LIGHTWEIGHT AI BASED FACE-MASK DETECTION SOFTWARE IN PYTHON

J Component Project

CSE3013 - Artificial Intelligence

SUBMITTED BY

1. Arshdeep Singh Bhatia (19BCB0086)
2. Kartik Nahta (19BCE2009)
3. Shivam Singhal (19BCE2112)
4. Arpit Khandelwal (19BCE0888)

SUBMITTED TO

Prof. Lokesh Kumar R

(Associate Professor Grade 1)

School of Computer Science and Engineering

Vellore Institute of Technology

Vellore

Fall Semester 2021

ACKNOWLEDGEMENTS

We would like to thank the SCOPE school for providing us this opportunity to learn and apply our skills in the world of Artificial Intelligence.
In addition to that we would like to thank Prof. Lokesh Kumar R. for guiding us throughout this journey and helping us to develop this application project. Lastly we would like to thank VIT for providing students with the necessary resources to accomplish such endeavours.

1. Arshdeep Singh Bhatia (19BCB0086)
2. Kartik Nahta (19BCE2009)
3. Arpit Khandelwal (19BCE0888)
4. Shivam Singhal (19BCE2112)

Vellore Campus

December 2021

Fall Semester 2021-2022

ABSTRACT

With the pandemic having such a widespread impact on the globe there is a great need for all of us to wear masks.

The latest forecast from the Institute of Health Metrics and Evaluation suggests that 33,000 deaths could be avoided by October 1 if 95 percent of people wore masks in public.However, intentionally or unintentionally we seem to remove our masks while entering public places.

We aim to make a tool which will be able to detect if a person is wearing a mask properly covering the vital areas of nose and mouth. Then based on the results of the AI we will be able to trigger further actions such as open the door, or send an alert message saying that the mask isn’t worn properly or not worn at all.

LIST OF FIGURES

LIST OF ABBREVIATIONS

- API - Application Programming Interface
- GPU - Graphics Processing Unit
- CCTV - Closed-Circuit Television surveillance

TABLE OF CONTENTS

INTRODUCTION

1.1 SYSTEM OVERVIEW

The overview of the system shall be covered in this section.The system software we have developed is developed in python programming language.We have also used python libraries like openCV and mobilnetV2 architecture. These libraries enable us to perform image processing and analysis features that further help in the prediction of whether the face in the video is covered by a mask or not. The system refers to a model which is trained using about 4000 images classified as with mask and without mask respectively.Upon model training a particular plot is generated which helps us understand the accuracy of the model and what levels of improvements can be made.

Now that all the prerequisites are satisfied the python system is started which takes in the video stream and compares it with the trained model and determines the percentage of the face covered with the mask. Above a threshold value of 50 % the system shall conclude a mask is worn and the frame surrounding the subject will be green else the system shall show as red and certain actions can be taken based on this.

1.2 OBJECTIVE

- Through this proposed Face mask recognition software, we aim to track and precisely compute the percentage of face covered by mask and classify it as safe or unsafe, among some other options.
- Then based on the results of the AI we will be able to trigger further actions such as open the door, or send an alert message saying that the mask either isn’t worn properly or not worn at all.
- We are going to use video streams in real-time and will use photos as the data set.
- The tool will be able to generate the output(with mask/without mask) without the need of storage of the video stream.

1.3 PROBLEM STATEMENT

- Over 40% of people in a democratic don't wear masks. 70% of those who wear masks wear it incorrectly and mouth and nose are exposed.
- It is not possible to make guards do manual surveillance as it poses a risk and is time-consuming.
- Surveillance by security guards in all regions is not feasible. Employing so many guards is not an issue but managing each security guard is uneconomic as well as includes excess manpower and hierarchical systems.
- Guards may not miss out on people without masks but it is a possibility that they may not consider vital spots that masks should properly cover.

1.4 LIMITATIONS

- This system may need more training of dataset as it can definitely work better and longer than humans can but its accuracy may not be as good as that of human monitoring system
- The system can only analyse subjects if the face is visible in the videostream and thus a particular hardware may limit the system's performance.
- The system only detects people wearing masks and may fail to detect in case facial features like eyes and forehead are covered.

SYSTEM ANALYSIS

2.1 PROPOSED SYSTEM

The work we propose to do refers to a research paper that proposes a real-time face mask detection system by applying computer vision and machine learning concepts like convolution neural networks (CNN) and refined Mobile Net V2 architecture to ease the deployment of the proposed model in embedded devices with limited computational capability.

The data set utilized here contains about 12000 images (6000 with mask 6000 without mask).The model is trained using Adam optimizer algorithm which is best suited for deep learning models and is built using keras , TensorFlow and openCV.

The proposed model touches 99% accuracy under various training to testing ratios like 70% training and 30% testing 50% training and 50% testing etc. Precision , recall,F1 score support are calculated for all trials.

