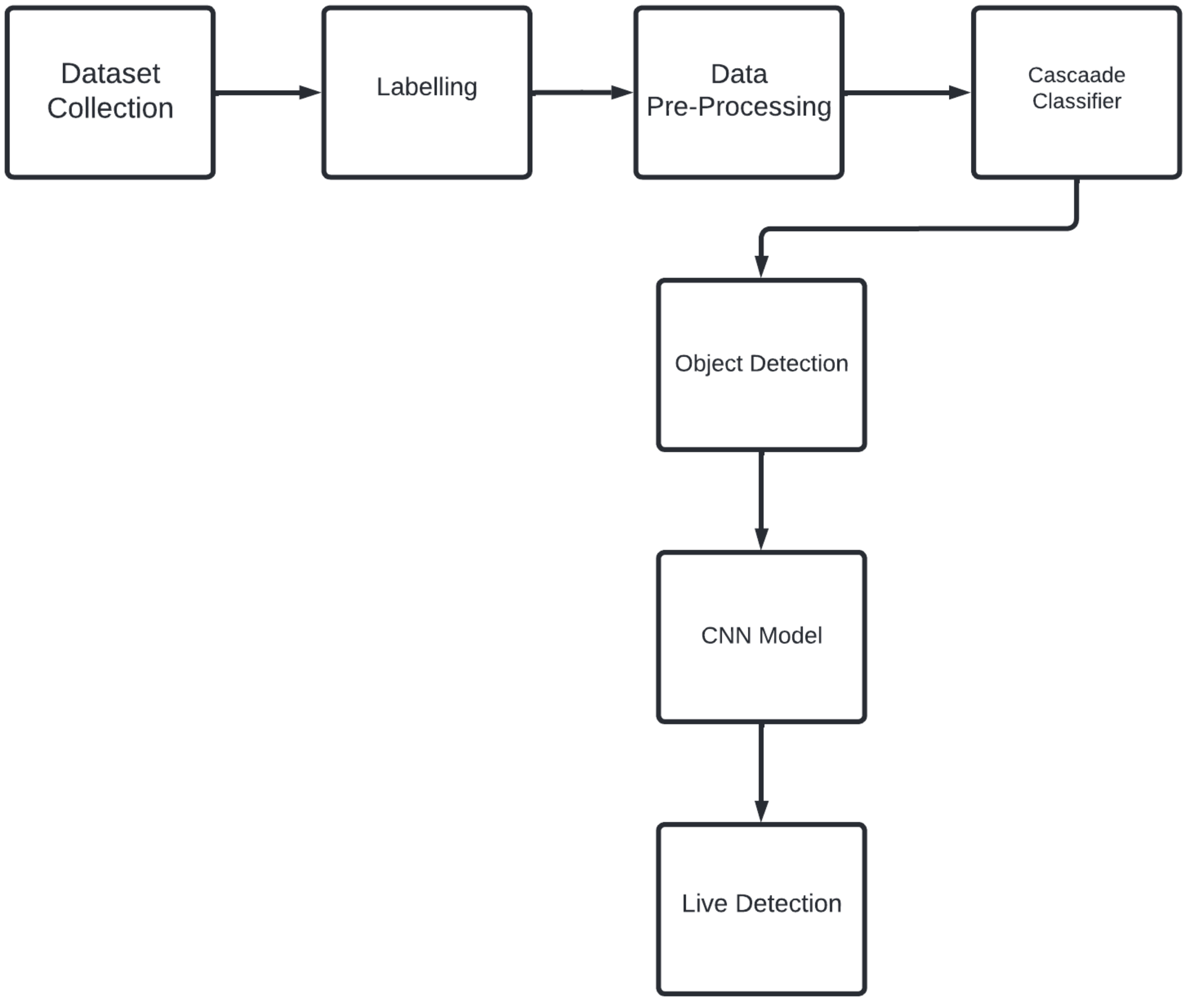
Total No of Images	:	690+686
Dataset	:	Customized dataset
Characteristic	:	Public Daraset
Description	:	Author Claimed to use customized Dataset
Composition	:	Contains both faced and unfaced images of Different Persons.

