T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞİM TEKNOLOJİLERİ MÜHENDİSLİĞİ ANABİLİM DALI

PYTHON FOR ENGINEERING APPLICATION

REAL-TIME FACE MASK DETECTION WITH OPENCV PYTHON

MD AL AMIN HOSSAIN

218264001009

JANUARY-2023

KONYA

Contents

1. INTRODUCTION 1
2. OBJECTİVES AND APPLICATIONS 2
3. REQUIRED TOOLS 2
4. IMPLEMENTATION USING PYTHON 5
4.2. Code Line Description 6
5. EXPERIMENTAL OUTPUTS 9
6. CONCLUDING REMARK 10
7. REFERENCES.. 11

1. INTRODUCTION

Most of the world's population has been harmed as a result of the COVID-19 (Coronavirus) pandemic. COVID-19 is a respiratory condition that causes severe pneumonia in those who are affected by it [14]. The virus enters an airspace through coughing, sneezing, or direct contact with an infected person. It can also enter through salivation beads, respiratory droplets, or nasal droplets emitted during these actions [16]. Globally, the COVID-19 virus claims thousands of lives every day. The World Health Organization (WHO) reports that there were 258 million confirmed cases of COVID-19 infections and 5,148,221 fatalities globally as of November 22, 2021 [1].

In order to prevent the viral spread of disease, people should wear face masks and maintain social distancing. Whether people are wearing face masks or not, a real-time application that monitors people in public should be created using an effective and efficient computer vision strategy. Identifying the face is the first step in determining whether a mask is present on it. This separates the entire process into two phases: face detection and face mask detection. The identification of the face mask is significantly impacted by computer vision and image processing. Face detection has a variety of case applications, including face recognition and facial motions, where the latter must accurately depict the face [2].

The risks posed by face mask detection technologies still seem to be successfully handled as machine learning algorithms advance quickly. As it is used to identify faces in pictures and in real-time video streams, this breakthrough is becoming more and more significant. Face detection alone is a highly difficult task for the currently suggested methods of face mask detection, bu [1]. The analysis of events and video surveillance is a constant challenge in the development of more advanced facial detectors, following the outstanding results of current face detectors.

The uniqueness of this research in comparison to other publications is the suggestion of an effective and precise method for real-time videos. The suggested method offers real-time, accurate facemask wear detection, including whether it is worn properly or not [3]. To do this, both our own and publicly available datasets are combined to create a full dataset. This method produces outstanding object detection results and has the advantages of being quick and appropriate for edge devices. To determine whether people are abiding by the law and carrying identification, the suggested solution can be applied in real-world surveillance cameras in public spaces. The solution is simple to implement and requires few resources[2].

The remainder of the report paper is organized as follows: The purposes and applications of this study are discussed in Section 2. The required tools for this project were specified in Section 3. The use of the method for real-time face-mask detection is demonstrated in Section 4. The experimental results were shown and analyzed in Section 5, and the conclusions and recommendations are given in Section 6.

2. OBJECTİVES AND APPLICATIONS

The real-time face mask detection opencv python was developed using python detection opencv, during pandemic covıd-19, WHO has made wearing masks compulsory to protect against this deadly virus [4]. Its purpose is to split data into soft versus hard images at the initial level followed by a mask and non-mask classification at a later level through a facemask classifier.There are others goals and application of real-time face mask detection as follows:

- Created a facemask detector implemented in three phases to assist in precisely detecting the presence of a mask in real-time using images and video streams.
- Face mask detection refers to detecting whether a person is wearing a mask or not. In fact, the problem is reverse engineering of face detection where the face is detected using different machine learning algorithms for the purpose of security, authentication, and surveillance.
- Its purpose is to split data into soft versus hard images at the initial level followed by a mask and non-mask classification at a later level through a facemask classifier.
- With this system we can ensure our health risks in crowded places like markets, shopping malls, hospitals, offices, educational institutions, and every public place.

