

AN AUTOMATED SYSTEM TO REDUCE COVID-19 SPREAD USING FACIAL MASK DETECTION

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

RIDHI MAYURA SHIVANI R (Reg. No. 37110640)

RUCHI OJHA (Reg. No. 37110649)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

CHENNAI – 600 119

APRIL - 2021



**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY**

(Established under Section 3 of UGC Act, 1956)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai -600119

www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Ridhi Mayura Shivani R (37110640)** and **Ruchi Ojha(37110649)** who carried out the project entitled “**AN AUTOMATED SYSTEM TO REDUCE COVID-19 SPREAD USING FACIAL MASK DETECTION**” under our supervision from December 2020 to April 2021.

Internal Guide

Dr. A. JESSUDOSS, M.E., Ph.D

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **Ridhi Mayura Shivani R (37110640)** and **Ruchi Ojha(37110649)**, hereby declare that the project report entitled ” **AN AUTOMATED SYSTEM TO LIMIT COVID-19 USING FACIAL MASK DETECTION** “ done by us under the guidance of **Dr. A. JESSUDOSS, M.E., Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering in Computer Science and Engineering.

DATE:

1. R RIDHI MAYURA SHIVANI

PLACE:

2. RUCHI OJHA

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board Of Management of Sathyabama for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. L. LAKSHMANAN, M.E., Ph.D.**, and **Dr. S. VIGNESHWRI, M.E., Ph.D.**, Head Of the Department, Dept. of computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our project Guide **Dr. A. JESSUDOSS, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all teaching and nonteaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for completion of the project.

ABSTRACT

The field of machine learning is a rapidly developing one. Recent developments in the field of image processing and machine learning has led to efficient extraction of features from images of people's face. Pandemic caused by novel coronavirus is continuously spreading until now all over the world. The impact of COVID-19 has been fallen on almost all sectors of development. The healthcare system is going through a crisis. Many precautionary measures have been taken to reduce the spread of this disease where wearing a mask is one of them. we propose a system that restrict the growth of COVID-19 by finding out people who are not wearing any facial mask in a smart city network where all the public places are monitored with Closed-Circuit Television (CCTV) cameras. While a person without a mask is detected, the corresponding authority is informed through the city network. A deep learning architecture is trained on a dataset that consists of images of people with and without masks collected from various sources.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURE	vi
LIST OF ABBREVIATION	vii

CHAPTER NO	TITLE	PAGE NO
1.	INTRODUCTION	1
	1.1 OBJECTIVE	2
2.	LITERATURE REVIEW	3
3.	MOTIVATION AND SCOPE OF PROJECT	16
	3.1 MOTIVATION	16
	3.2 SCOPE OF THE PROJECT	16
	3.3 EXISTING SYSTEM	16
	3.4 PROPOSED SYSTEM	16
	3.5 ADVANTAGE	16
4.	PROPOSED WORK	18
	4.1 SOFTWARE AND HARDWARE REQUIREMENTS	18

	4.1.1 SOFTWARE REQUIREMENTS	18
	4.1.2 HARDWARE REQUIREMENTS	18
	4.2 PROPOSED ARCHITECTURE	18
	4.3 ALGORITHM	19
	4.4 MASK DETECTION	19
	4.4.1 DATASET CREATION	19
	4.4.2 FACIAL LANDMARKS	19
	4.4.3 EYES	19
	4.4.4 EYEBROWS	20
	4.4.5 NOSE	20
	4.4.6 MOUTH	20
	4.4.7 JAWLINE	20
5	IMPLEMENTATION	23
	5.1 TRAINING PHASE	23
	5.2 DEPLOYMENT PHASE	23
	5.3 METHODOLOGY	23
	5.4 MODULE IMPLEMENTATION	24
	5.4.1 DATA PREPROCESSING	24
	5.4.2 TRAINING THE CNN	24
	5.4.3 DETECTING MASK AND GENERATING ALERTS	25
6	RESULTS AND DISCUSSION	26
	6.1 SUGGESSTIONS FOR IMPROVEMENT	29
7	CONCLUSION	31
	7.1 DRAWBACKS	31

7.1.1 POOR IMAGE QUALITY LIMITS	31
FACIAL RECOGNITION	
EFFECTIVENESS	
7.1.2 SMALL IMAGE SIZE	31
7.1.3 DIFFERENT FACE ANGLES	31
7.1.4 DATA PROCESSING AND STORAGE	32
	33
REFERENCE	
	37
APPENDIX	
A SAMPLE CODE	37
B SCREEN SHOTS	51

LIST OF FIGURES

FIGURE NUMBER	TITLE	PAGE NO
3.1	HUMAN FACE WITH AND WITHOUT MASK	17
4.1	ARCHITECTURE	18
4.2	CNN ARCHITECTURE	18
4.4.1	FACE WITHOUT MASK	20
4.4.2	APPLY FACE DETECT	21
4.4.3	EXTRACT THE ROI	21
4.4.4	DETECT FACIAL LANDMARKS	21
4.4.5	FACIAL MASK	22
4.4.6	MASKED FACE	22
4.4.7	SET OF COVID-19 FACE MASK	22
5.1	ISOLATING FACE SHAPES	24
6.1	EPOCH VALUES	26

