International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 03 | March - 2023 Impact Factor: 7.185 ISSN: 2582-3930

Face Mask Detection Using TensorFlow, Keras and OpenCV

Bhanu Pratap Sahoo
19SCSE1050001
B.Tech-CSE
Galgotias University

Satvik Tandon
19SCSE1120002
B.Tech-CSE
Galgotias University

Abstract—***During the COVID-19 pandemic, WHO has made it mandatory to wear masks to protect against this deadly virus. In this tutorial, we will develop a machine learning project, a real-time face mask detector using Python. Build a real-time system to detect whether a webcam person is wearing a mask. Train a face mask detector model using Keras and OpenCV.***

We will create this project in two parts. In the first part, we write a Python script that Keras uses to train the face mask detector model. Part 2 uses OpenCV to test the results on a real-time webcam.

Keywords—Coronavirus, Covid-19, Machine Learning, Face Mask Detection, Convolutional Neural Network, TensorFlow

I. INTRODUCTION

As countries resume COVID-19 lockdowns, governments and public health officials are taking precautions to protect us when we go out in public to contain the spread of the coronavirus and thereby contribute to public health. recommends face masks as an essential means of Regardless of discourses about medical resources and the diversity of masks, all countries mandate covering the nose and mouth in public. In order to mandate the use of face masks, it is imperative to develop some technology to force individuals to wear masks before going out in public. Very useful in public areas such as markets, shopping malls, etc. The method proposed here is performed in two steps. The first step is to train a face mask detector using transfer learning.

The second step is to use this trained face mask detector on an image or video of a person to check if they are wearing a mask. To create this model, we use a face mask dataset provided by Prajna Bhandary. It consists of about 1,376 images, of which 690 of his include people wearing masks and 686 of them without. Using these images, using Tensor Flow he builds a CNN model to detect if she is wearing a face mask using her webcam on her PC. Additionally, you can use your phone's camera to do the same.

II. LITERATURE SURVEY

All governments around the world are battling COVID-19, which is causing a serious fitness crisis. Therefore, the use of face masks can gradually reduce the excessive spread of this virus. Our venture product with face mask detection. Where you used Visible Studio to code in the past.

Used Python as the programming language for coding. For database reasons, we used the Prajna Bhandary data set. This data set has about 700 photo records of people wearing masks, some of which are not currently wearing masks.

The proposed method used here is a two-step process. Step 1 is to train the face mask detector using switch mastering.

III. RELATED WORK

Face detection methods detect faces from images containing multiple attributes. , facial recognition research requires expression recognition, face tracking, and pose estimation. For a single image, identifying faces from the image is a challenge. Face recognition is a difficult task because face sizes, shapes, colors, etc. are variable and not fixed. For opaque images obscured by something else that doesn't face the camera, etc., it becomes a tedious task. Creators of occlusive face recognition face two major challenges: dataset containing both masked and unmasked faces, and 2) exclusion of facial expressions in the covered regions. A dictionary trained on a Local Linear Embedding (LLE) algorithm and a very large pool of masked faces, synthesized mundane faces, and multiple misplaced facial expressions that can be restored , and over-strength faces Feature dominance can be used to mitigate the degree. According to the work reported in [11], computer vision convolutional neural networks (CNNs) have strict limits on the size of the input image.

The main problem in this task is to correctly identify faces in photos and check if they are covered by masks. The proposed approach also needs to recognize faces along with other facial features to perform surveillance tasks.

Two datasets were used for the current mask experiment and the moving mask experiment.

Dataset 1 contains 1376 photographs, of which 690 show a person wearing a mask, the remaining 686 of him showing a person without a mask. Most front face poses only include a face in a frame and the same type of mask, all white.

Dataset 2 on Kaggle has 853 photos with faces cleared with and without masks. Some face collections include head rotation, tilt, tilt, many faces in frames, and different masks with different colors.

IV. Incorporated Packages

A. TensorFlow

Machine learning frameworks such as TensorFlow are used to develop applications in various areas of computer science. They can be used to analyze sentiment, search and extract geographic information, perform text summarization, and identify errors in computer-assisted drug discovery.

In the proposed model, the entire CNN architecture consisting of multiple layers uses TensorFlow on the backend. It is also used to perform image processing.

B. Keras

Keras provides basic considerations and building blocks for designing and deploying high-repetition ML arrays. Take full advantage of TensorFlow's scalability and cross-platform capabilities. Layers and models are his two main data structures in Keras [19]. Keras is used to implement each layer of the CNN model. This is useful for building global models and converting class vectors to binary class matrices in data processing. C. OpenCV

OpenCV (Open Source Computer Vision Library), an open source machine vision and machine learning software library, provides face identification and recognition, object detection, motion grouping in recordings, progressive module tracking, and eye gesture tracking. , used for camera action tracking and red removal. - Eyes from flash photos, finding comparable photos from image databases, recognizing landscapes, setting markers to overlay more realistically, etc. [20]. The proposed solution uses these OpenCV functions to resize and color correct the data image.