Limitations:

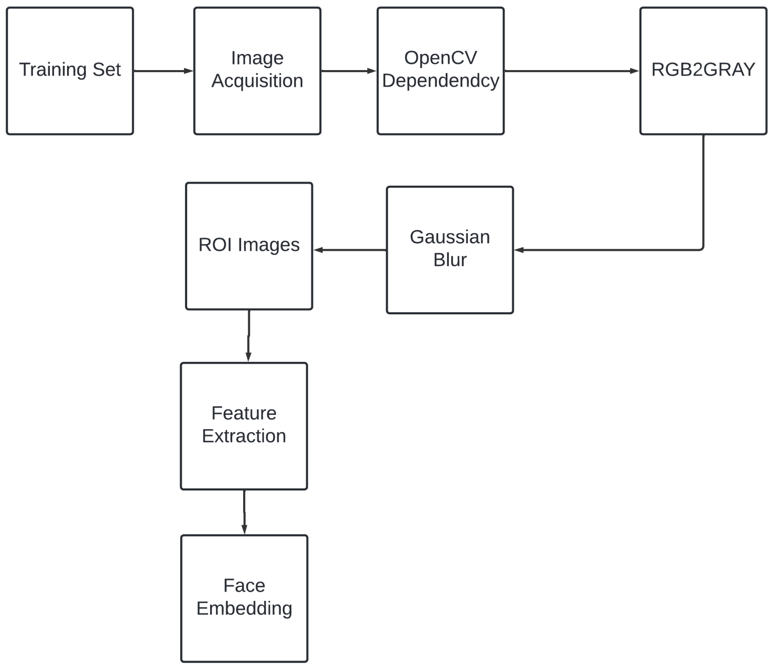
1. Limited in Number
2. Biased towards asian

Review-2

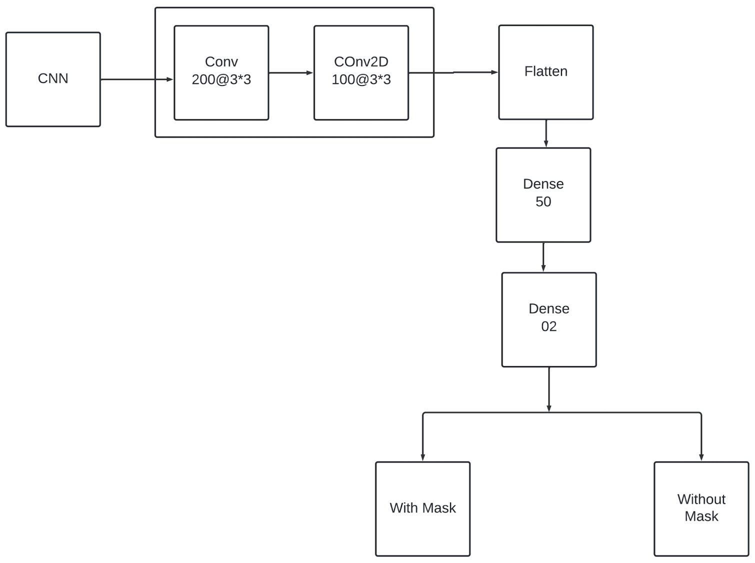
Segmented Architecture diagram with mathematical/algorithmic/ Technical Description:

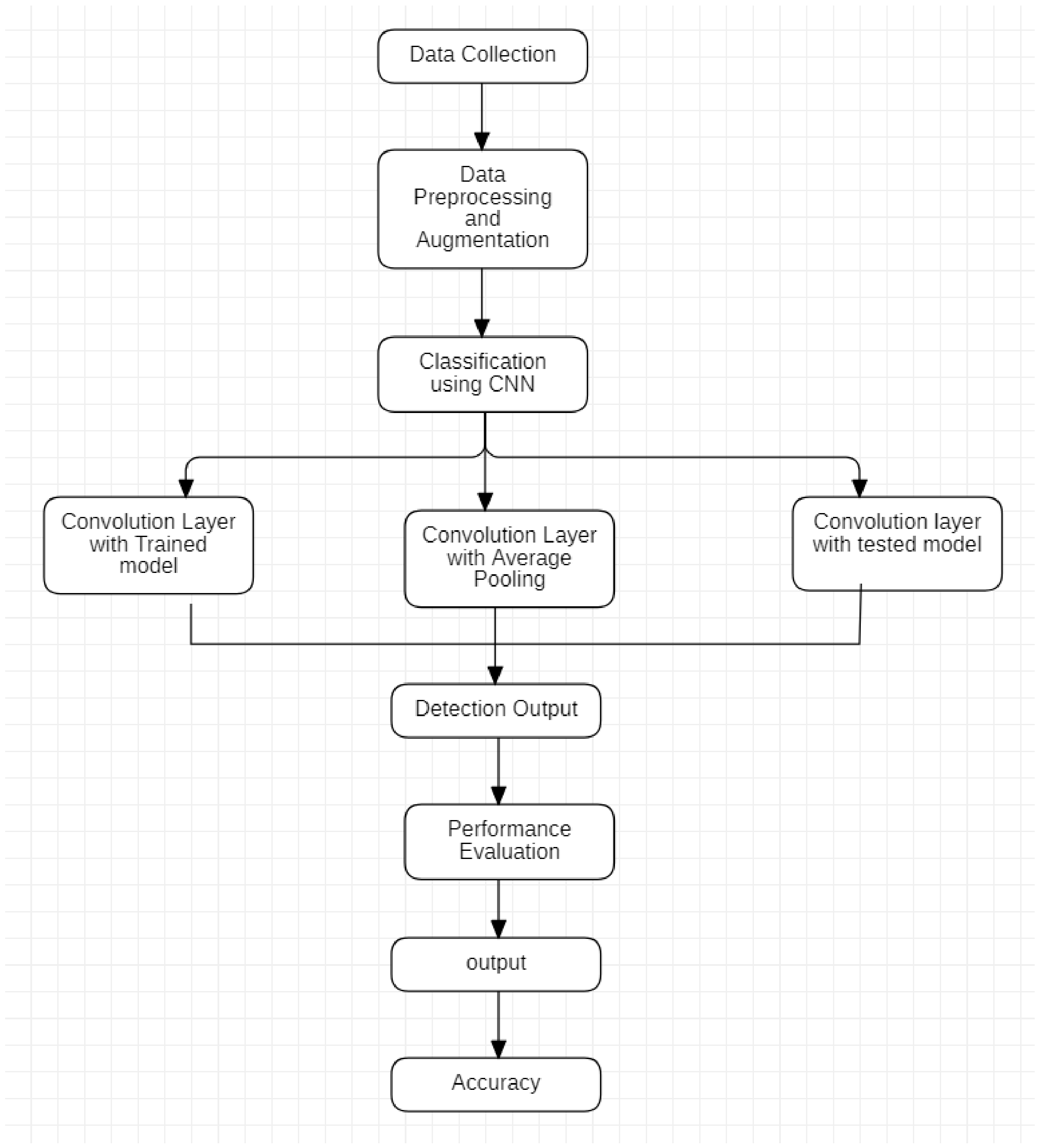


Data Preprocessing:



CNN MODEL:




Data Collection
Data Preprocessing and Augmentation
Classification using CNN
Convolution Layer with Trained model
Convolution Layer with Average Pooling
Convolution layer with tested model
Detection Output
Performance Evaluation
output
Accuracy

Technical Description :

Our system's goal is to identify those who aren't wearing a face mask. The learning architecture produces a response based on the input image, categorizing it as mask or no mask. If a person is spotted without a mask on, a beep alarm will sound until the mask is put on.

The learning model is based on CNN, which is an image pattern recognition algorithm that is particularly useful. The data from both classes must be seen by the neural network. An input layer, many hidden layers, and an output layer make up the network. Multiple convolution layers make up the hidden layers. Multiple dense neural networks utilize the features generated by CNN for categorization purposes. Three pairs of convolution layers are followed by one max pooling layer in the architecture. The convolution layer has 100 kernels with a 3x3 window size. Window size 2x2 maximum pooling layer This layer will combine the results from the preceding convolution layer and select the maximum value inside the 2x2 window.

