K L HYDERABAD

FRESHMAN ENGINEERING DEPARTMENT

A Project Based Lab Report

On

Face Mask Detection COVID-19

SUBMITTED BY:

I.D NUMBER	NAME
2010030168	Tahseen Begum
2010030165	Syed Reshma Banu
2010030234	Bikkasani Kavya Mani Gayathri
2010030549	Grishma Neha Chowdary

UNDER THE ESTEEMED GUIDANCE OF

<DESIGNATION>

KONERU LAKSHMAIAH EDUCATION FOUNDATION

(Deemed to be University)
Moinabad Road, Aziz Nagar, Hyderabad - 500075

DEPARTMENT OF BASIC ENGINEERING SCIENCES

CERTIFICATE

This is to certify that the project based laboratory report entitled "**Face Mask Detection COVID-19**" submitted by Mr./Ms**. Tahseen Begum, Syed Reshma Banu, Bikkasani Kavya Mani Gayathri, Grishma Neha Chowdary** bearing Regd. No. 2010030168, 2010030165, 2010030234, 2010030549 to the **Department of Basic Engineering Sciences, KL University** in partial fulfilment of the requirements for the completion of a project in "" course in II B Tech III Semester, is a Bonafede record of the work carried out by him/her under my supervision during the academic year 2020-21.

PROJECT SUPERVISOR

<>

HEAD OF THE DEPARTMENT

<>

ACKNOWLEDGEMENTS

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project-based laboratory report.

I express the sincere gratitude to our Principal **Dr. L. Koteswara Rao** for his administration towards our academic growth.

I express sincere gratitude to our Coordinator **<> and** HOD-BES-1 **<>** for her leadership and constant motivation provided in successful completion of our academic semester. I record it as my privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to our project supervisor **<>** for her novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

Name: Tahseen Begum

Regd. No: 2010030168

Name: Syeda Reshma Banu

Regd. No: 2010030165

Name: Bikkasani Kavya Mani Gayathri

Regd. No: 2010030234

Name: Grishma Neha Chowdary

Regd. No: 2010030549

ABSTRACT

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society. This paper presents a simplified approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras, OpenCV and Scikit-Learn. The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.

INDEX

INTRODUCTION

According to the World Health Organization (WHO)'s official Situation Report – 205, coronavirus disease 2019 (COVID-19) has globally infected over 20 million people causing over 0.7million deaths. Individuals with COVID-19 have had a wide scope of symptoms reported – going from mellow manifestations to serious illness. Respiratory problems like shortness of breath or difficulty in breathing is one of them. Elder people having lung disease can possess serious complications from COVID-19 illness as they appear to be at higher risk. Some common human coronaviruses that infect public around the world are 229E, HKU1, OC43, and NL63. Before debilitating individuals, viruses like 2019-nCoV, SARS-CoV, and MERS-CoV infect animals and evolve to human coronaviruses. Persons having respiratory problems can expose anyone (who is in close contact with them) to infective beads. Surroundings of a tainted individual can cause contact transmission as droplets carrying virus may withal arrive on his adjacent surfaces.

To curb certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. The public should be aware of whether to put on the mask for source control or aversion of COVID-19. Potential points of interest of the utilization of masks lie in reducing vulnerability of risk from a noxious individual during the "pre-symptomatic" period and stigmatization of discrete persons putting on masks to restraint the spread of virus. WHO stresses on prioritizing medical masks and respirators for health care assistants Therefore, face mask detection has become a crucial task in present global society?

Face mask detection involves in detecting the location of the face and then determining whether it has a mask on it or not. The issue is proximately cognate to general object detection to detect the classes of objects. Face identification categorically deals with distinguishing a specific group of entities i.e. Face. It has numerous applications, such as autonomous driving, education, surveillance, and so on. This paper presents a simplified approach to serve the above purpose using the basic Machine Learning (ML) packages such as TensorFlow, Keras, OpenCV and Scikit-Learn.