3. REQUIRED TOOLS

For implementing the real-time face mask detection using opencv python, we need two types of apparatus. One is a hardware-related apparatus, and the second is a software-related apparatus. such as,

- Hardware Apparatus:
    - A computer with windows version 7 or up
    - A good CPU or precessor
    - Internet connection
- Software Apparatus:
    - A software development environment like Anaconda, Pycharm, visual studio, Matlab etc.
    - Python IDE with opencv, Matplotlib, Numpy library
    - Specifically tensorflow==1.15.2, keras==2.3.1, imutils==0.5.3, numpy==1.18.2, opencvpython==4.2.0., matplotlib==3.2.1, scipy==1.4.1

I. Python

Python is a high-level and general-purpose programming language. Python is a programming language that may be used to create desktop gui apps, websites, web applications, mathematics, system scripting and so many. It has an autonomous memory management system and a dynamic type of system. It contains a large comprehensive library and supports different programming paradigms such as object-oriented, imperative, functional, and procedural. Python is predominantly a dynamic typed programming language that was released in 1991. Guido van Rossum invented it and after that the Python Software Foundation developed it. It has simple syntax that lets programmers to express concepts in fewer lines of code. Primarily it was built with code readability in mind [5].

II. OpenCV

Open source Computer Vision is basically a library, used for real-time Computer Vision. It supports a wide range of programming languages like Python, C, C++, Java etc and also supports different platforms including Windows, Linux, MacOS. By using it, one can process images and videos to identify objects, faces or even handwriting of a human. When it integrated with various libraries, such as Numpy, Python is capable of processing the OpenCV array structure for analysis. To identify image pattern and its various features we use vector space and perform mathematical operations on these features [6].

III. TensorFlow

TensorFlow is an open-source machine learning software that focuses on deep neural networks from start to finish. TensorFlow is a collection of libraries, tools, and community resources that are diverse and comprehensive. It enables programmers to construct and deploy cutting-edge machine learningbased applications. The Google Brain team first designed the TensorFlow python deep-learning library for internal usage. The open-source platform's application in R&D and manufacturing systems has increased since then. There are a few key principles in TensorFlow. Tensors are TensorFlow's fundamental building blocks. In figure 3.1 depicted the basic operational procedure of TensorFlow[2].

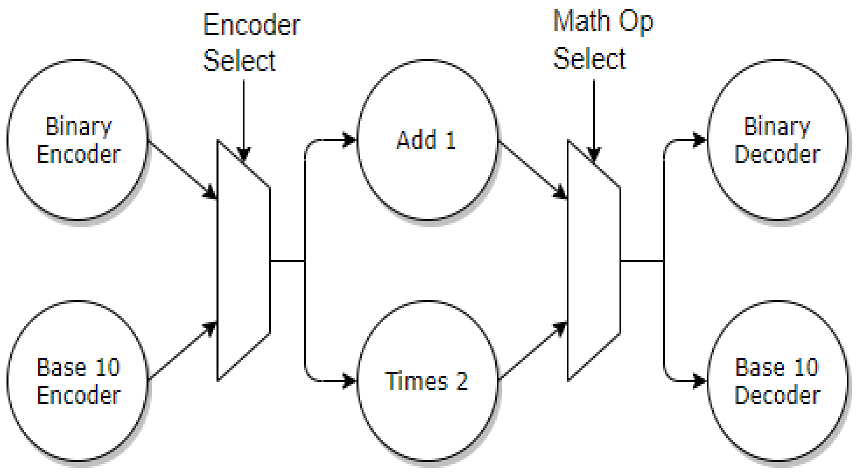


Figure 3.1: TensorFlow basic principle

IV. Keras

Google developed Keras, a high-level deep learning API for building neural networks. It is written in python and helps with neural network development. It is modular, quick, and simple to use. It was created by Google developer Francois Chollet. Low-level computation is not handled by Keras. Instead, it makes use of a library known as the "Backend". Keras provides the ability to swap between different back ends. TensorFlow, Theano, PlaidML, MXNet, and CNTK (Microsoft Cognitive Toolkit) are among the frameworks support by Keras. TensorFlow is the only one of these five frameworks that has accepted Keras as its official high-level API. Keras is a deep learning framework that is built on top of TensorFlow and has built-in modules for all neural network computations [1].

V. **NumPy and SciPy**

NumPy is the most important python package for scientific computing. It is a library that includes a multidimensional array object, derived objects (such as masked arrays and matrices), and a variety of routines for performing fast array operations, such as mathematical, logical, shape manipulation, sorting, selecting, basic linear algebra, basic statistical operations, random simulation, and more. Many other popular python packages, like as pandas and matplotlib, are compatible with NumPy. (NumPy 2022.). On the other hand, SciPy is another important python library for scientific and technical computing that is free and open source. It is a set of mathematical algorithms and utility functions based on the python numpy extension. It gives the user a lot of power by providing high-level commands and classes for manipulating and displaying data in an interactive python session. SciPy builds on NumPy, developers do not need to import NumPy if SciPy has already been imported [3].