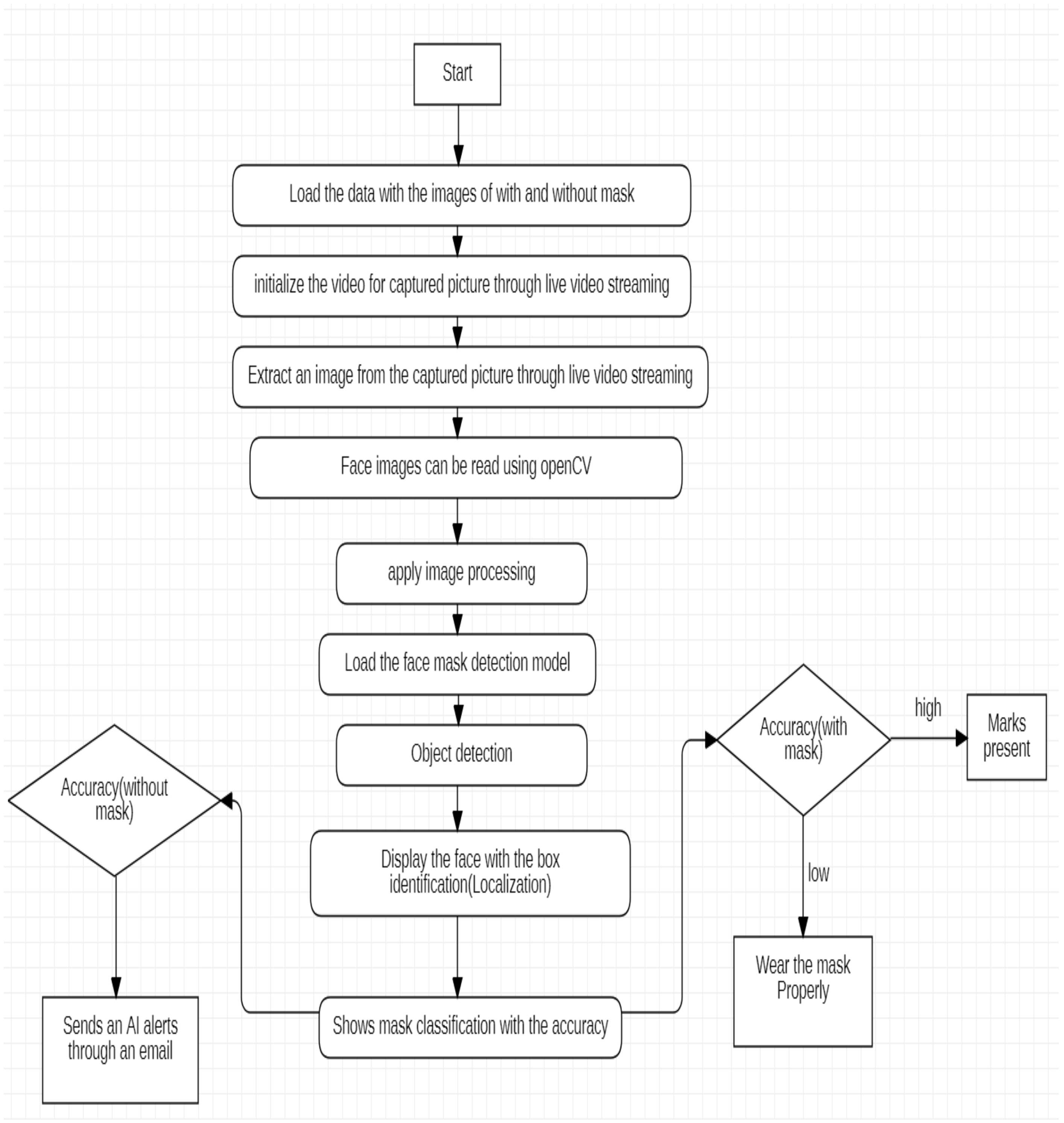
It reduces the number of parameters by shrinking the representation's spatial size. As a result, the network's computation is simplified. The convolution layers' output will be flattened and turned into a 1-D array. Then there are two thick layers and one dropout layer. By dropping out units, the dropout layer protects the network from overfitting. The dense layer is made up of a group of neurons that each learn nonlinear characteristics. The result will be flattened and given to the first dense layer of 50 nodes. Finally, there is a second dense layer with two nodes, as there are two classes.