AIM

The Face Mask Detection System can be used at office premises to detect if employees are maintaining safety standards at work. It monitors employees without masks and sends them a reminder to wear a mask.

Advantages: -

1. Intelligent Alerts.
2. Facial Recognition.
3. Camera Agnostic.
4. Easy Implementation.
5. Staff Friendly.
6. No New Hardware Needed.

Disadvantages: -

Lack of control over their personal information." Critics of mask recognition also think that this new technology could be prone to some of the same pitfalls as facial recognition. Many of the training datasets uses for facial recognition are dominating by light-skinned individuals.

⇒ In Future, we are planning to create camera using AI. Which we can use in streets, hospitals, schools, colleges and many more public places.

SYSTEM REQUIREMENTS

1. SOFTWARE REQUIREMENTS:

The major software requirements of the project are as follows:

Language : Python

Operating system: Windows XP or later.

Tools: Anaconda (Syder)

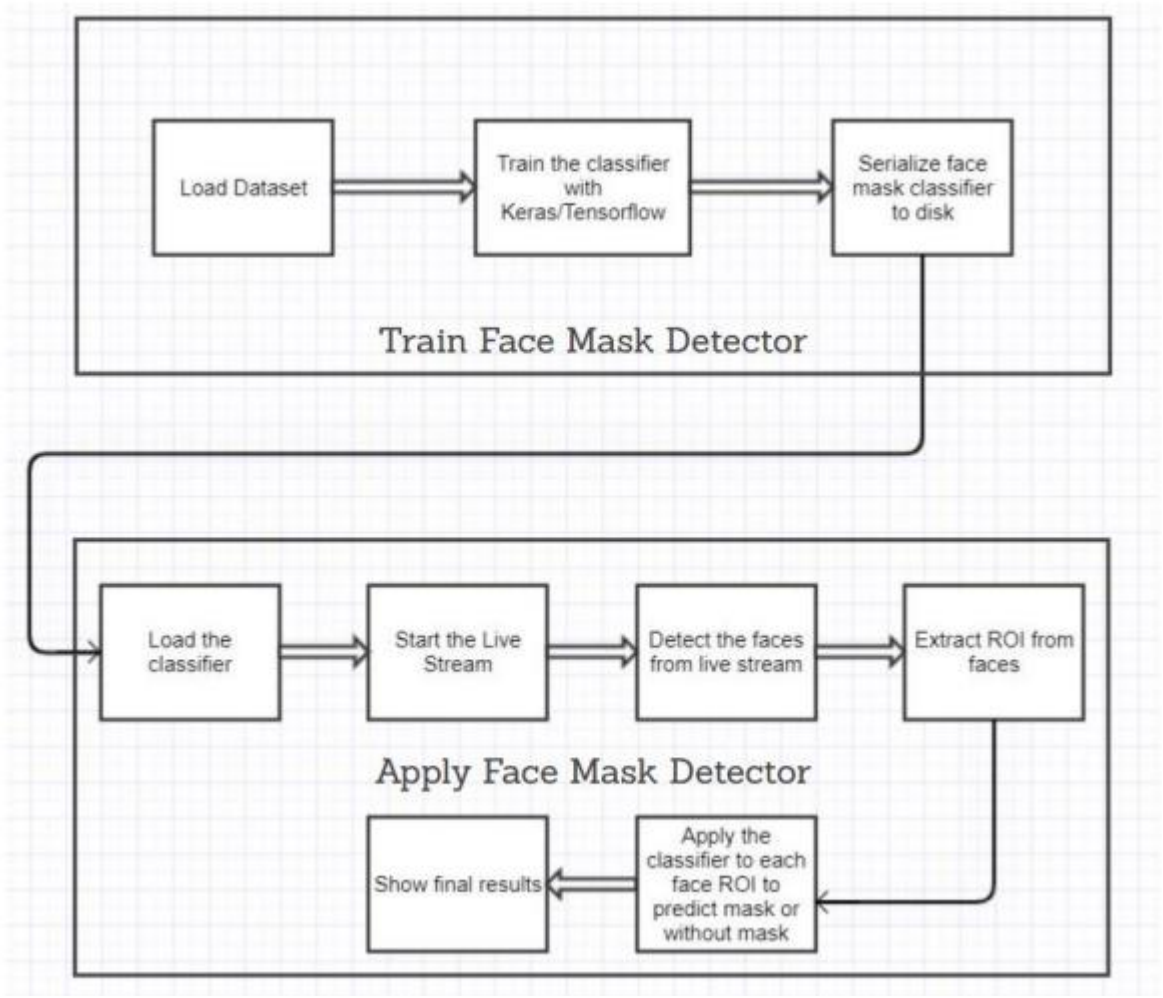
2. HARDWARE REQUIREMENTS:

The hardware requirements that map towards the software are as follows:

RAM : 1007

Processor : Intel

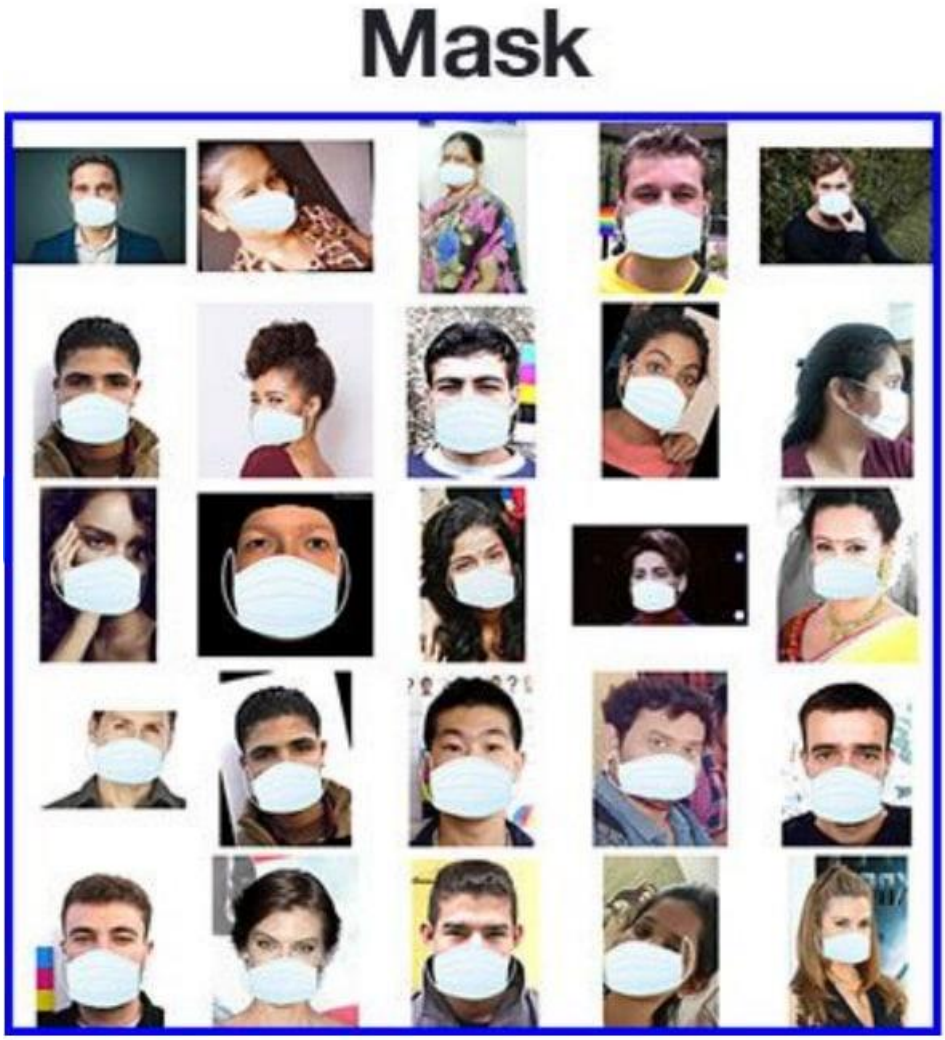
TRAIN FACE MASK DETECTOR MODEL

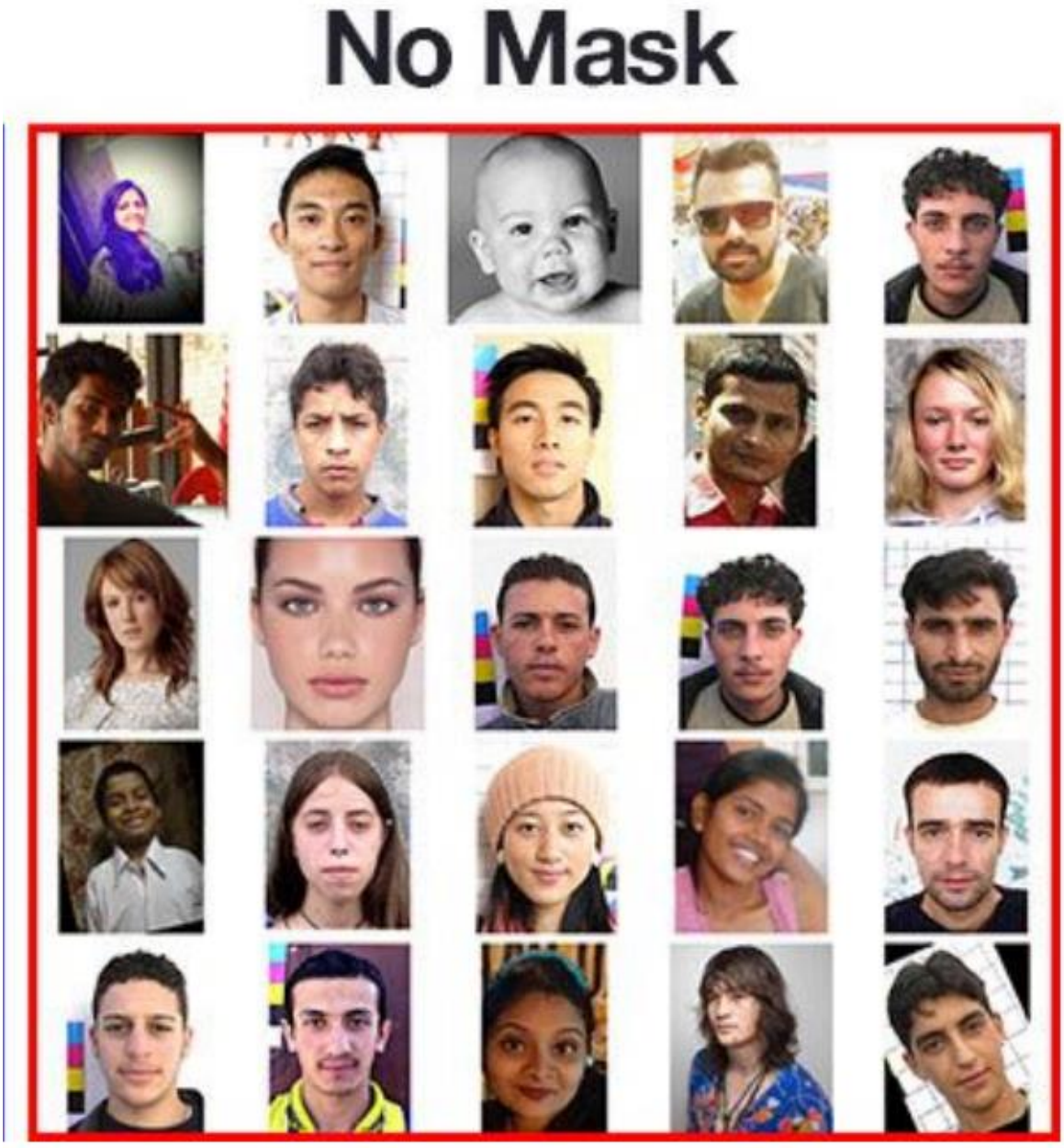


IMPLEMENTATION

DATA SET

Two datasets have been used for experimenting the current method. Dataset consists of 3176 images in which 1379 images with people wearing face masks and the rest 1797 images with people who do not wear face masks. Mostly contains front face pose with single face in the frame and with same type of mask having white colour only. Some face collections are head turn, tilt and slant with multiple faces in the frame and different types of masks having different colours as well.





