

Face Mask Detection Using CNN and TensorFlow

Review-2

for

Artificial Intelligence (CSE3013)

By

Devansh Jain	19BCI0173
Anshul Ola	19BCI0183
Chaitanya Gupta	19BCI0198

Slot: F1+TF1

Submitted to: Professor Nitin Singh Rajput
(SCOPE)

Keywords:

Object Detection; Object Tracking; COVID-19; Face Masks; Safety Improvement; TensorFlow; OpenCV; Deep Learning

1. Introduction:

After the global outbreak of the COVID-19 pandemic, there arises a severe want of protection mechanisms, masks being the first one. COVID-19 has continuing to be the reason for plight for several lives and businesses even in 2020. As the world recovers from the pandemic and plans to come to a state of normalcy, there's a wave of hysteria among all people, particularly those that will resume in-person activity. Studies have proven that sporting a mask considerably reduces the danger of infective agent transmission moreover as provides a way of protection. However, it's not possible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This can facilitate track safety violations, promote the utilization of face masks, and guarantee a secure operating surroundings.

2. Objective:

More than five million people were infected by COVID-19 in less than 6 months across 188 countries. The virus spreads through close contact and in crowded and over-crowded areas. The coronavirus epidemic has given rise to an extraordinary degree of worldwide scientific cooperation. Artificial Intelligence (AI) based on Machine learning and Deep Learning can help to fight Covid-19 in many ways. Mask Detection APIs can be integrated with pre-installed CCTV cameras. This can facilitate track safety violations, promote the utilization of face masks, and guarantee a secure operating surroundings.

3. Problem Statement:

- a) Training programme for facial detection.
- b) Building a detection system for images.
- c) Building a detection system for live video feed.
- d) Detection of various mask types i.e. Medical mask, Regular mask and how is it worn by people.

4. Contributions:

Chaitanya Gupta: Model Training, Pre-processing, Research Papers

Anshul Ola: Research Papers, Pre-processing, Image and Video detection

Devansh Jain: Dataset Collection, Image and Video detection

5. Literature Survey:

[1] ARTICLE 1 (Big Data Analytics and Computational Intelligence (ICBDAC) (pp.23-28).IEEE.)

In the ongoing past, arrangement of Convolutional Neural Networks (CNN) has prompted enormous achievement in many pattern recognition undertakings. This is chiefly because of the very idea of CNN, that is its capacity to work along these lines to that of the visual system of the human brain. A standout amongst the most energizing utilization of pattern recognition which is the point of convergence of the proposed work is face recognition. Along these lines, the models sent in the proposed work are fit for exploiting spatially restricted connections in a facial picture to create reliably high accuracy.

[2] ARTICLE 2 (Video-based expression recognition using CNN-RNN and C3D hybrid networks)

In this paper, the centre module of this framework is a hybrid network that consolidates recurrent neural network (RNN) and 3D convolutional networks (C3D) in a late-fusion fashion. RNN and C3D encode appearance and motion data in various ways. In particular, RNN takes appearance features extricated by convolutional neural network (CNN) over singular video outlines as info and encodes motion later, while C3D models appearance and motion of video all the while. Broad analyses demonstrate that joining RNN and C3D

together can improve video-based expression recognition recognizably.

[3]

RESEARCH PAPER 1 :- REAL TIME FACE DETECTION AND TRACKING USING OPENCV

https://www.digitalxplore.org/up_proc/pdf/43-1392225813137-140.pdf

Human face localization and detection is often the first step in applications such as video surveillance, human computer interface, face recognition and image database management. Locating and tracking human faces is a prerequisite for face recognition and/or facial expressions analysis, although it is often assumed that a normalized face image is available. Face detection is defined as the procedure has many applications like face tracking, pose estimation or compression. Face detection is used in biometrics, often as a part of (or together with) a facial recognition system. It is also used in video surveillance, human computer interface and image database management. Some recent digital cameras use face detection for autofocus. Different methods and algorithms of face detection have been reviewed in this paper. The choice of a face detection method in any study should be based on the particular demands of the application. None of the current methods is the universal best for all applications. In order to be successful a face detection algorithm must possess two key features, accuracy and speed.