6.2	LOSS AND ACCURACY PLOT	26
6.3	MASKED FACE	26
6.4	NO MASK FACE	26
6.5	MASK AND NO MASK FACE	26
7.1	EPOCH	51
7.2	ACCURACY TEST	51
7.3	ACCURACY PLOT	52
7.4	MASKED FACE	53
7.5	NO MASK FACE	54

LIST OF ABBREVIATIONS

ABBREVIATIONS

EXPANSIONS

CNN

CONVOLUTIONAL NEURAL
NETWORK

CCTV

CLOSED CIRCUIT ABBREVIATION

CHAPTER 1

INTRODUCTION

Machine learning and deep learning plays an important role in computer technology and artificial intelligence. With the use of deep learning and machine learning, human effort can be reduced in recognizing, learning, predictions and many more areas.

The year 2020 has shown mankind some mind-blowing series of events amongst which the COVID-19 pandemic caused by novel coronavirus is continuously spreading until now all over the world. The impact of COVID-19 has been fallen on almost all sectors of development.

The healthcare system is going through a crisis. Many precautionary measures have been taken to reduce the spread of this disease where wearing a mask is

one of the personal protective equipment. people wear mask once they step out of house and authorities strictly ensure that people are wearing face mask while they are in groups and public places.

To monitor that people are following the basic safety principle, a strategy should be developed. In this project, we propose a system that restrict the growth of COVID19 by finding out people who are not wearing any facial mask in a smart city network where all the public places are monitored with Closed-Circuit Television (CCTV) cameras.

The scope of this article is to compare the different classifiers with different parameters and try to achieve near-human performance.

Face mask detection system is the working of a machine to train itself or recognizing the faces with and without masks from different sources like videos, images, etc.

we have developed a deep learning model for face mask detection using Python, Keras, and OpenCV.

We developed the face mask detector model for detecting whether person is wearing a mask or not. We have trained the model using Keras with network architecture. Training the model is the first part of this project and testing using webcam using OpenCV is the second part.

1.1 OBJECTIVE:

The aim of this project is to identify the person in the video obtained from cctv wearing face mask with the help of keras , opencv , tensorflow , MobileNetV2.

CHAPTER 2

LITERATURE SURVEY

Hanan a Hosni Mahmoud and Hanan Abdullah Mengash(2020)face mask detection. This paper proposes a novel technique for automated concealed face detection in surveillance vedios. The model gives improved accuracy, but the modelwas not 100 percent accurate.

Amrit Kumar Bhadani and Anurag Sinha(2020)face mask detector. A face mask detector using machine learning and image processing techniques. Model gives improved accuracy but it was not 100 percent accurate.

T.Subhama Stan Rao, S. Anjali Devi and P. Dileep(2020)face mask detection. A novel approach to detect face mask to control covid using deep learning. The model gives improved accuracy but it was not 100 percent accurate.

Dr. Vandana S. Bhat, Arpita Durga Shambavi. Komal Mainalli, K M Manushree, Shraddha V Lakamapur(2021)Human Recognition with Face Mask. This paper aims to present a review of various methods and algorithms used for human recognition with face mask. Different approaches like Haar cascade, Adaboost, VGG-16 model,etc. are described in this paper. A comparative analysis is made on these methods to conclude which approach is feasible. With the advancement of technology and time more reliable methods for human recognition with a face mask can be implemented in the future. Finally, it includes some of the applications of face detection. This system has various applications at public places, schools, etc. where people need to be detected with the presence of a face mask and recognize them and help society.

MAMATA S. KALAS(2014) REAL TIME FACE DETECTION AND TRACKING USING OPENCV. This research paper propose system based on Adaboost. Pre-processing and cropping filter: The images of the dataset are already cropped around the face, so there is no need of a face detection stage to localize the face from each image. To do so, we detect 68 facial landmarks using Dlib-ml open- source library. According to the eyes location, we apply a 2D rotation to make them horizontal. The next step is to apply a cropping filter in order to extract only the non- masked region. To do so, we firstly normalize all face images into 240 x 240 pixels. Next, we use the partition into blocks. The principle of this technique is to divide the image into 100 fixed-size square blocks (24 x 24 pixels in our case). Then we extract only the blocks including the non-masked region (blocks from number 1 to 50). Finally, we eliminate the rest of the numbers of the blocks.

Feature extraction layer: They extract deep features using VGG16 face CNN descriptor [20] from the 2D images. It is trained on ImageNet dataset which has over 14 million images and 1000 classes. Its name VGG16 comes from the fact that it has 16 layers. Its layers consist of convolutional layers, Max Pooling layers, Activation

layers, Fully connected layers. There are 13 convolutional layers, 5 Max Pooling layers and 3 Dense layers which sums up to 21 layers but only 16 weight layers. In this work, we only consider the feature maps (FMs) at the last convolutional layer, also called channels. These features will be used in the following in the quantization stage.