VI. **Imutils and Matplotlib**

Imutils are a set of convenience functions for OpenCV and python 2.7 and python 3 that make basic image processing functions including translation, rotation, scaling, skeletonization, and presenting matplotlib pictures easier. Matplotlib is an important python visualization library for 2D array plots. Matplotlib is a multi-platform data visualization package based on numpy arrays and intended to operate with scipy stack. It was first introduced in 2022 by John Hunter. One of the most important advantages of visualization is that it provides visual access to large volumes of data in simple images. Matplotlib has a variety of plots such as line, bar, scatter, histogram, and so on. It is a cross-platform package that provides a variety of tools for creating in python from data stored in lists or arrays, and it is one of the most capable libraries available in python [2].

4. IMPLEMENTATION USING PYTHON

A Face Mask Detection model is the subject of this project. A model has been developed to determine whether people are wearing masks. The primary camera on the computer was used in this system, and the video was sent as input to the model that was deployed. This system is built using OpenCV libraries, as well as images that are supplied into the system during the learning process.

4.2. Code Line Description

In this section, we explained the code line of real-time face mask detection utilizing the python OpenCV method. First, we have to install the OpenCV library which can be easily installed by using the pip command: ***pip install opencv-python*** and other libraries using pip command. The python programming language, as well as machine learning and deep learning methods were used to identify the face masks. Inside the project path, several required libraries were installed. Libraries were installed using the cmd-command prompt. Packages like tensorflow, numpy, imutils, keras, opencv, scipy, and matplotlib were imported after the installation. These packages take care of their own functions as needed within the application

To get started first we have to import all the libraries and packages required as follows.

```
# import libraries
# 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
```

Now we can grab the dimension of a frame and create the blob. Blob analysis is used on the acquired image datasets to analyse each face shapes, features etc. Then pass the bolb through the network to obtain face detection.

```
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)
```

Now we initialize our list of faces for their corresponding location and the list of predictions from our face mask network. Then create a loop to follow the detected faces.

```
# 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]
```

In this stage filter out weak detections by ensuring the confidence that is grater than the minimum confidence.

```
# filter out weak detections 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))
```

This parts of code convert BGR to RGB frame format using face extracting ROI method and resize the image frame.

```
# 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)
         locs.append((startX, startY, endX, endY))
```

```
# only make a predictions if at least one face was detected
   if len(faces) > 0:
# 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)
```

This section of code detect the face from the disk loader and most important things initialize the video stream frame for detect the face.

```
# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath =
r"face_detector\res10_300x300_ssd_iter_140000.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, width=400)
```

This parts detect faces in the frame and evaluate the weather if persons are wearing a face mask or not.

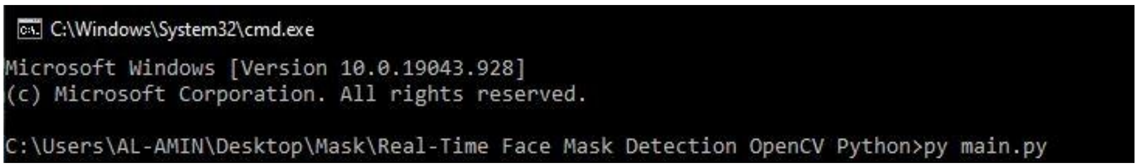
```
     # detect faces in the frame and determine if they are
wearing a face mask or not
   (locs, preds) = detect_and_predict_mask(frame, faceNet,
maskNet)
   # loop over the detected face locations and their
corresponding
   # locations
   for (box, pred) in zip(locs, preds):
      # unpack the bounding box and predictions
      (startX, startY, endX, endY) = box
      (mask, withoutMask) = pred
```

If persons are wearing or not a mask this line of code displays the label and bounding box rectangle on the video screen.

```
 # 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()
```