[4]

RESEARCH PAPER 2 :- Facial Recognition using OpenCV

http://jmeds.eu/index.php/jmeds/article/view/Facial_Recognition_using_OpenCV/pdf

The goal of this article was to provide an easier human-machine interaction routine when user authentication is needed through face detection and recognition. With the aid of a regular web camera, a machine is able to detect and recognize a person's face; a custom login screen

with the ability to filter user access based on the users' facial features will be developed. The objectives of this thesis are to provide a set of detection algorithms that can be later packaged in an easily portable framework amongst the different processor architectures we see in machines (computers) today. These algorithms must provide at least a 95% successful recognition rate, out of which less than 3% of the detected faces are false positives.

6. Design:

We'll use a two-step process to implement our idea.

Then the dataset will be obtained from various online sources and then fed into the trainer program.

The face mask detector will be implemented using a Python.

Various libraries will be imported for a better result and robustness.

OpenCV for real-time computer vision.

Tensorflow for training neural networks.

Then we'll proceed to implement two more additional Python programs used to:

1. Detect COVID-19 face masks in images.
2. Detect face masks in real-time video feed.

7. Timeline:

<u>Modules</u>	<u>Time</u>
PHASE 1 :- TRAINING 1. Finding Appropriate Face Mask Dataset - 5 days 2. Creating a Code to Load Face Mask Dataset Into our Program - 8 days 3. Training Face Mask Classifier with TensorFlow - 4 days 4. Create A program for Storing Face Mask Classifier to Disk - 4 days	3 weeks




PHASE 2 :- DEPLOYMENT 1. Loading Face Mask Classifier From Disk - 4 days 2. Create a Video Stream using OpenCV To Detect Faces in Real Time - 10 days 3. Creating a Code to extract Each Face Data - 7 days 4. Create a Code to Apply Face Mask Classifier to each Face to determine 'Mask' or 'No Mask' - 10 days 5. Checking Results - 2 days	4 weeks
PHASE 3 :- TESTING AND MODIFICATION	2 weeks

8. Implementation Details:

Dataset: The dataset was obtained from Kaggle which hosts a lot of public datasets which provide a lot of images of people with masks and without masks to train our program properly.

Python scripts:

We have used three python scripts for a better work flow and easier code review and editing.

 image.py	23-05-2021 05:58 PM	Python Source File	4 KB
 trainer.py	23-05-2021 05:58 PM	Python Source File	6 KB
 video.py	23-05-2021 05:58 PM	Python Source File	5 KB

Trainer script:


```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import os
```

We have imported tensorflow.keras for data augmentation.

Imutils for listing images of the dataset.

matplotlib for plotting graphs and curves.

Numpy for storing image RGB values.

Argparse for passing command line arguments.

```
INIT_LR = 1e-4
EPOCHS = 20
BS = 32
```

Setting learning rate, training epochs and batch size

```
data = np.array(data, dtype="float32")
labels = np.array(labels)
```

Converting image data to NumPy array

```

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

model = Model(inputs=baseModel.input, outputs=headModel)

for layer in baseModel.layers:
    layer.trainable = False

```

Constructing model

```

print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
              metrics=["accuracy"])

```

```

print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

```

Compiling model and making predictions

```

print("[INFO] saving mask detector model...")
model.save(args["model"], save_format="h5")

```

```
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

Saving model to disk and plotting Training Loss and accuracy

Image Script:

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import argparse
import cv2
import os
```

```
print("[INFO] loading face mask detector model...")
model = load_model(args["model"])
```

```
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]
```

Loading trained model from disk and loading image data

```

for i in range(0, detections.shape[2]):
    confidence = detections[0, 0, i, 2]

    if confidence > args["confidence"]:

        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")
        (startX, startY) = (max(0, startX), max(0, startY))
        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
        face = image[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)
        face = np.expand_dims(face, axis=0)

        (mask, withoutMask) = model.predict(face)[0]

        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

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

        cv2.putText(image, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)

```

Predicting mask or no mask based on trained model.

```

cv2.imshow("Output", image)
cv2.waitKey(0)

```

Showing output

Video script:

```
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):
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),
                                   (104.0, 177.0, 123.0))

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

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

    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]
```

Defining a function to take video input

```

if confidence > args["confidence"]:
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
    (startX, startY, endX, endY) = box.astype("int")
    (startX, startY) = (max(0, startX), max(0, startY))
    (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

    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)

    faces.append(face)
    locs.append((startX, startY, endX, endY))

```

Predicting mask or no mask based on model

```

while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
        cv2.putText(frame, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

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

cv2.destroyAllWindows()
vs.stop()

```

Showing output video stream with prediction.