Deep bag of features layer: From the i th image, we extract feature maps using the feature extraction layer described above. In order to measure the similarity between the extracted feature vectors and the codewords also called term vectors, we applied the RBF kernel as a similarity metric as proposed in. Thus, the first sub layer will be composed of RBF neurons, each neuron is referred to as a codeword. The size of the extracted feature map denotes the number of the feature vectors that will be used in the BoF layer. Here we refer by V_i to the number of feature vectors extracted from the i th image.

The most used automatic algorithm is of course k-means. Let F the set of all the feature vectors, defined by $F = \{V_{ij}, i = 1$

... $V, j = 1 \dots V_i\}$ and V_k is the number of the RBF neurons centers referred by c_k . Note that these RBF centers are learned afterward to get the final codewords. The quantization is then applied to extract the histogram with a pre-defined number of bins, each bin is referred to as a codeword. RBF layer is then used as a similarity measure, it contains 2 sub layers:

RBF layer: Measures the similarity of the input features of the probe faces to the RBF centers. Formally: the j th RBF neuron (X_j) is defined by: $(X_j) = \exp (||x - c_j||^2 / j)$,
(1) Where x is a feature vector and c_j is the center of the j th RBF neuron.

Quantization layer: The output of all the RBF neurons is collected in this layer that contains the histogram of the

global quantized feature vector that will be used for the classification process. The final histogram is defined by:

Where (V) is the output vector of the RBF layer over the ck bins.

Once the global histogram is computed, pass the classification stage to assign each test image to its identity. To do so, a Multilayer perceptron classifier (MLP) is applied where each face is represented by a term vector. Deep BoF network can be trained using back-propagation and gradient descent. Note that the 10 cross-validation strategy is applied in our experiments on the RMFRD dataset. $V = [v_1, \dots, v_k]$ the term vector of each face is noted, where each v_i refers to the occurrence of the term i in the given face. t is the number of attributes, and m is the number of classes (face identities). Test faces are defined by their codeword V . MLP uses a set of term occurrences as input values (v_i) and associated weights (w_i) and a sigmoid function (g) that sums the weights and maps the results to an output (y).

RMFRD faces were firstly pre-processed as described. Using the normalized 2D faces of sizes 240×240 pixels, VGG16 pretrained model is applied to extract the best features from the last convolutional layer as presented. The quantization is then applied to extract the histogram of 70 bins as presented. Finally, MLP is applied to classify faces. In this experiment, the 10 cross-validation strategy is used to evaluate the recognition performance. The experiments are repeated ten times in the RMFRD dataset, where 9 samples are used as the training set and the remaining sample as the testing set, and the average results are calculated.

Vinitha.V, Velantina.V(2020). covid-19 facemask detection with deep learning and computer vision. The proposed system focuses on how to identify the person on image/video stream wearing face mask with the help of computer vision and deep learning algorithm by using the OpenCV, Tensor flow, Keras and PyTorch library.

Approach

1. Train Deep learning model (MobileNetV2)
2. Apply mask detector over images / live video stream

The majority of the images were augmented by OpenCV. The set of images were already labeled mask and no mask.

The images that were present were of different sizes and resolutions, probably extracted from different sources or from machines (cameras) of different resolutions.

Face Mask Detection in webcam stream.

The flow to identify the person in the webcam wearing the face mask or not. The process is two-fold.

1. To identify the faces in the webcam.
2. Classify the faces based on the mask.

Identify the Face in the Webcam: To identify the faces a pre-trained model provided by the OpenCV framework was used. The model was trained using web images. OpenCV provides 2 models for this face detector:

1. Floating-point 16 version of the original Caffe implementation.
2. 8 bit quantized version using Tensor flow.

C.Jagadeeswari, M. Uday Theja(2020) Performance Evaluation of Intelligent Face Mask Detection System with various Deep Learning Classifiers. Corona virus disease (COVID-19) is an airborne infectious disease caused by a newly discovered corona virus. The best way to prevent or slow down the transmissions to have knowledge on the COVID-19 virus, the disease it can cause and how it passes. There are many steps suggested by WHO (World Health Organization) to prevent the spread. One of which is wearing medical masks which is highly desirable even after the lockdown period until a vaccine/Medicine is invented. This system aims at classifying whether a person is wearing a mask or not by taking input from Images

and Real time streaming Videos. The classification of the images is done by training the model in 2 phases:

Phase 1: Face mask dataset is loaded into the system. Different classifiers like MobileNetV2, ResNet50, and VGG16 are used to generate a trained model.

Phase2: Load the face mask classifier model.

Detect faces in the images/video stream. Apply the classifier to each face RoI. Classify the images to be With Mask and Without Mask with Confidence.

This system may then be interfaced with

Case 1: Existing access control system so that violators can be restricted.