This means that the system is computationally effective and could potentially be used in places like railway stations , airports , ATMs and other public places to detect people not wearing face masks and ensure the safety to certain extent in this pandemic.

If our system is able to detect that the person does not wear a mask a small speaker or other such hardware devices could be used and an alert message could be sent or an alert audio could be played in front of the person to alert or notify the person to wear the mask.

Algorithm -

Train Mask:

initialize the initial learning rate, number of epochs to train for, # and batch size

grab the list of images in our dataset directory, then initialize the list of data (i.e., images) and class images

perform one-hot encoding on the labels

construct the training image generator for data augmentation

load the MobileNetV2 network, ensuring the head FC layer sets are left off

construct the head of the model that will be placed on top of the base model

place the head FC model on top of the base model (this will become the actual model we will train)

2.1.1 Benefits of Proposed System

- Light-Weight And Quick Face Detection And Analysis.
- Can Be Integrated With Security Cameras And Microphones To Give Alerts.
- Real-Time Monitoring And Multi Face Analysis

REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

- Configured GPU (optional)
- Webcam

3.2 SOFTWARE REQUIREMENTS

Development environment

- Laptop with following software
    - Python
    - Tensorflow
    - Keras
    - Numpy
    - MobilenetV2
    - OpenCV
    - Frontal Face Classifier API

SYSTEM DESIGN SPECIFICATION

4.1 SYSTEM ARCHITECTURE

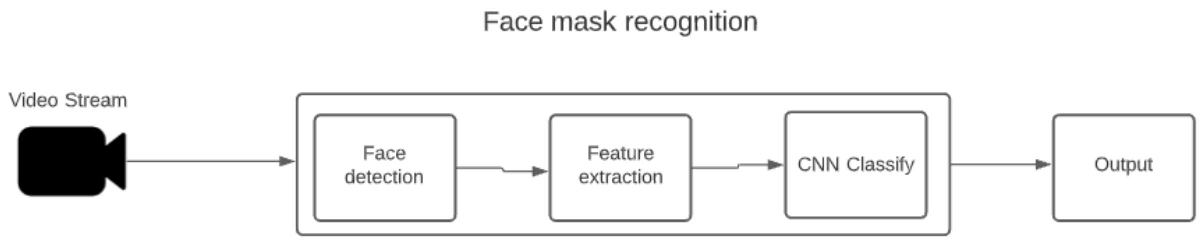


Fig 4.1.1: High level design

4.2 DETAILED DESIGN

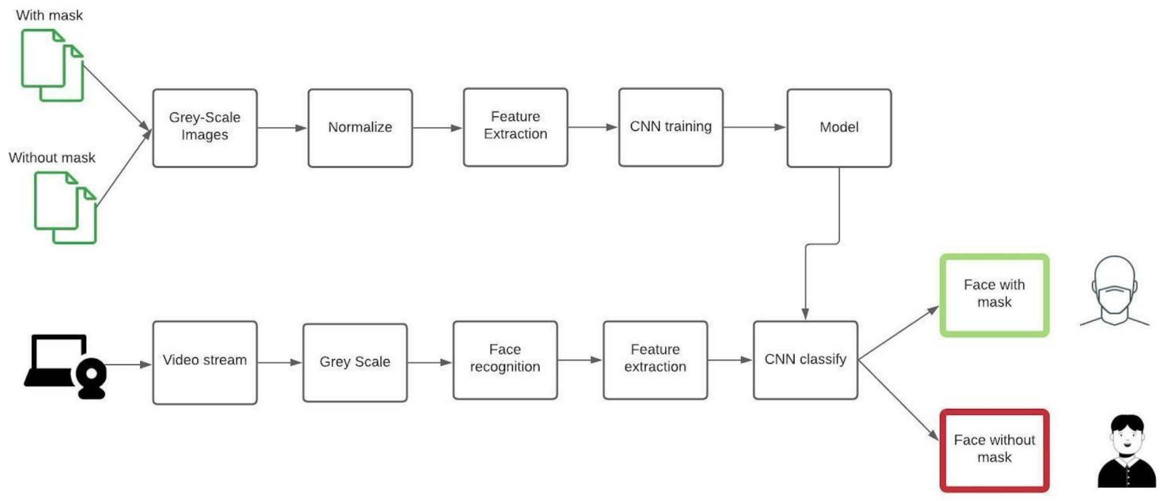


Fig 4.1.2: Detailed Modular design

SYSTEM IMPLEMENTATION

List of modules

1. DataSet
    1. With mask
    2. Without mask
2. Face Detection essentials
    1. Contains data models that capture and identify important features of a human face
3. Data Preprocessing
4. Train model
5. Model
6. Detect face mask

5.1 MODULE DESCRIPTION

1. Dataset

Our dataset contains two directories:

a. With Mask
b. Without Mask

Both these directories contain 500 pngs and jpegs of respective categories.