5. EXPERIMENTAL OUTPUTS

To get output results and open the video stream we have to go file folder of source code and open the file folder with command prompt ***cmd*** and write the command ***py main.py*** as follows and finally get the output like figure 5.1.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\AL-AMIN\Desktop\Mask\Real-Time Face Mask Detection OpenCV Python>py main.py
```

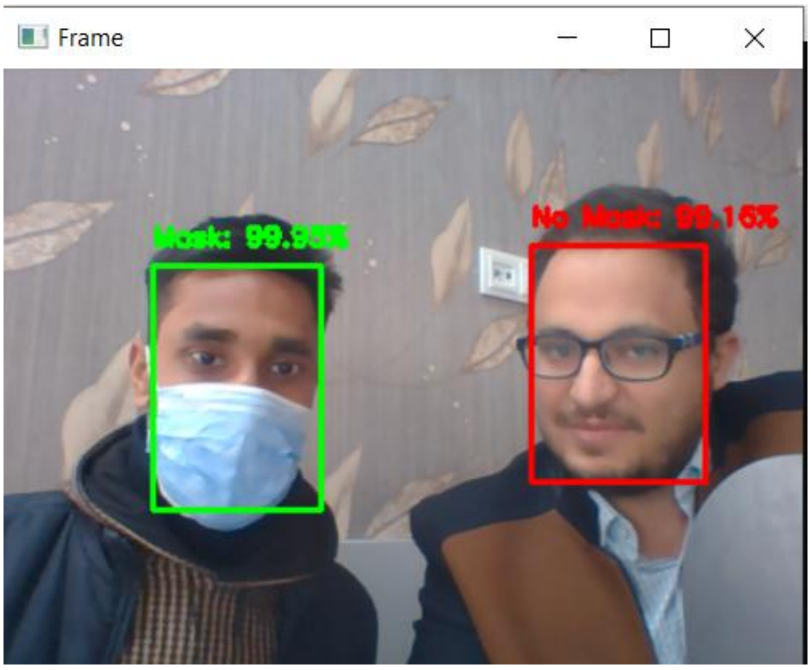


Figure 5.1: Output screenshoot image of face mask detection

6. CONCLUDING REMARK

This work offered a deep-learning-based method for detecting masks covering faces in public settings, hence limiting the spread of the coronavirus among nearby communities. The ensemble method greatly increases detection speed while also assisting in achieving high accuracy. Additionally, rigorous testing on an unbias dataset combined with transfer learning on pre-trained models produced a dependable and affordable solution. The results demonstrate the app's viability in real-world situations, hence assisting in the containment of the pandemic spread.

These days, masks are becoming more and more popular. Masks are one of the few means to defend against the coronavirus in the absence of immunization, and they are crucial for safeguarding people's health from respiratory infections. This project may be deployed in a variety of public locations, such as airports, railway stations, offices, schools, and public spaces, and it can be integrated with embedded technology to protect human safety. To increase accuracy, future work can be expanded to address additional mask-wearing problems. The created approach can be used with biometric surveillance equipment, particularly in polluted sectors with face masks and facial feature detection.

7. REFERENCES

[1] W. Boulila, A. Alzahem, A. Almoudi, M. Afifi, I. Alturki, and M. Driss, "A Deep Learning-based Approach for Real-time Facemask Detection," *Proc. - 20th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2021*, no. Dl, pp. 1478–1481, 2021, doi: 10.1109/ICMLA52953.2021.00238.

[2] A. K. Suzon, "FACE MASK DETECTION IN REAL-TIME USING PYTHON Thesis CENTRIA UNIVERSITY OF APPLIED SCIENCES Degree Programme," no. April, 2022.

[3] H. Goyal, K. Sidana, C. Singh, A. Jain, and S. Jindal, "A real time face mask detection system using convolutional neural network," *Multimed. Tools Appl.*, vol. 81, no. 11, pp. 14999–15015, 2022, doi: 10.1007/s11042-022-12166-x.

[4] M. A. S. Ai *et al.*, "Real-Time Facemask Detection for Preventing COVID-19 Spread Using Transfer Learning Based Deep Neural Network," *Electron.*, vol. 11, no. 14, 2022, doi: 10.3390/electronics11142250.

[5] Angel Jude suarez, "Real-Time Face Mask Detection OpenCV Python Source Code," *itsource*, 2021. https://itsourcecode.com/free-projects/python-projects/real-time-face-mask-detection-opencv-python-source-code/

[6] Muhammad Nadeem Ali, "Face and Face Parts Detection in Image Processing," *Lahore Garrison Univ. Res. J. Comput. Sci. Inf. Technol.*, vol. 1, no. 1, pp. 62–68, 2017, doi: 10.54692/lgurjcsit.2017.01017.