Case 2: There could be some scenarios in work places where people may forget or just put off the mask when it becomes uneasy for them to get accustomed to the new face masks. In such cases, alarm by the system may be disturbing other workers. Hence the concerned authorities can take proper measures to alert the user so that they can wear the mask again.

Karan Malik(2020), Real Time Face Mask Detection using Deep Learning and Opencv. This paper proposes a project that uses a deep neural network, more specifically a Convolutional Neural Network, to differentiate between images of people with and without masks. The CNN manages to get an accuracy of **98.2% on the training set** and **97.3% on the test set**. Then the stored weights of this CNN are used to classify as mask or no mask, in real time, using OpenCV. With the webcam capturing the video, the frames are preprocessed and fed to the model to accomplish this task. The model works efficiently with no apparent lag time between wearing/removing mask and display of prediction.

The model is capable of predicting multiple faces with or without masks at the same time.

Farhan Sadaf(2019), Face Mask Detection. This paper proposes a project that follows three steps to detect masks:

1. **DATA PREPROCESSING** : All images have their bounding boxes in the PASCAL VOC format and their info saved in XML format in annotations directory. Only the region bounded by bounding box taken as input and their respective labels taken as output.
2. **TRAINING MASK DETECTOR MODEL** : Transfer learning was used to train the inputs. The classifier model was built with InceptionV3 neural network architecture. After training for 20 epochs, accuracy on test set was 96.81%.
3. **DETECTING FACE MASK IN REAL-TIME** : First task was to detect faces from each frame of the video. At first It used Haarcascade classifier from OpenCV for face detection. Average FPS which got while running on my machine was around 16. But face detection wasn't that accurate. This classifier struggled detecting faces with mask. In low-light condition it struggled.
Then It tried MTCNN for face detection. This algorithm performed great detecting faces, even in the low light. But while running on my machine, the average FPS I got was about 1.4. Which is pretty slow comparing with haarcascade classifier.

Guru Charan(2021), human face mask detection. This paper propose system based on CNN algorithm. In order to protect ourselves from the COVID-19 Pandemic, almost every one of us tend to wear a face mask. It becomes increasingly necessary to check if the people in the crowd wear face masks in most public gatherings such as Malls, Theatres, Parks. The development of an AI solution to detect if the person is wearing a face mask and allow their entry would be of great help to the society. In this, a simple Face Mask detection system is built using the Deep Learning technique called as Convolutional Neural Networks (CNN). This CNN Model is built using the TensorFlow framework and the OpenCV library which is highly used for real-time applications. This model can also be used to develop a full-fledged software to scan every person before they can enter the public gathering. Using this

model, an accuracy of over 96% is obtained. This can also be used further to achieve even higher levels of accuracy.

Anagha S. and Dr. R. K. Kulkarni(2014) Face Detection and Facial Mask Detection System. This paper proposes Automatic Face Mask Detection System. In this system there are three phases 1.Face Detection 2.Feature extraction 3.Mask Detection. The first phase mask detection are done by YCbCr color model, lighting compensation for getting face and morphological operations for retaining required face i.e eye and mouth of the face.

Hemanth(2019) A Real time DNN based face mask detection system using single shot multibox detector and mobilenetv2. In this proposed paper to predict whether a person has worn a mask correctly, the initial stage would be to train the model using a proper dataset. After training the classifier, an accurate face detection model is required to detect faces, so that the SSDMNv2 model can classify whether the person is wearing a mask or not. The task in this paper is to raise the accuracy of mask detection without being too resource-heavy. For doing this task, the DNN module was used from OpenCV, which contains a 'Single Shot Multibox Detector' (SSD) ([Liu et al., 2016](#)) object detection model with ResNet-10 ([Anisimov & Khanova, 2017](#)) as its backbone architecture. This approach helps in detecting faces in real-time, even on embedded devices like Raspberry Pi. The following classifier uses a pre-trained model MobileNetV2 ([Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018](#)) to predict whether the person is wearing a mask or not.

Ramanan(2006) face mask detection. In this project proposed a Random forest tree model in face mask detection, which accurately guesses face structures and facial poses. one of the renowned researchers made a DPM-based face mask detector using around 30, 000 faces divided into masks and without masks category. His work achieved an exceptional accuracy of 97.14 %. Further models of face mask detectors were made . Typically, DPM-based face mask detection models can achieve majestic precisions, but it may be tolerant from the very towering cost of computation due to the use of DPM.

In the recent past, various researchers and analysts mainly focused on gray-scale face image by Ojala, Jala, Pietikainen, & Maenpaa(2002). While some were completely built on pattern identification models, possessing initial information of the face model while others were using AdaBoost Kim, Park, Woo, Jeong, & Min(2011), which was an excellent classifier for training purposes. Then came the Viola-Jones Detector, which provided a breakthrough in face detection technology, and real-time face detection got possible. It faced various problems like the orientation and brightness of the face, making it hard to intercept. So basically, it failed to work in dull and dim light. Thus, researchers started searching for a new alternative model that could easily detect faces as well as masks on the face.