The with mask directory has photos of people wearing mask correctly, which fulfils the following requirement:

a. Nose is correctly and fully covered with mask

b. Mouth is covered with mask

c. Mask covers whole nose-mouth-chin region and is worn over the ears/neck

The without mask directory has normal images of people from different countries and ethnicities.

Steps:

2. Data Preprocessing

a. initialize the initial learning rate, number of epochs to train for and batch size
b. grab the list of images in our dataset directory, then initialize the list of data (i.e., images) and class images
c. perform one-hot encoding on the labels

3. Face Detection essentials

d. construct the training image generator for data augmentation
e. load the MobileNetV2 network, ensuring the head FC layer sets are left off

4. Train model

f. construct the head of the model that will be placed on top of the base model
g. place the head FC model on top of the base model (this will be the actual model we will train)
h. loop over all layers in the base model and freeze them so they will not be updated during the first training process

5. Model

i. compile our model
j. train the head of the network
k. make predictions on the testing set
l. for each image in the testing set we need to find the index of the label with corresponding largest predicted probability
m. serialize the model to disk
n. plot the training loss and accuracy

6. Detect face mask

a. grab the dimensions of the frame and then construct a blob from it

b. initialize our list of faces, their corresponding locations,and the list of predictions from our face mask network

c. Loop over the detections and extract the confidence (i.e., probability) associated with the detection

d. compute the (x, y)-coordinates of the bounding box for the object and ensure the bounding boxes fall within the dimensions of the frame

e. add the face and bounding boxes to their respective lists

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

g. load our serialized face detector and mask detector model from disk

h. initialize the video stream and loop over the frames from the video stream

i. grab the frame from the threaded video stream and resize it to have a maximum width of 400 pixels

j. detect faces in the frame and determine if they are wearing a face mask or not

k. loop over the detected face locations and their corresponding locations

l. determine the class label and color we'll use to draw the bounding box and text

m. include the probability in the label and display the label and bounding box rectangle on the output frame

n. show the output frame

CONCLUSION AND FUTURE ENHANCEMENTS

We aim to make a tool which will be able to detect if a person is wearing a mask properly covering the vital areas of nose and mouth. Then based of the results of the AI we will be able to trigger further actions such as open the door, or send an alert message saying that the either mask isn't worn properly or not worn at all

This realtime mask detection module can be further used in the following cases:

a. Public places like malls, railway stations, airports, parks and anywhere with CCTVs and raspberry Pi with a monitor to show who is not wearing a mask properly.
b. At corporations, factories and places where people may want to remove mask but pandemic isn't over yet
c. This technology can also be used with thermal screening modules with required IoT devices to track who may possess the Corona virus and segregate them from public for further inspection to prevent spread of virus.
d. The face mask detection module also has capabilities to capture the faces and their mask on recorded videos, so the healthcare services can use this to track epicenters and further track the people who originally were infected and didn't wear their masks properly which led to spike in covid cases in particular regions/neighbourhoods.

APPENDICES

7.1 APPENDIX 1 - SAMPLE SOURCE CODE

detect_mask_video:

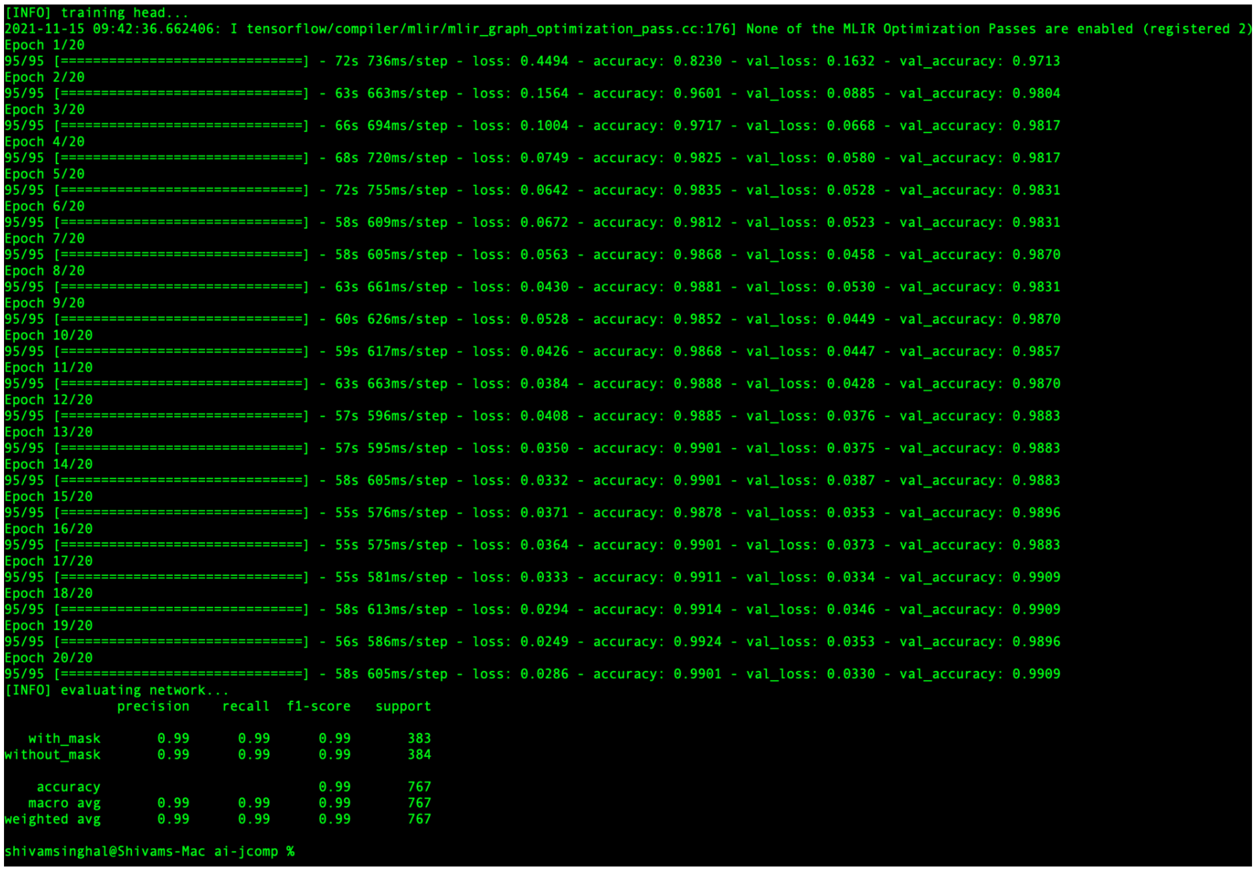
```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os

#Devs
#Shivam Singhal
#Kartik Nahta
#Arshdeep Bhatia
#Arpit Khandelwal


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, i, 2]

                # 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))

                        # 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)

# 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

# vs = VideoStream(src=0).start()
# vs = VideoStream(src='video.mp4').start()

a = int(input("Enter 0/1 (Recorded Video/ Live Stream ::"))
print("[INFO] starting video stream...")
if(a==1):
        # print("[INFO] starting video stream...")
  vs = VideoStream(src=0).start()
else:
        # print("[INFO] starting video stream...")
  vs = VideoStream(src='video.mp4').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)

        # 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

                # 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)

                # 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()
```