(i) Training the model with the taken dataset

(ii) Deploying the model

We have tested the model for different conditions with different hyper parameters, for which the results are mentioned in the next section. First we feed the dataset in the model, run the training program, whichtrainsthe model on the given dataset. Then we run the detection program, which turns on the video stream, captures the frames continuously from the video stream with an anchor box using an object detection process.

Complete Design flowchart with technical details (i,e diagram with algorithm/technique used):


Start
Load the data with the images of with and without mask
initialize the video for captured picture through live video streaming
Extract an image from the captured picture through live video streaming
Face images can be read using openCV
apply image processing
Load the face mask detection model
Object detection
Display the face with the box identification(Localization)
Shows mask classification with the accuracy
Accuracy(without mask)
Sends an AI alerts through an email
Accuracy(with mask)
high
Marks present
low
Wear the mask Properly

Technical details:

Data Preprocessing:

The images in the dataset are not all the same size, so preprocessing was required for this study. The training of deep learning models necessarily requires a large amount of data. We used Keras' Image Data Generator method to resize all of the images to 256 × 256 pixels. We normalized all images after converting them to 256 × 256. For faster calculation, images are converted to NumPy arrays. Increase the amount of data by rotating, zooming, shearing, and horizontal flipping. Images are gathered as well. The images are then resized to 128 x 128 for passing through the second convolution layer, and then to 64 x 64 for passing through the third convolution layer.

CNN Model:

For classification and image processing, CNN is used. CNN consists of one or more convolution layers. CNN aims to find features that are effective inside an image rather than working with an entire image. There are several secret layers in CNN, as well as an input layer and an output layer.

Transfer Learning:

The method of using MobileNetV2 is called Transfer Learning. Transfer learning is using some pre-trained model to train your present model and get the prediction which saves time and makes using training the different models easy. We tune the model with the hyper parameters : learning rate, number of epochs and batch size. The model is trained with a dataset of images with two classes, withmaskand without mask.

Technical description of our project:

To develop an efficient network for the identification of facemasks, we used a python script, tensor flow, and CNN as deep learning architecture. Our goal is to train a bespoke CNN model to detect whether or not someone is wearing a mask. This project is capable of quickly detecting the mask's faces from any angle. It converts an RGB input image into an output image in any orientation. This function's main task is to extract features from photos and classify them. The image is sketched and turned into a new image in the feature extraction system, and the image generated is more efficient than the prior image. In this section, the dimensionality of images is reduced to an efficient representation.

The camera can be used to detect the face of the mask in our suggested model. To begin, resize the input image to 100*100 pixels then extract and predict features. It gives us some model data with their accuracy level after the training process is completed. Later, we'll use that model to forecast the outcome using a webcam.

Implementation (75%):

Code:

Model Generation

```
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
import tensorflow as tf
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

from google.colab import files
from sklearn.metrics import classification_report,confusion_matrix


!unzip \*.zip


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


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)
  plt.title(target_class)
  plt.axis("off")
  print(f"image shape {img.shape}")

  return img


data=[]
labels=[]
no_mask=os.listdir("/content/New Masks Dataset/Train/Non Mask/")
for a in no_mask:
  image = cv2.imread("/content/New Masks Dataset/Train/Non Mask/"+a,)
  image = cv2.resize(image, (224, 224))



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

no_mask=os.listdir("/content/New Masks Dataset/Test/Non Mask/")
for a in no_mask:

  image = cv2.imread("/content/New Masks Dataset/Test/Non Mask/"+a,)
  image = cv2.resize(image, (224, 224))



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




mask=os.listdir("/content/New Masks Dataset/Train/Mask/")
for a in mask:

  image = cv2.imread("/content/New Masks Dataset/Train/Mask/"+a,)
  image = cv2.resize(image, (224, 224))



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



mask=os.listdir("/content/New Masks Dataset/Test/Mask/")
for a in mask:

  image = cv2.imread("/content/New Masks Dataset/Test/Mask/"+a,)
  image = cv2.resize(image, (224, 224))
    data.append(image)
    labels.append(1)



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



X_train,   X_test,   y_train,   y_test   =   train_test_split(data,   labels,   test_size=0.1,
random_state=42,shuffle=True,
                                        stratify = labels)



base_model   =   tf.keras.applications.MobileNet(input_shape=[224,224,3],   weights   =
"imagenet", include_top=False)

base_model.trainable = False

# for layer in base_model.layers[30:]:
#   layer.trainable = False


model = Flatten()(base_model.output)
model = Dense(units=256, activation="relu")(model)
model = Dense(units=64, activation="relu")(model)
prediction_layer = Dense(units=1, activation="sigmoid")(model)

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



model.fit(X_train, y_train, epochs=15,validation_split= 0.1, batch_size=32)
```

```
predictions = model.predict(X_test)

predict=[]

for i in range(len(predictions)):
  if predictions[i][0]>0.5:
    predict.append(1)
  else:
    predict.append(0)



pd.DataFrame(confusion_matrix(y_test, predict), columns= ["No Mask", "Mask"], index = ["No Mask", "Mask"])
```