In the past, many datasets for face detection were developed to form an impression of face mask detection models. Earlier datasets consisted of images fetched in supervised surroundings, while recent datasets are constructed by taking online images like WiderFace by Yang, Luo, Loy, & Tan(2016), IJB-A Klare et al., (2015a), MALF Yang, Yan, Lei, & Li(2015), and CelebA Klare et al., (2015b). Annotations are provided for present faces in these datasets as compared to earlier ones. Large datasets are much more needed for making better training and testing data and perform real-world applications in a much simpler way. This calls for various deep learning algorithms which can read faces and mask straight from the data provided by the user.

Face Mask detection models have many variations. These can be divided into several categories. In Boosting-based classification, boosted cascades with easy haar features were embraced using the Viola-Jones face detector by Jones, Viola, & Jones(2001), which was discussed above in this section. Then a Multiview face mask detector was made motivated by the Viola-Jones detector model. In addition to this, a face mask detector model was made using decision trees algorithms. Face mask detectors in this category were very effective in detecting face masks.

In Deformable Part Model-based classification, the structure and orientations of several different faces are modelled using DPM. Zhang, Zhang, Li, and Qiao (2016), one of the renowned researchers made a DPM-based face mask detector using

around 30, 000 faces divided into masks and without masks category. His work achieved an exceptional accuracy of 97.14 %. Further models of face mask detectors were made by Chen, Ren, Wei, Cao, and Sun (2014). Typically, DPM-based face mask detection models can achieve majestic precisions, but it may be tolerant from the very towering cost of computation due to the use of DPM.

In Convolutional Neural Network-based classification, face detector models learn directly from the user's data and then apply several deep learning algorithms on it ([Ren, He, Girshick, & Sun, 2015](#)). In the year 2007, Li, Lin, Shen, Brandt, and Hua (2015) came up with Cascade CNN. In Yang, Yan et al. (2015), Yang et al. came up with the idea of features aggregation of faces in the face detection model. In further research works, Ojala et al. (2002) upgraded the AlexNet architecture for fine-tuning the image dataset. For uninhibited circumstances, Zhu et al. (2017) propose a Contextual Multi-Scale Region-based Convolutional Neural Network (CMS-RCNN), which brought a significant impact on the face detection models. To minimize the error on the substitute layers of CNN layers and dealing with the biased obstructions generated in the mask detection models, Opitz et al. (2016) prepared a grid loss layer. As technology advanced, further CNN-based 3D models started coming up; one was proposed by Li et al. (2015). It was a learning structure for face mask detection models. Several other works were done in the sphere of pose recognition, gender estimation, localization of landmarks, etc.

The Face mask detection model named SSDMNv2 has been developed using deep neural network modules from OpenCV and TensorFlow, which contains a Single Shot Multibox Detector object detection model. Typical classification architectures like ResNet-10 which is used as a backbone architecture for this model and image classification and fine-tuned MobileNetV2 classifier has been used, MobileNetV2 classifier has been an improvement over MobileNetV1 architecture classifier as it consisted of 3×3 convolutional layer as the initial layer, which is followed by 13 times the previous building blocks. In contrast, MobileNetV2 architecture is made up of 17, 3×3 convolutional layers in a row accompanied by a 1×1 convolution, an

average layer of max pooling, and a layer of classification. The residual connection is a new addition in the MobileNetV2 classifier.

F. Abdat and A. Pruski(2016) Human computer interaction using face mask detection. The author propose an system fully automatic facial expression and recognition system based on two step face detection, facial characteristics extraction. This system proposed anthropometric model to detect the face feature point combined to shi and Thoms method. In this method the variation of 21 distances which describe the facial feature from neutral face and the classification based on SVM.

Authors developed a face mask wearing condition identification method. They were ready to classify three categories of face mask-wearing. The categories are face maskwearing, incorrect face mask-wearing and no face mask-wearing. Saber et al, have applied the principal component analysis on masked and unmasked face recognition to acknowledge the person. Also, PCA was utilized . The author proposed a way that's used for removing glasses from human frontal faces. In, the authors used the YOLOv3 algorithm for face detection. YOLOv3 uses Darknet-53 because the backbone. Nizam et al proposed a completely unique GAN-based network, which will automatically remove mask covering the face area and regenerate the image by building the missing hole. The authors presented a system for detecting the presence or absence of a compulsory medical mask within the OR. The general is to attenuate the false positive face detection as possible without missing mask detection so as to trigger alarms just for medical staff who don't wear a surgical mask. Shaik et al used deep learning real-time face emotion classification and recognition. They used VGG-16 to classify seven countenance. Under the present Covid-19 lock-in time, this technique is effective in preventing spread in may use cases. Here are some use cases which will benefit form system.

Amrit Kumar Bhadani and Anurag Sinha(2020), A Facemask detector using machine learning and image processing techniques. The proposed system focuses on how to identify a person wearing a mask on the image/video stream. Help with computer vision and deep learning algorithm by using Open CV, tesnsorflow, Keras library.