V. Code

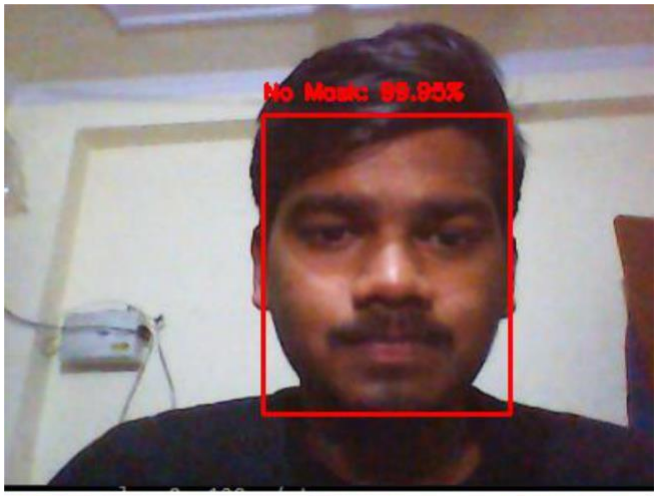
```
# python detect_mask_video.py

# 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 argparse
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, (300, 300),
                (104.0, 177.0, 123.0))

        # pass the blob through the network and obtain
the face detections
        faceNet.setInput(blob)
        detections = faceNet.forward()

        # 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, i, 2]

                # filter out weak detections by ensuring
the confidence is
                # greater than the minimum confidence
                if confidence > args["confidence"]:
                        # compute the (x, y)-
coordinates of the bounding box for
```

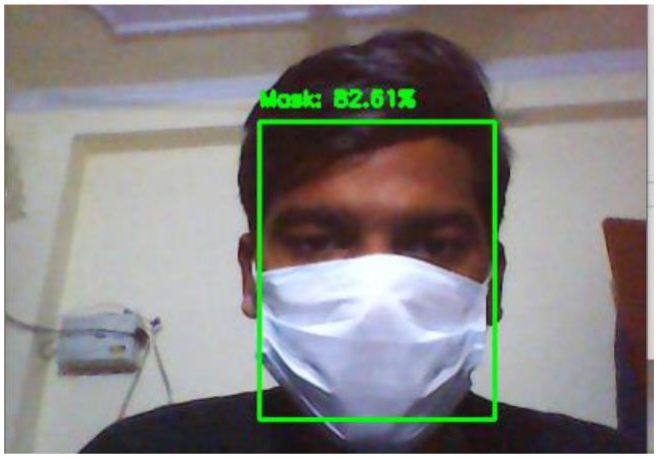
```
# the object
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                    (startX, startY, endX, endY) =
box.astype("int")

# 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]

if face.any():

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
# construct the argument parser and parse the
arguments ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str,
        default="face_detector",
        help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
        default="mask_detector.model",
        help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float,
default=0.5,
        help="minimum probability to filter weak
detections")
args = vars(ap.parse_args())

# load our serialized face detector model from
disk print("[INFO] loading face detector
model...") prototxtPath =
os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
        "res10_300x300_ssd_iter_140000.caffemodel"]
)
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
print("[INFO] loading face mask detector
model...") maskNet = load_model(args["model"])

# initialize the video stream and allow the camera
sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)

# 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, width=400)
# the bounding box and text
                label = "Mask" if mask > withoutMask
else "No Mask"
                color = (0, 255, 0) if label == "Mask"
else (0, 0, 255)

                # include the probability in the label
                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)

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

VI . Result and Output

Two datasets are used to train, validate, and test the model. According to dataset 1, the accuracy of this method is up to 95 The cost of inaccuracy is reduced by this improved precision, . Data set 2 is more adaptive than data set 1 as it contains many faces in the frame and different types of masks with different colors. As a result, the model achieves 94.58% accuracy on dataset 2. A comparison of training loss and validation loss for dataset 2 8. MaxPooling contributes significantly to this accuracy.

In addition to reducing the amount of parameters the model has to learn, it gives the internal representation some basic translation invariance.





VII. Conclusions

This publication first briefly described the motivation behind this work. The model's earned income and performance allocation were then presented. This strategy uses simple ML tools and simplified procedures to achieve relatively high accuracy. It has many uses. Given the Covid-19 issue, wearing a mask may become necessary soon. Customers are required to wear masks properly to use the services of multiple public service providers. Public health systems would greatly benefit from the models used. In the future, it may even be possible to detect whether someone is wearing a mask correctly.

VIII. References

[1] W.H.O., “Coronavirus disease 2019 (covid-19): situation report, 205”. 2020

[2] “CoronavirusDisease2019(COVID-19)– Symptoms”, CentersforDiseaseControland Prevention, 2020. [Online].

Available: https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html. 2020.

[3] “Coronavirus — Human Coronavirus Types — CDC”, Cdc.gov, 2020. [Online]. Available: https://www.cdc.gov/coronavirus/types.html. 2020.

[4] W.H.O., “Advice on the use of masks in the context of COVID-19: interim guidance”, 2020.

[5] M. Jiang, X. Fan and H. Yan, “RetinaMask: A Face Mask detector”,

arXiv.org, 2020. [Online]. Available: https://arxiv.org/ abs/2005.03950. 2020.