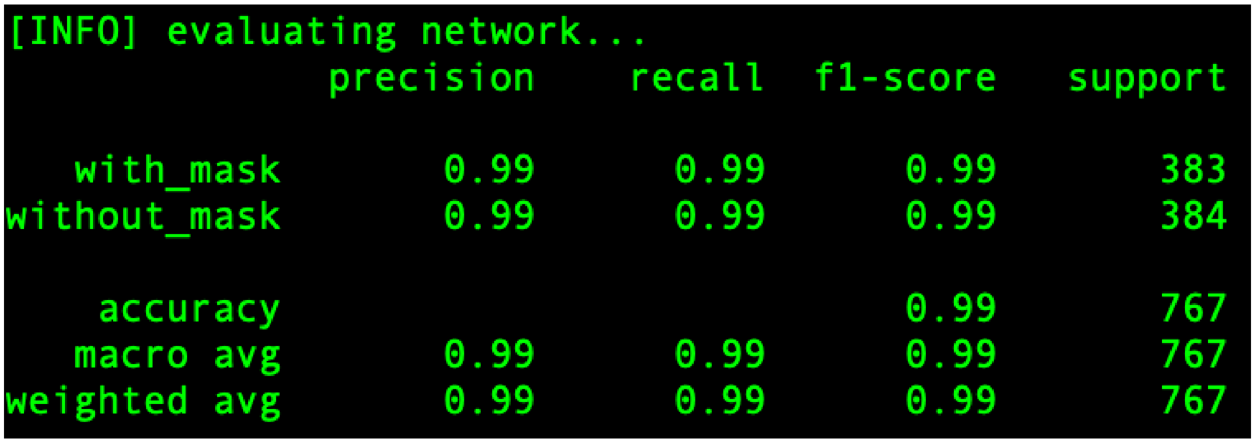
7.2 APPENDIX 2 - SCREENSHOTS /OUTPUTs

TRAINING



```
[INFO] training head...
2021-11-15 09:42:36.662406: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/20
95/95 [==============================] - 72s 736ms/step - loss: 0.4494 - accuracy: 0.8230 - val_loss: 0.1632 - val_accuracy: 0.9713
Epoch 2/20
95/95 [==============================] - 63s 663ms/step - loss: 0.1564 - accuracy: 0.9601 - val_loss: 0.0885 - val_accuracy: 0.9804
Epoch 3/20
95/95 [==============================] - 66s 694ms/step - loss: 0.1004 - accuracy: 0.9717 - val_loss: 0.0668 - val_accuracy: 0.9817
Epoch 4/20
95/95 [==============================] - 68s 720ms/step - loss: 0.0749 - accuracy: 0.9825 - val_loss: 0.0580 - val_accuracy: 0.9817
Epoch 5/20
95/95 [==============================] - 72s 755ms/step - loss: 0.0642 - accuracy: 0.9835 - val_loss: 0.0528 - val_accuracy: 0.9831
Epoch 6/20
95/95 [==============================] - 58s 609ms/step - loss: 0.0672 - accuracy: 0.9812 - val_loss: 0.0523 - val_accuracy: 0.9831
Epoch 7/20
95/95 [==============================] - 58s 605ms/step - loss: 0.0563 - accuracy: 0.9868 - val_loss: 0.0458 - val_accuracy: 0.9870
Epoch 8/20
95/95 [==============================] - 63s 661ms/step - loss: 0.0430 - accuracy: 0.9881 - val_loss: 0.0530 - val_accuracy: 0.9831
Epoch 9/20
95/95 [==============================] - 60s 626ms/step - loss: 0.0528 - accuracy: 0.9852 - val_loss: 0.0449 - val_accuracy: 0.9870
Epoch 10/20
95/95 [==============================] - 59s 617ms/step - loss: 0.0426 - accuracy: 0.9868 - val_loss: 0.0447 - val_accuracy: 0.9857
Epoch 11/20
95/95 [==============================] - 63s 663ms/step - loss: 0.0384 - accuracy: 0.9888 - val_loss: 0.0428 - val_accuracy: 0.9870
Epoch 12/20
95/95 [==============================] - 57s 596ms/step - loss: 0.0408 - accuracy: 0.9885 - val_loss: 0.0376 - val_accuracy: 0.9883
Epoch 13/20
95/95 [==============================] - 57s 595ms/step - loss: 0.0350 - accuracy: 0.9901 - val_loss: 0.0375 - val_accuracy: 0.9883
Epoch 14/20
95/95 [==============================] - 58s 605ms/step - loss: 0.0332 - accuracy: 0.9901 - val_loss: 0.0387 - val_accuracy: 0.9883
Epoch 15/20
95/95 [==============================] - 55s 576ms/step - loss: 0.0371 - accuracy: 0.9878 - val_loss: 0.0353 - val_accuracy: 0.9896
Epoch 16/20
95/95 [==============================] - 55s 575ms/step - loss: 0.0364 - accuracy: 0.9901 - val_loss: 0.0373 - val_accuracy: 0.9883
Epoch 17/20
95/95 [==============================] - 55s 581ms/step - loss: 0.0333 - accuracy: 0.9911 - val_loss: 0.0334 - val_accuracy: 0.9909
Epoch 18/20
95/95 [==============================] - 58s 613ms/step - loss: 0.0294 - accuracy: 0.9914 - val_loss: 0.0346 - val_accuracy: 0.9909
Epoch 19/20
95/95 [==============================] - 56s 586ms/step - loss: 0.0249 - accuracy: 0.9924 - val_loss: 0.0353 - val_accuracy: 0.9896
Epoch 20/20
95/95 [==============================] - 58s 605ms/step - loss: 0.0286 - accuracy: 0.9901 - val_loss: 0.0330 - val_accuracy: 0.9909
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767

shivamsinghal@Shivams-Mac ai-jcomp %
```

Fig 7.2.1: Training Screenshots



```
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767
```