9. Result Analysis:

Running trainer.py

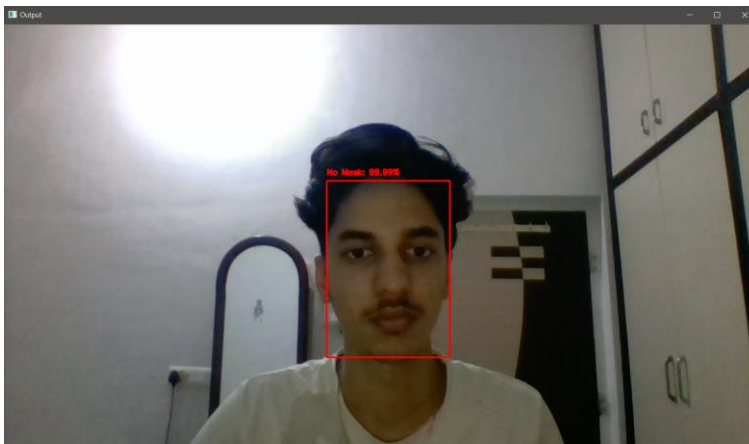
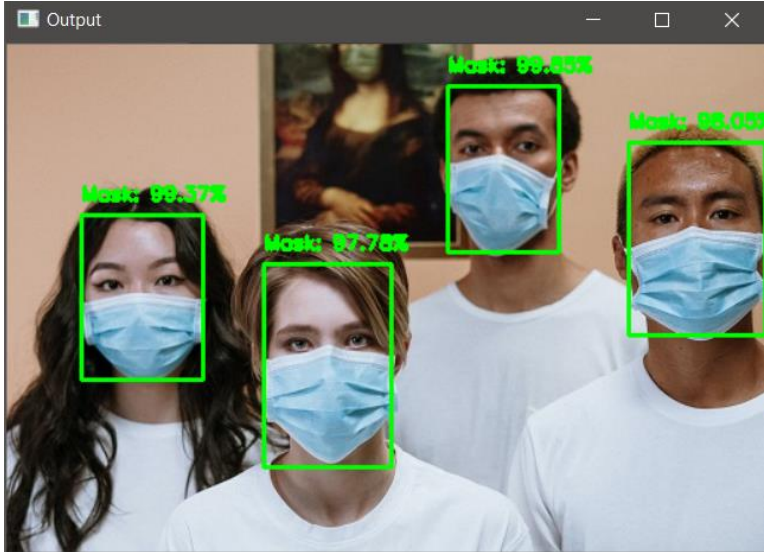
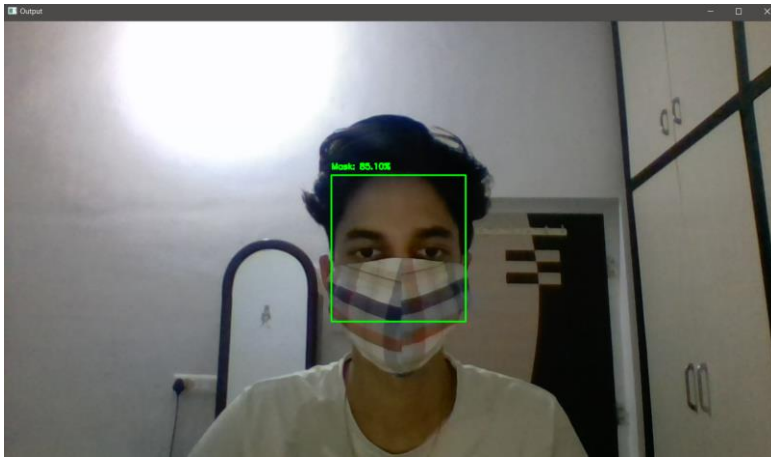
```
C:\Windows\System32\cmd.exe - python trainer.py --dataset dataset
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

E:\Anshul\4th sem\Artificial Intel\AI Project\Custom mask detector>python trainer.py --dataset dataset
[INFO] loading images...
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224].
2021-05-23 21:00:12.117044: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA devices, tf_
2021-05-23 21:00:12.119286: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is
ormance-critical operations: AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[INFO] compiling model...
[INFO] training head...
2021-05-23 21:00:14.196417: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the ML
Epoch 1/20
17/34 [=====>.....] - ETA: 11s - loss: 0.8449 - accuracy: 0.5818_
```