Method: 1. Train deep learning model(MobileNetV2). 2. Apply face mask detector over images/live video stream. Step 1. Data visualization In the first step, let us visualize the total number of images in the dataset in these two categories. We can see that there are 800 images in the “Yes” category and 750 images in the “No” category.

Step 2. Data Augmentation In the next step, we expand the data set to include a larger number of images for training. In this step of data expansion, we rotate and flip each image in the data set. We see that after data expansion, we have total 2851 images of which the “yes” category contains 1430 images, and the “no” category contains 1421 images.

Step 3. Splitting the data In this step, we divide the data into a training set, and the training set will contain the images on which the CNN model will be trained and test set and the images on which the model will be tested. In this case, we use split_size =0.8, which means that 80% of the total images will enter the training set, and the remaining 20% of the images will enter the test set. After segmentation, we see that the required image percentage has been allocated to the training set and test set as described .

Step 4. Building the Model In the next step, we will use Conv2D, MaxPooling2D, Flatten, Dropout, and dense to build a sequential CNN model. In the last dense layer, we use the “ soft ax ” function to output vector that gives the probability of each of the two categories.

Step 5. Pre-Training the CNN model After setting up the model let us create”train generator” and “validation_generator” to make it fit our model in the next step. A total of 2250 images in the training set and a total of 551 images in the test set.

Step 6. Training the CNN model This is the main step in which we put Images into the training set and test set to use the sequence model built by the keras library. Trained the model for 20 epochs. However, train more epochs to obtain higher accuracy, so as to avoid overfitting. history = model.fit_generator(train_generator, epochs=20, validation_data=validation_generator, callbacks=[checkpoint])>>Epoch 20/20 220/220 [=====] - 231s 1s/step - loss: 0.0717

- acc: 0.9710 - val_loss: 0.0270 - val_acc: 0.9922 see that after the 20th epoch, our model has accuracy of 97.86% on the training set and 99.22% on the test set. This means that it is well-trained without any overfitting.

Step 7. Labeling the Information After building the model, we label the results with two probabilities. ["0" is "without mask", "1" is "with_mask"], also set the color of the bounding rectangle using RGB values. ["RED" stands for "without mask" "GREEN" stands for "with mask"].
`labels_dict={0:'without_mask',1:'with_mask'}`
`color_dict={0:(0,0,255),1:(0,255,0)}`. Step 8. Importing the Face Detection Program From now on, we plan to use it to detect whether we are wearing a mask through the pc's webcam. For this, first of all, we need to implement face detection

Step 9. Detecting the Faces with and Without Mask In the last step, we use the OpenCV library to run an infinite loop to use our webcam, where the cascade classifier is used to detect faces. The code `webcam=cv2.VideoCapture(0)` indicates the usage of the webcam. The model will predict the likelihood of each of the two categories [without mask, with mask]). Based on a higher probability, tags will be selected and displayed around our face.

Balaji Srinivasan(2020) Face Mask Detection. The proposed paper used keras, tensorflow, mobilenet, and opencv to build this project. The face mask detector did not use any morphed masked images dataset. The model was accurate and since the mobilenetv2 architecture is used, it is also computationally efficient and thus making it easier to deploy the model to embedded systems. This system can therefore be used in realtime applications which require face mask detection for safety purposes due to the outbreak of covid 19.

Thakshila(2020) face mask detection using CNN. In this paper the proposed project is built using CNN, python, keras, tensorflow and opencv. With further improvement, these types of models can be integrated with CCTV or other types of cameras to detect and identify people without masks.

Hanoor(2020) Face Mask Detection. In this proposed paper the project is build using pyhton, opencv and tensorflow. It uses brain- storming, concepts of transfer learning, deep learning, prediction on unknown images.

Nicholas(2020) Face Mask Detector. This paper proposes project which is build using opencv matplotlib.

Hemanth(2020), mask detection using python. In this proposed paper the project uses deep learning neural network. It will use RES10 model for face recognition, and recognises face using webcam.

Misbah Mohammed(2020) Face Mask Detection. In this proposed paperthe project uses keras, python and opencv to detect masked face .

CHAPTER 3

MOTIVATION AND SCOPE OF THE PROJECT

3.1 MOTIVATION:

In the present scenario due to Covid-19, there is no efficient face mask detection applications which are now in high demand for transportation means, densely populated areas, residential districts, large-scale manufacturers and other enterprises to ensure safety. Also, the absence of large datasets of 'with_mask' images has made this task more cumbersome and challenging.

3.2 SCOPE OF THE PROJECT:

Research on face mask detection is a very challenging field that targets methods to make effective human computer interaction. The scope of this project face mask detection regarding the application of machine learning algorithms based on image pre-processing and feature extraction. Additionally the purpose is to not only to improve the current recognition performance, but also to seek the highest reliability in the applications of face mask detection.

3.3 EXISTING SYSTEM:

Existing systems are highly complex in terms of time and storage for recognizing user behavior. Existing system has only 70 percentage of accuracy and the loss rate is high. To improve the accuracy rate and to decrease the loss rate this project is been developed.

