**Micro-Project Report Mask Detection**

**Rationale**

The aim of this Micro project is to detect the mask where a person were a mask or not

**Aims/Benefits of the Micro-Project**

The implemented Object Detection using OpenCv module can open up the application (if it’s installed in the system) draw the graphic.

**Course Outcomes Addressed**

|  |  |
| --- | --- |
| **Co. No.** | **Course Outcome Statement** |
| 1. | Use Python IDE to display message on screen |
| 2. | Develop Python program to demonstrate use of operators |
| 3. | Demonstrate operations on data structures in Python |
| 4. | Develop functions for given problem |
| 5. | Use Exception Handling |

**Literature Review**

Face Mask detection had seen significant progress in the domains of Image processing and Computer vision, since the rise of the Covid-19 pandemic.COVID-19 has had a lasting impact in many countries worldwide since 2019.

This model can be used for safety purpose since it is very resource efficient to deploy . Face mask detection has become a trending application due to the Covid-19 pandemic, which demands a person to wear face masks. Face mask detection has turned up to be astonishing problem in the demand of image processing and computer vision.

**Actual Methodology Followed** write step wise work done, (write member wise contribution in brief)

Followed by algorithm, flowchart and program.

* + 1. The actual video frame read (one frame on each loop)
    2. A return code
       - scaleFactor – Parameter specifying how much the image size is reduced at each image scale.
       - minNeighbors – Parameter specifying how many neighbours each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces.
       - flags –Mode of operation
       - minSize – Minimum possible object size. Objects smaller than that are ignored.
       - The original image
       - The coordinates of the top-left point of the detection
       - The coordinates of the bottom-right point of the detection
       - The colour of the rectangle (a tuple that defines the amount of red, green, and blue (0- 255)).In our case, we set as green just keeping the green component as 255 and rest as zero.
       - The thickness of the rectangle lines

# Code:

# import the necessary packages from

tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.preprocessing.image import img\_to\_array

from tensorflow.keras.models import load\_model

from imutils.video import VideoStream import numpy as np

import imutils import time import cv2 import os

def detect\_and\_predict\_mask(frame, faceNet, maskNet):

# grab the dimensions of the frame and then construct a blob

# from it

(h, w) = frame.shape[:2]

blob = cv2.dnn.blobFromImage(frame, 1.0, (224,

224),

(104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections

faceNet.setInput(blob) detections = faceNet.forward() print(detections.shape)

# initialize our list of faces, their corresponding locations,

# and the list of predictions from our face mask network

faces = [] locs = [] preds = []

# loop over the detections

for i in range(0, detections.shape[2]): # extract the confidence (i.e.,

probability) associated with

# the detection

confidence = detections[0, 0,

startY, endX, endY))

locs.append((startX,

i, 2]

# filter out weak detections

# only make a predictions if at least one face was detected

if len(faces) > 0:

by ensuring the confidence is

# greater than the minimum

confidence

if confidence > 0.5:

# compute the (x, y)- coordinates of the bounding box for

# the object

box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])

(startX, startY, endX, endY) = box.astype("int")

# ensure the bounding boxes fall within the dimensions of

# the frame (startX, startY) =

(max(0, startX), max(0, startY))

(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

# extract the face ROI, convert it from BGR to RGB channel

# ordering, resize it to 224x224, and preprocess it

face = frame[startY:endY, startX:endX]

face = cv2.cvtColor(face, cv2.COLOR\_BGR2RGB)

face = cv2.resize(face, (224, 224))

face =

img\_to\_array(face)

face = preprocess\_input(face)

# add the face and bounding boxes to their respective

# lists faces.append(face)

# for faster inference we'll make batch predictions on all

# faces at the same time rather than one-by-one predictions

# in the above `for` loop faces = np.array(faces,

dtype="float32")

preds = maskNet.predict(faces, batch\_size=32)

# return a 2-tuple of the face locations and their corresponding

# locations

return (locs, preds)

# load our serialized face detector model from disk

prototxtPath = r"face\_detector\deploy.prototxt" weightsPath = r"face\_detector\res10\_300x300\_ssd\_iter\_14 0000.caffemodel"

faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk

maskNet = load\_model("mask\_detector.model")

# initialize the video stream print("[INFO] starting video stream...") vs = VideoStream(src=0).start()

# loop over the frames from the video stream

while True:

# grab the frame from the threaded video stream and resize it

# to have a maximum width of 400

pixels

frame = vs.read()

frame = imutils.resize(frame,

# show the output frame cv2.imshow("Frame", frame)

width=400)

# detect faces in the frame and determine if they are wearing a

# face mask or not (locs, preds) =

detect\_and\_predict\_mask(frame, faceNet, maskNet)

# loop over the detected face locations and their corresponding

# locations

for (box, pred) in zip(locs, preds): # unpack the bounding box

and predictions

(startX, startY, endX, endY)

key = cv2.waitKey(1) & 0xFF

# if the `q` key was pressed, break from the loop

if key == ord("q"): break

# do a bit of cleanup cv2.destroyAllWindows() vs.stop()

= box

(mask, withoutMask) = pred

# determine the class label and color we'll use to draw

# the bounding box and text label = "Mask" if mask >

withoutMask else "No Mask"

color = (0, 255, 0) if label ==

"Mask" else (0, 0, 255)

the label

# include the probability in

label = "{}:

{:.2f}%".format(label, max(mask, withoutMask) \* 100)

# display the label and bounding box rectangle on the output

# frame cv2.putText(frame, label,

(startX, startY - 10),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.45, color, 2)

cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2

**Outputs of the Micro-Projects:**



**Skill Developed / Learning outcomes:**

* + 1. Develop a program using NumPy, matplotlib.
    2. Develop functions for given problem.
    3. Handle user defined exceptions.
    4. Make use of Turtle efficiently.

**Applications of the Micro-Project:**

* 1. This object detection project simply used in this current covied-19 situation for detecting mask
  2. It is simply saved time of person for detecting of mask.
  3. System automatically detect mask if person wear or not and show warning.