Incorporated Packages

A. TensorFlow

TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of areas of computer science, including sentiment analysis, voice recognition, geographic information extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research [18]. In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data (image) in the data processing.

B. Keras

Keras gives fundamental reflections and building units for creation and transportation of ML arrangements with high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models [19]. All the layers used in the CNN model are implemented using Keras. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model.

C. OpenCV

OpenCV (Open-Source Computer Vision Library), an open-source computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth [20]. The proposed method makes use of these features of OpenCV in resizing and colour conversion of data images.

Techniques used to create images:

1. Taking normal images of faces

2. Creating a custom CV script to add face masks to them.

Usually, we infer the location of facial structures such as: Eyes, nose, eyebrows, mouth, jawline, teeth, moustache etc.

Steps:

1. Start with an image of person without mask

2. Apply face detection to compute the bounding box location of face.

3. Extract face Region of Interest (ROI)

4. Get image of a mask, and align it on top of the face properly.

5. Repeat the steps for multiple images

SOURCE CODE

Train_mask_detector.py

```
# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os



# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32
```

```
DIRECTORY = r"C:\Users\tahseen\OneDrive - K L University\PROJECT\FaceMaskDetection-main\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = []
labels = []

for category in CATEGORIES:
   path = os.path.join(DIRECTORY, category)
   for img in os.listdir(path):
         img_path = os.path.join(path, img)
         image = load_img(img_path, target_size=(224, 224))
         image = img_to_array(image)
         image = preprocess_input(image)

         data.append(image)
         labels.append(category)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

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

```
(trainX, testX, trainY, testY) = train_test_split(data, labels,
	test_size=0.20, stratify=labels, random_state=42)

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
	rotation_range=20,
	zoom_range=0.15,
	width_shift_range=0.2,
	height_shift_range=0.2,
	shear_range=0.15,
	horizontal_flip=True,
	fill_mode="nearest")

# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
	input_tensor=Input(shape=(224, 224, 3)))

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

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

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

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
	layer.trainable = False




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

# train the head of the network
print("[INFO] training head...")
H = model.fit(
	aug.flow(trainX, trainY, batch_size=BS),
	steps_per_epoch=len(trainX) // BS,
	validation_data=(testX, testY),
	validation_steps=len(testX) // BS,
	epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
```

```
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
        target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")



# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

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 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, 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
print("[INFO] starting video stream...")
```

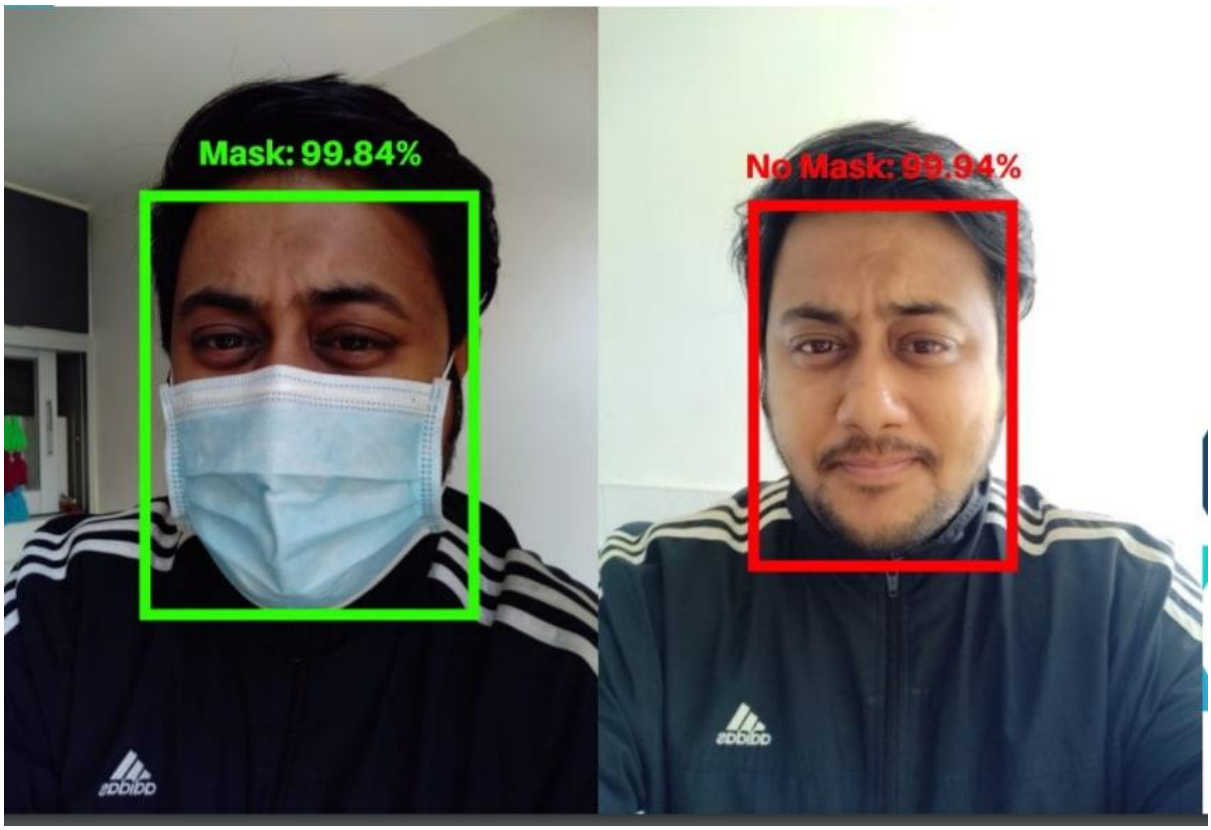
```
vs = VideoStream(src=0).start()

# 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=800)

	# 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, safe" if mask > withoutMask else "No Mask, not safe"
		color = (0, 255, 0) if label == "Mask, safe" 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_Face_Mask_Detection", 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()
```

Result And Analysis

The model is trained, validated and tested upon two datasets. Corresponding to dataset 1, the method attains accuracy up to 95.77% depicts how this optimized accuracy mitigates the cost of error. Dataset 2 is more versatile than dataset 1 as it has multiple faces in the frame and different types of masks having different colors as well. Therefore, the model attains an accuracy of 94.58% on dataset depicts the contrast between training and validation loss corresponding to dataset . One of the main reasons behind achieving this accuracy lies in MaxPooling. It provides rudimentary translation invariance to the internal representation along with the reduction in the number of parameters the model has to learn. This sample-based discretization process down-samples the input representation consisting of image, by reducing its dimensionality. Number of neurons has the optimized value of 64 which is not too high. A much higher number of neurons and filters can lead to worse performance. The optimized filter values and pool_size help to filter out the main portion (face) of the image to detect the existence of mask correctly without causing over-fitting.

OUTPUTS



CONCLUSION

& Future Work

- In this work, a deep learning-based approach for detecting masks over faces in public places to curtail the community spread of Coronavirus is presented. The proposed technique efficiently handles occlusions in dense situations by making use of an ensemble of single and two-stage detectors at the pre-processing level. The ensemble approach not only helps in achieving high accuracy but also improves detection speed considerably. Furthermore, the application of transfer learning on pre-trained models with extensive experimentation over an unbiased dataset resulted in a highly robust and low-cost system. The identity detection of faces, violating the mask norms further, increases the utility of the system for public benefits.

- Finally, the work opens interesting future directions for researchers. Firstly, the proposed technique can be integrated into any high-resolution video surveillance devices and not limited to mask detection only. Secondly, the model can be extended to detect facial landmarks with a facemask for biometric purposes