3.4 PROPOSED SYSTEM:

The proposed system uses mobilenetv2 architecture and CNN algorithm to detect people's face with and without mask in video and images.

The proposed model predicts people face with and without mask at 99 percentage accuracy. the model follows two phases for detecting face and mask.

This helps police to monitor people and identify those without masks so that it helps in reducing covid-19 transmission from one person to another.

3.5 ADVANTAGE

Fast, accurate, and convenient to use. The response time of mask detection is typically about one-half seconds that might assist people in having fast track assist to the monitored area.

The US Department of Homeland Security (DHS) is piloting facial recognition technologies that can see through the face masks with a promising level of accuracy, meaning that travellers could end up breezing through airports without the need to uncover their mouths and noses at border checks.



Fig 3.1 :human face with and without mask detection

In the above figure 3.1 the image has people face both with and without mask.

CHAPTER 4

PROPOSED WORK

In this project, we have developed a deep learning model for face mask detection using Python, Keras, and OpenCV. We developed the face mask detector model for detecting whether person is wearing a mask or not. We have trained the model using

Keras with network architecture. Training the model is the first part of this project and testing using webcam using OpenCV is the second part.

4.1 SOFTWARE AND HARDWARE REQUIREMENTS:

4.1.1 SOFTWARE REQUIREMENT:

Windows 7 and above (64-bit)

Jupyter notebook

4.1.2 HARDWARE REQUIREMENT:

Hard disc-500 GB and above.

The proposed model uses Ram: 4GB and above.

4.2 PROPOSED ARCHITECTURE:

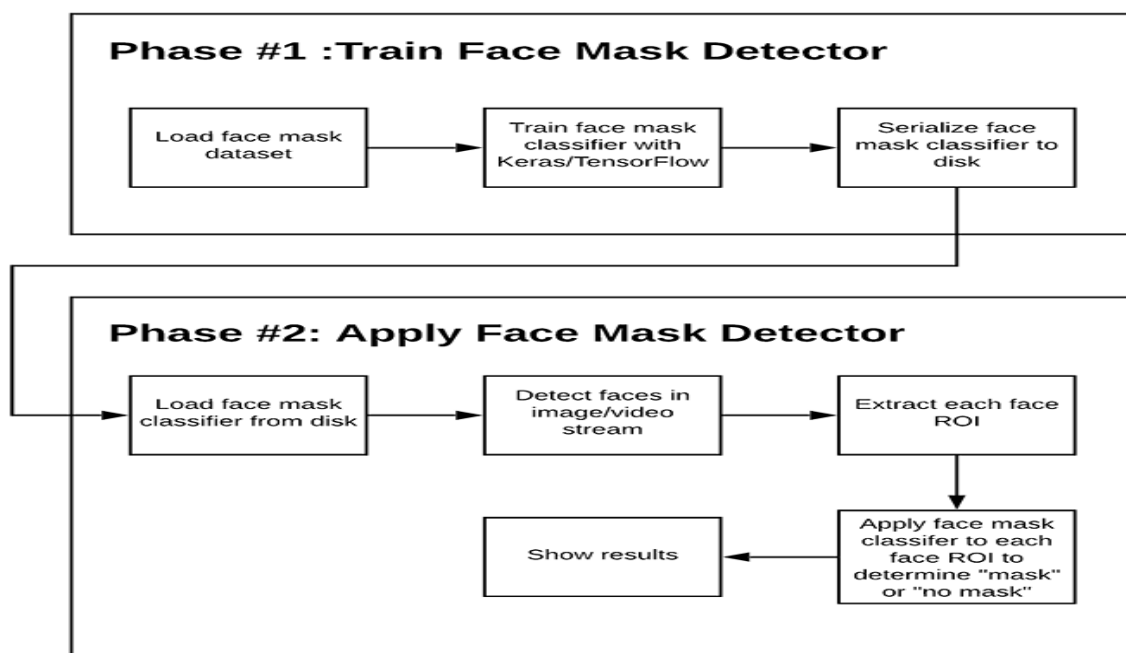


Fig 4.1 ARCHITECTURE

The architecture used here MobileNet V2 architecture.

4.3 PROPOSED ALGORITHM

The proposed project uses convolutional neural network algorithm(CNN).