Fig 7.2.2: Training statistics

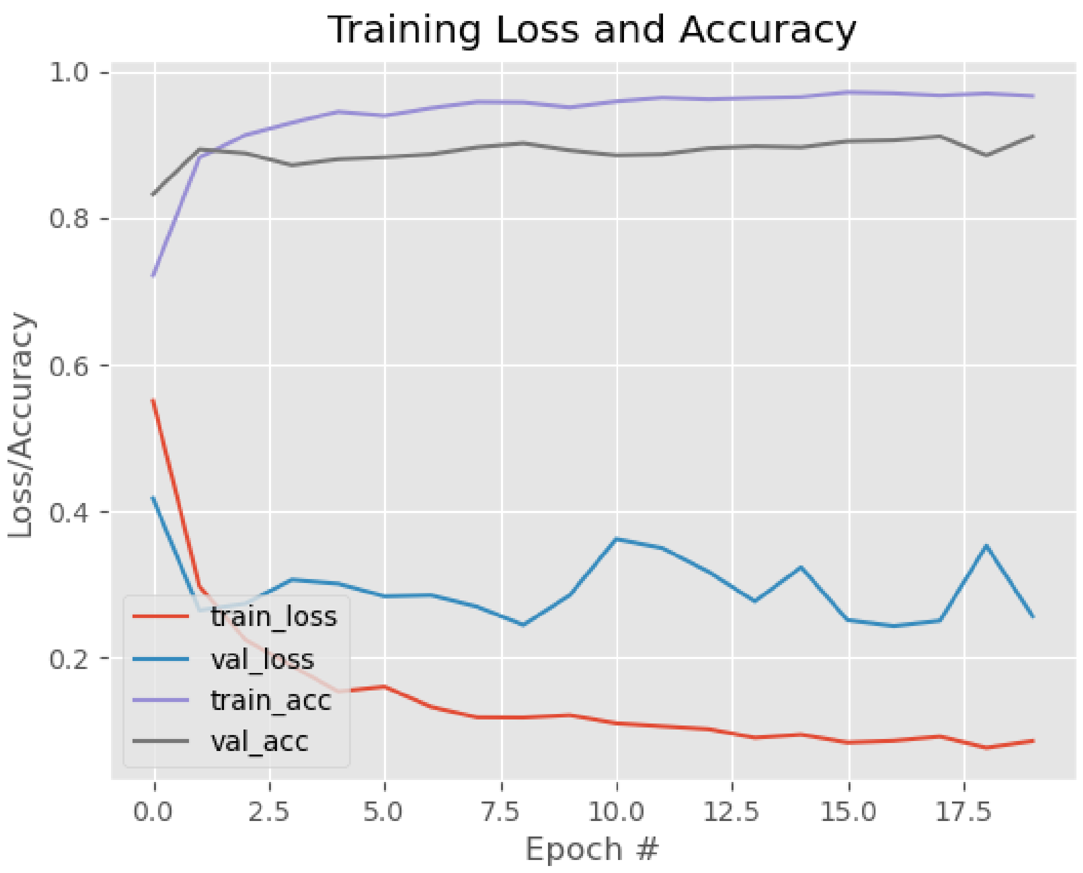
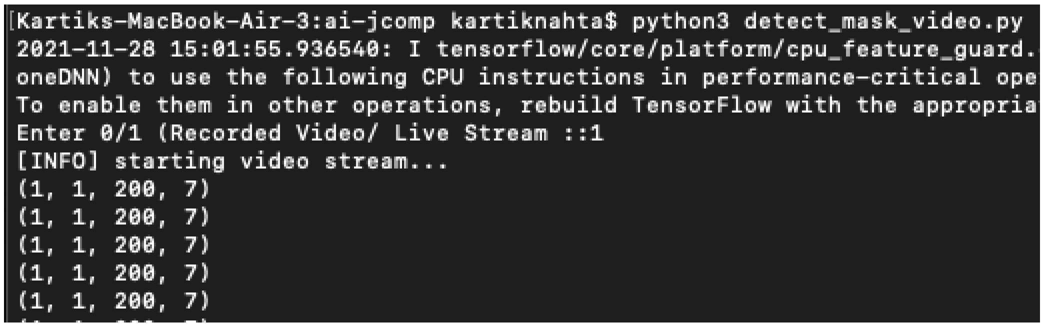


Fig 7.2.3: Plot generated after training

MASK DETECTION

Input 1 runs camera (live video stream)



```
[Kartiks-MacBook-Air-3:ai-jcomp kartiknahta$ python3 detect_mask_video.py
2021-11-28 15:01:55.936540: I tensorflow/core/platform/cpu_feature_guard.
oneDNN) to use the following CPU instructions in performance-critical ope
To enable them in other operations, rebuild TensorFlow with the appropria
Enter 0/1 (Recorded Video/ Live Stream ::1
[INFO] starting video stream...
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
```