Results and discussion (as of second review) + dataset description:

- The CNN Model Got developed and this model got used for prediction along with haar cascade classifier, a machine learning object detection program that identifies objects in an image and video. which justifies the objective of the project
- Dataset went through manual sorting and applied augmentation on the dataset to increase the accuracy of model for prediction

Review-3

Innovation Introduced in Design:

- Check Individuals And Crowds Wearing Masks In Public
- Use Digital Screens To Remind Visitors To Wear Masks
- Alert Staff When No Masks Are Detected
- Works With Existing USB Or IP Cameras With RTSP Streams
- Anonymous & Spoof Proof

Ultimate Utility value of the project to the society and/or industry:

Face Mask Detection, an application to solve the problem of breaking the distribution chain of COVID-19 in the public service area, because when someone is outside the house there will be lots of threats of virus transmission. So the whole community needs to use masks. Then, we aim to develop applications and investigate mask detectors to prevent Covid-19 because the use of masks can prevent the transmission of sparks that can be used to protect others, and help contaminate the environment due to this spark, to minimize the spread of COVID-19 due to the use of masks which has been monitored in public areas.

- Check Individuals And Crowds Wearing Masks In Public
- Use Digital Screens To Remind Visitors To Wear Masks
- Alert Staff When No Masks Are Detected
- Works With Existing USB Or IP Cameras With RTSP Streams
- Anonymous & Spoof Proof

New Learning experience gained (Like using new software package):

- Numpy
- OpenCV
- HaarCascade
- TensorFlow

Numpy:

NumPy is a scientific computation package It offers many functions and utilities to work with N-Dimension arrays Largely used by other libraries such as OpenCV, TensorFlow and PyTorch to deal with multi dimensional arrays (e.g., tensors or images)

OpenCV:

OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.

HaarCascade:

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper "Rapid Object Detection using a Boosted Cascade of Simple Features" published in 2001.

TensorFlow:

TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use. The tf. data API enables you to build complex input pipelines from simple, reusable pieces.

Explanation about complete description:

In order to train a custom face mask detector, we need to break our project into two distinct phases,

Training:

Here we'll focus on loading our face mask detection dataset from disk, training a model on this dataset, and then serializing the face mask detector to disk

Deployment:

Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as `with_mask` or `without_mask`

Step 1: Data Visualization:

In the first step, let us visualize the total number of images in our dataset in both categories. We can see that there are 690 images in the class and 686 images into respective classes.

Step 2: Data Augmentation

In the next step, we augment our dataset to include more images for our training. In this step of data augmentation, we rotate and flip each of the images in our dataset. We see that, after data augmentation.

Step 3: Splitting the data:

In this step, we split our data into the training set which will contain the images on which the CNN model will be trained and the test set with the images on which our model will be tested.

In this, we take split_size =0.8, which means that 80% of the total images will go to the training set and the remaining 20% of the images will go to the test set.

Step 4: Building the Model

In the next step, we build our Sequential CNN model with various layers such as Conv2D, MaxPooling2D, Flatten, Dropout and Dense. In the last Dense layer, we use the 'softmax' function to output a vector that gives the probability of each of the two classes.