Running image.py

```
Select C:\Windows\System32\cmd.exe - python image.py --image sample/1.jpg
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

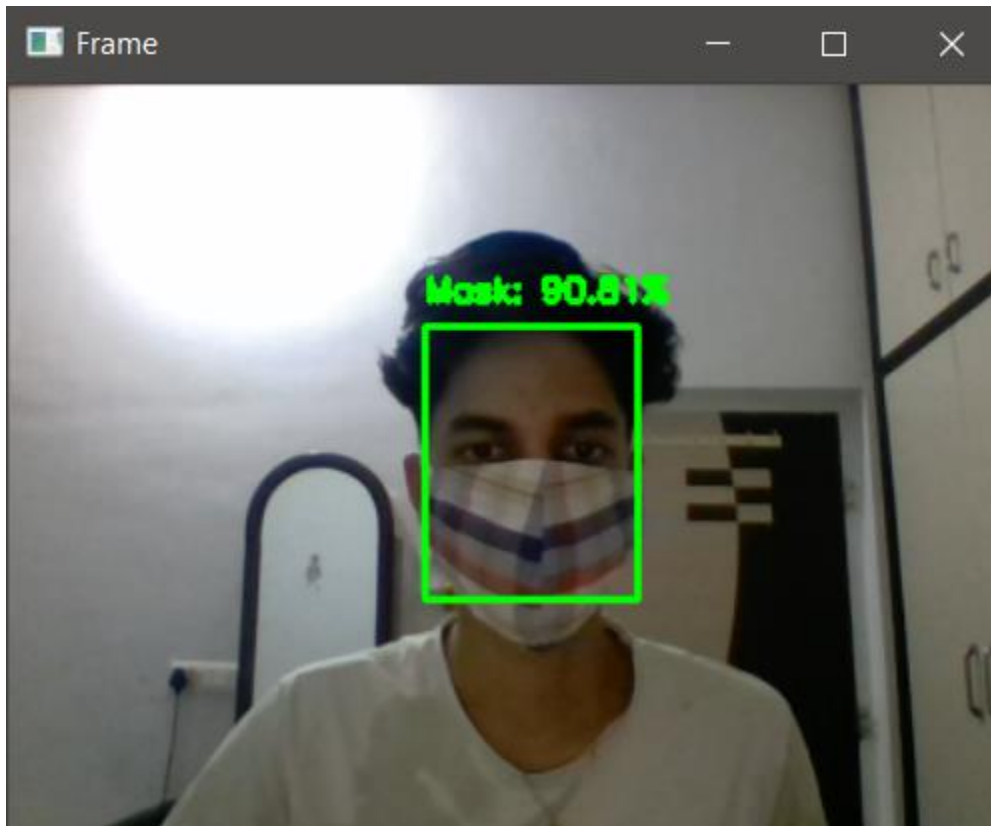
E:\Anshul\4th sem\Artificial Intel\AI Project\Custom mask detector>python image.py --image sample/1.jpg
[INFO] loading face detector model...
[INFO] loading face mask detector model...
2021-05-23 21:03:25.292793: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA devices, tf_xla_enable_xla_
devices not set
2021-05-23 21:03:25.293247: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[INFO] computing face detections...
2021-05-23 21:03:26.707483: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimizatio
n passes are enabled (registered 2)
```



Running Video.py

```
C:\Windows\System32\cmd.exe - python video.py
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

E:\Anshul\4th sem\Artificial Intel\AI Project\Custom mask detector>python video.py
[INFO] loading face detector model...
[INFO] loading face mask detector model...
2021-05-23 21:07:18.359513: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not creating XLA devices, tf_xla_enable_xla_devices not set
2021-05-23 21:07:18.359945: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
ormance-critical operations: AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[INFO] starting video stream...
2021-05-23 21:07:33.388040: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
```



10. Conclusion/Remark:

As we can clearly see from the results that the program works really well in detecting a person with mask or no mask.

It can also detect multiple people at the same time and works really well in live video stream too.

This can be deployed on airports, hospitals, workplaces, shopping mall entrances, etc.

11. References:

1. Vinay A Reddy, D.N.Sharma, A.C.Daksha, S.Bhargav, N.S.Kiran, Natrajan,(2017,March). G-CNN and F-CNN: Two CNN based architectures for face recognition.In 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC) (pp.23-28).IEEE.
2. Fan Y,Lu X,Li D,& Liu Y. (2016,October). Video-based expression recognition using CNN-RNN and C3D hybrid networks. In Proceedings of the 18th ACM International Conference on Multimodal Interaction (pp.445-450) .ACM.