Fig 7.2.4: Enter option 1 in terminal for live stream

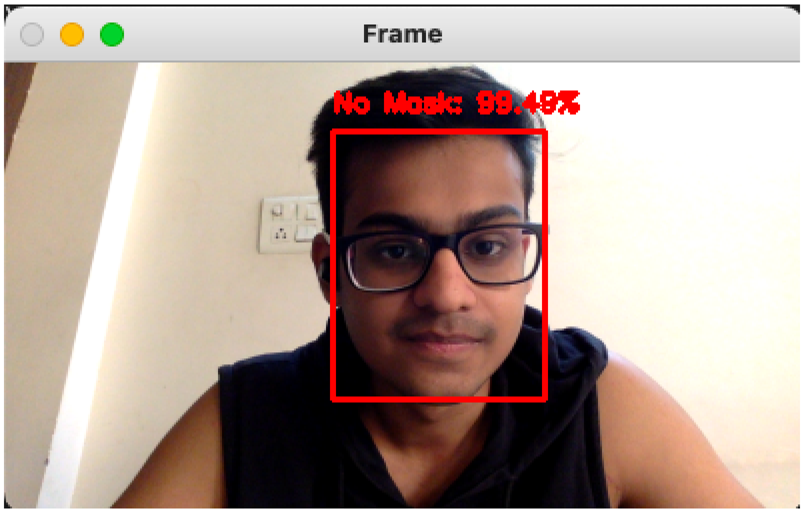


Fig 7.2.5: No mask worn

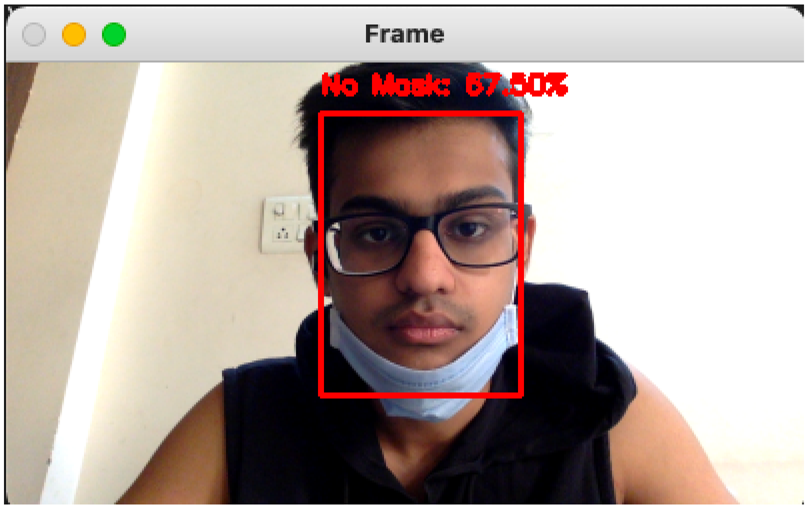


Fig 7.2.6: Mask worn improperly (mouth & nose exposed)

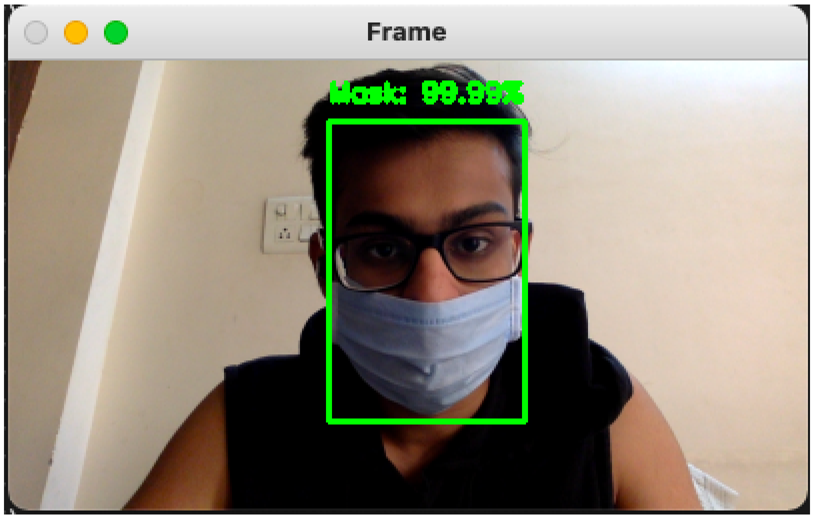
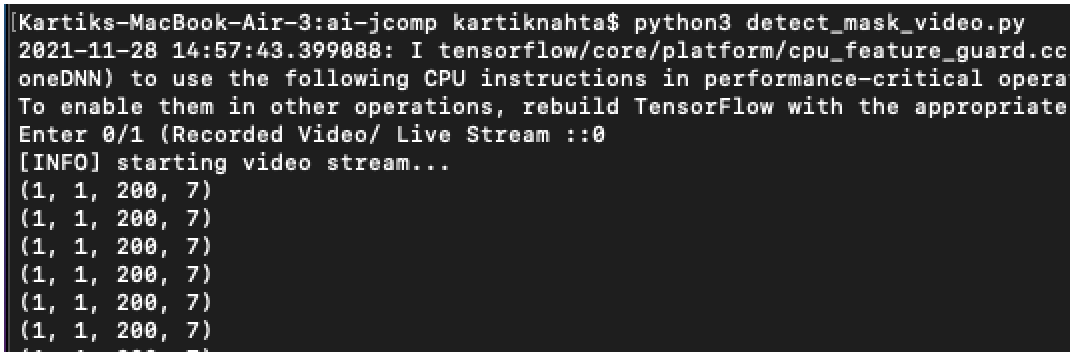


Fig 7.2.7: Mask worn properly (mouth & nose covered completely)

Input 0 runs prerecorded video



```
[Kartiks-MacBook-Air-3:ai-jcomp kartiknahta$ python3 detect_mask_video.py
2021-11-28 14:57:43.399088: I tensorflow/core/platform/cpu_feature_guard.cc
oneDNN) to use the following CPU instructions in performance-critical opera
To enable them in other operations, rebuild TensorFlow with the appropriate
Enter 0/1 (Recorded Video/ Live Stream ::0
[INFO] starting video stream...
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
(1, 1, 200, 7)
```