Step 5: Pre-Training the CNN model

After building our model, let us create the 'train_generator' and 'validation_generator' to fit them to our model in the next step. We see that there are a total of 2200 images in the training set and 551 images in the test set.

Step 6: Labeling the Information

After building the model, we label two probabilities for our results. ['0' as 'without_mask' and '1' as 'with_mask']. I am also setting the boundary rectangle color using the RGB values.['RED' for 'without_mask' and 'GREEN' for 'with_mask]

Step 7: Importing the Face detection Program

After this, we intend to use it to detect if we are wearing a face mask using our PC's webcam. For this, first, we need to implement face detection. In this, I am using the Haar Feature-based Cascade Classifiers for detecting the features of the face.
This cascade classifier is designed by OpenCV to detect the frontal face by training thousands of images.

Step 8: Detecting the Faces with and without Masks

In the last step, we use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the Cascade Classifier. The code `webcam = cv2.VideoCapture(0)`denotes the usage of webcam.The model will predict the possibility of each of the two classes . Based on which probability is higher, the label will be chosen and displayed around our faces.

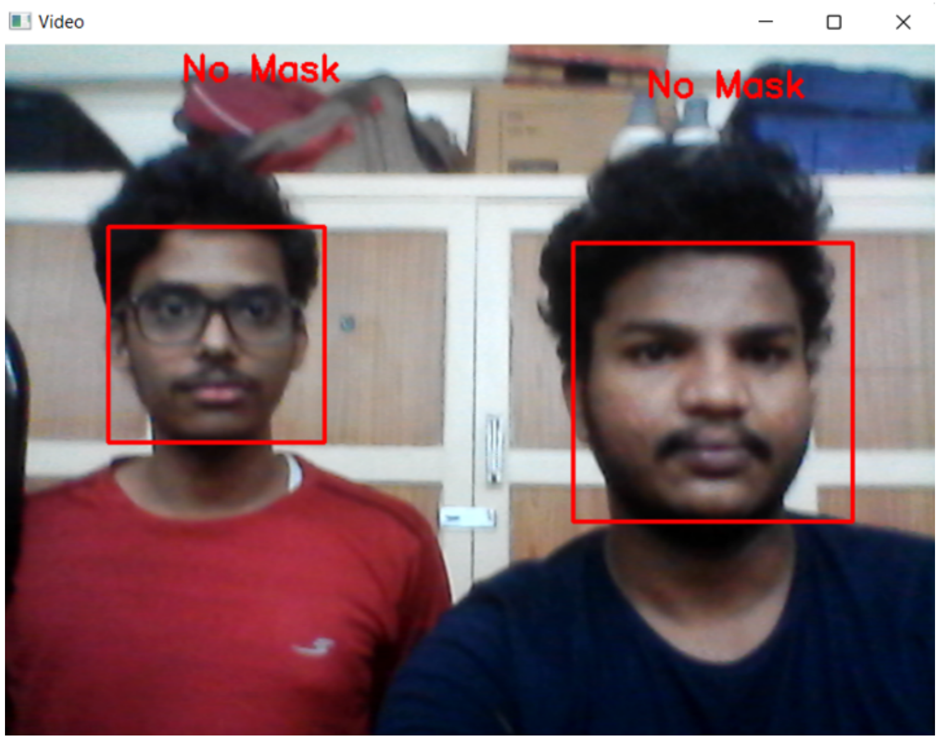
Implementation:

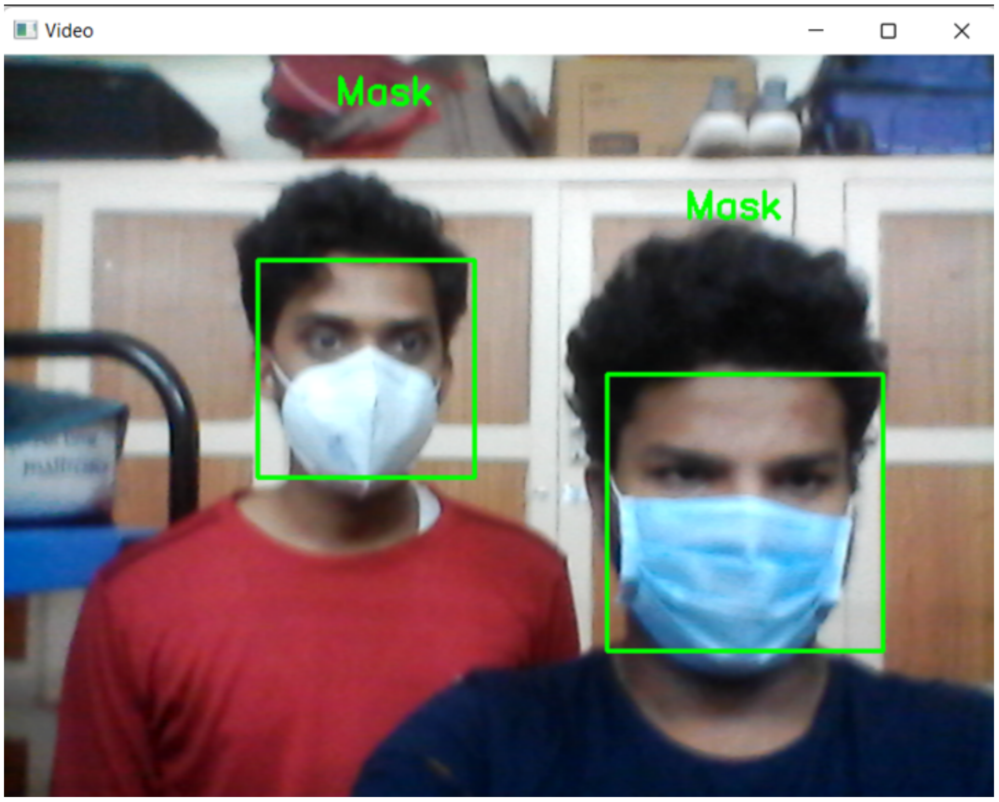
Now, we are importing our model that was developed in Rev 2 along with Haar Cascade Classifier,

```
import cv2
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import numpy as np
```

```
%config Completer.use_jedi = False



face_classifier = cv2.CascadeClassifier("\haarcascade_frontalface_default.xml")
mask_detection = tf.keras.models.load_model("mask_detection_best.h5")

text_mask = "Mask"
text_no_mask = "No Mask"
font = cv2.FONT_HERSHEY_SIMPLEX
scale = 0.8

def predict(image):

    face_frame = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    face_frame = cv2.resize(face_frame, (224, 224))
    face_frame = img_to_array(face_frame)
    face_frame = np.expand_dims(face_frame, axis=0)
    face_frame =  preprocess_input(face_frame)
    prediction = mask_detection.predict(face_frame)

    return prediction[0][0]

def detector(gray_image, frame):

    faces = face_classifier.detectMultiScale(gray_image, 1.1, 5)

    for (x,y,w,h) in faces:

        roi_color = frame[y:y+h, x:x+w]
        mask = predict(roi_color)


        if mask > 0.5:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 2)
                cv2.putText(frame, text =text_mask, org =(x+50,y-100), fontFace =font, fontScale =
scale, color =(0,255,0),
            thickness = 2)

        elif mask<=0.5:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0,0,255), 2)
              cv2.putText(frame, text =text_no_mask, org =(x+50,y-100), fontFace =font, fontScale =
scale , color =(0,0,255),
            thickness = 2)

    return frame



video_cap = cv2.VideoCapture(0)

while True:

    ret, frame = video_cap.read()
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    detect = detector(gray_frame, frame)

    cv2.imshow("Video", detect)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

video_cap.release()
cv2.destroyAllWindows()
```

Results:





Considering the High quality external Webcam accesses by the help of openCV and this test was performed in the Open Ground at Proper Light exposure,

Note: while accessing an external web camera, **video_cap = cv2.VideoCapture(0)** should be changed to **video_cap = cv2.VideoCapture(1).**

Index	Distance	Validation	Prediction Accuracy(%)
0	2ft	yes	99
1	4ft	yes	98
2	8ft	yes	97
3	16ft	yes	91
4	32ft	yes	70