Fig 7.2.8: Enter option 0 for pre-recorded video

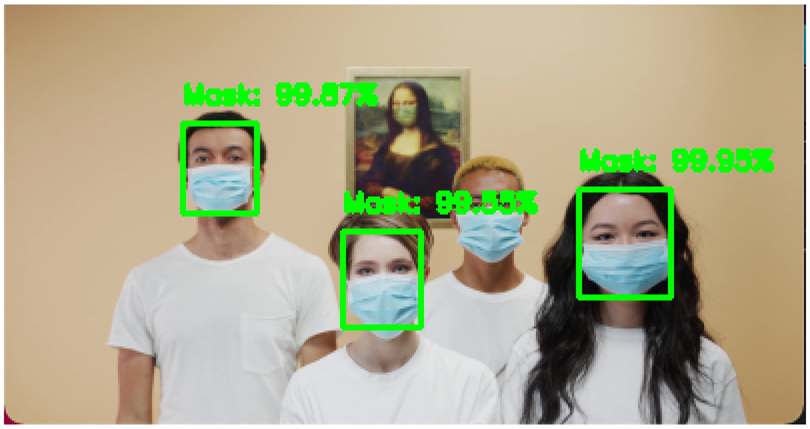


Fig 7.2.9: Detection of multiple people wearing mask

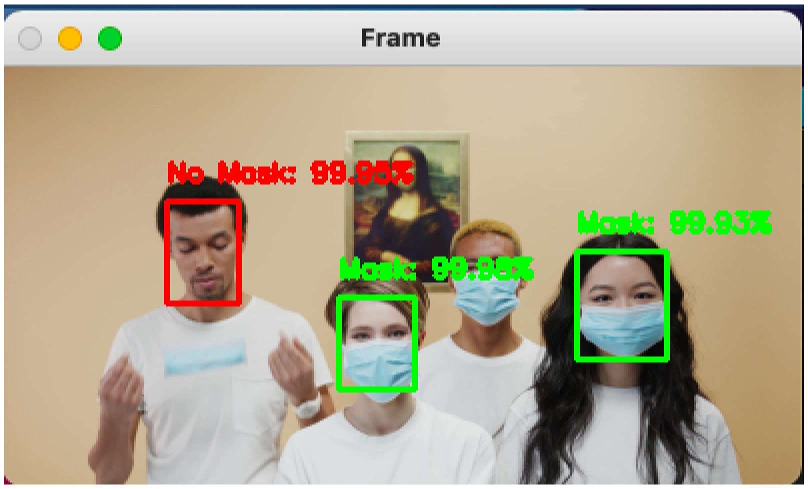


Fig 7.2.10: Detection of no mask amongst multiple people

REFERENCES -

8.1 LIST OF JOURNALS

[1]R. N. S and M. N, "Computer-Vision based Face Mask Detection using CNN," 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 1780-1786, doi: 10.1109/ICCES51350.2021.9489098.

[2]"Coronavirus Disease 2019 (COVID-19) – Symptoms", Centers for Disease Control and 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.

[4] M. Jiang, X. Fan and H. Yan, "RetinaMask: A Face Mask detector",

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

2020.

[5] B. Suvarnamukhi and M. Seshashayee, "Big Data Concepts and

Techniques in Data Processing", International Journal of Computer Sciences and Engineering, vol. 6, no. 10, pp. 712-714, 2018. Available: 10.26438/ijcse/v6i10.712714.

[6] F. Hohman, M. Kahng, R. Pienta and D. H. Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers," in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 8, pp. 2674-2693, 1 Aug. 2019, doi: 10.1109/TVCG.2018.2843369.

[7] C. Kanan and G. Cottrell, “Color-to-Grayscale: Does the Method Matter in Image Recognition?”, PLoS ONE, vol. 7, no. 1, p. e29740, 2012. Available: 10.1371/journal.pone.0029740.

[8] Opencv-python-tutroals.readthedocs.io. 2020. Changing Colorspaces — Opencv-Python Tutorials 1 Documentation. [online] Available at:https://opencv-python-tutroals.readthedocs.io/en/latest/py tutorials/ py imgproc/py colorspaces/py colorspaces.html. 2020.

[9] M. Hashemi, “Enlarging smaller images before inputting into convolu- tional neural network: zero-padding vs. interpolation”, Journal of Big Data, vol. 6, no. 1, 2019. Available: 10.1186/s40537-019-0263-7 . 2020.

[10] S. Ghosh, N. Das and M. Nasipuri, “Reshaping inputs for con- volutional neural network: Some common and uncommon meth- ods”, Pattern Recognition, vol. 93, pp. 79-94, 2019. Available: 10.1016/j.patcog.2019.04.009.

8.2 LIST OF WEBSITES (URLs)

https://www.ucsf.edu/news/2020/06/417906/still-confused-about-masks-heres-science-behind-how-face-masks-prevent

http://www.healthdata.org/news-release/new-ihme-covid-19-model-projects-nearly-180000-us-deaths

https://ieeexplore.ieee.org/abstract/document/9489098

https://keras.io/api/applications/mobilenet/

https://keras.io/api/

https://www.tensorflow.org/guide

https://www.kaggle.com/arbazkhan971/face-mask-detection-using-cnn-98-accuracy