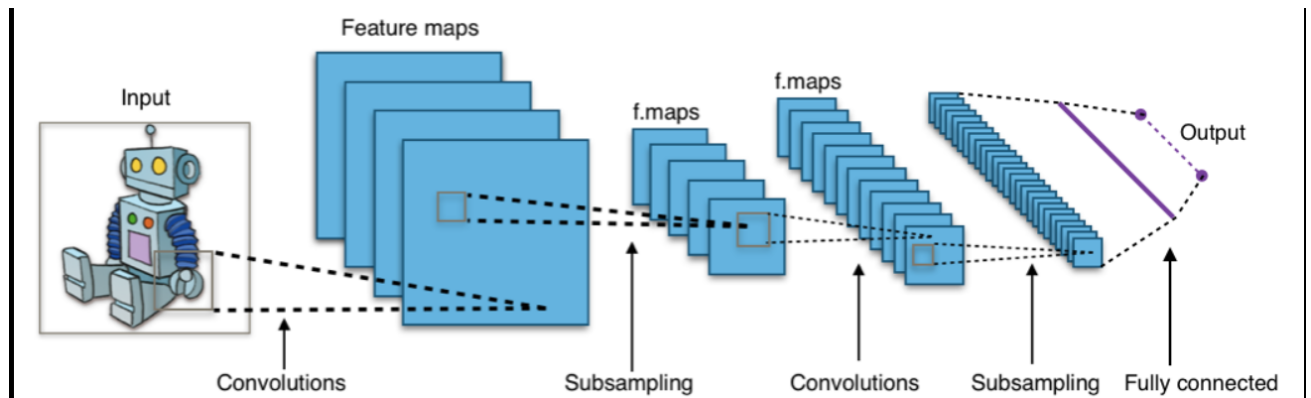


FIG 4.2 CNN ARCHITECTURE

4.4 MASK DETECTION:

Face mask detector is trained using Keras and Tensorflow. The trained COVID-19 face mask detector, proceeds to implement two more additional Python scripts used to Detect COVID-19 face masks in images Detect face masks in real-time video streams. This dataset consists of 1,376 images belonging to two classes namely `with_mask` 690 images `without_mask` 686 images.

4.4.1 DATASET CREATION

To create this dataset, the proposed project needs taking normal images of faces then creating a custom computer vision Python script to add face masks to them, thereby creating an artificial (but still real-world applicable) dataset.

4.4.2 FACIAL LANDMARKS:

This method is actually a lot easier than it sounds once you apply facial landmarks to the problem. Facial landmarks allow us to automatically infer the location of facial structures, including:

4.4.3 EYES:

The eyes display strong vertical edges(horizontal transitions) due to its iris and eye white. Thus, the sobel mask is applied to an image and the horizontal projection of vertical edges can be obtained to determine the y coordinate of the eyes.

4.4.4 EYEBROWS:

Two rectangular regions in the edge image which lies directly above each of the eye regions are selected as the eyebrow regions. The edge images of these two areas

are obtained for further refinement. Now sobel methods was used in obtaining the edge images, since it can detect more edges than Roberts method. These obtained edge images are then dilated and the wholes are filled. The result edge images are used in refining the eyebrow regions.

4.4.5 NOSE:

Facial regions can be accessed by using simple python indexing and the nose using [27,35]

4.4.6 MOUTH:

The top, bottom, rightmost and leftmost points of the mouth are been extracted and the centroid of the mouth is calculated.

4.4.7 JAWLINE

By using python indexing the jawline can be detected using index [0,17].

To use facial landmarks to build a dataset of faces wearing face masks, it is needed to first start with an image of a person not wearing a face mask



FIGURE 4.4.1 FACE WITHOUT MASK

From there, we apply face detection to compute the bounding box location of the face in the image.



FIG 4.4.2 APPLY FACE DETECT

Once we know where in the image the face is, we can extract the face Region of Interest (ROI)



FIGURE 4.4.3 EXTRACT THE FACE ROI

And from there, we apply facial landmarks, allowing us to localize the eyes, nose, mouth, etc.



FIG 4.4.4 DETECT FACIAL LANDMARKS USING DLIB

Next, we need an image of a mask (with a transparent background) such as the one below

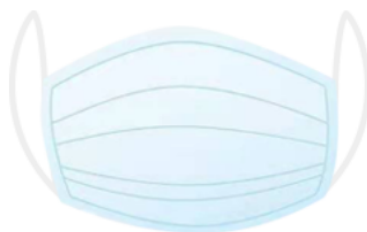


FIG 4.4.5 FACIAL MASK

This mask will be automatically applied to the face by using the

facial landmarks (namely the

points along the chin and nose) to compute where the mask will be placed.

The mask is then resized and rotated, placing it on the face:



FIG 4.4.6 MASKED FACE

We can then repeat this process for all of our input images, thereby creating our artificial face mask dataset.



FIG 4.4.7 AN ARTIFICIAL SET OF COVID-19 FACE MASK IMAGES.

This set will be part of our “with mask” / “without mask” dataset for COVID-19 face mask detection with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

CHAPTER 5

IMPLEMENTATION

MODULE DESCRIPTION

In order to train a custom face mask detector, we need to break our project into two distinct phases, each with its own respective sub-steps namely training phase and deployment phase.

5.1 TRAINING PHASE:

Here the focus is on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk

5.2 DEPLOYMENT PHASE:

Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with mask or without mask.

5.3 METHODOLOGY:

Isolate Face Shapes:

- Resulting image from neural net had regions of interests that were not true faces.
- The unique oval-shape true faces was used.
- To isolate most probable regions of interest, the test image is convolved with an oval mask.
- Split Multi-Face Images: k -Means Clustering
- Regions may contain more than one face.
- Estimate number of faces using the Distance Transform
- Use this estimate to initialize k .
- Feature vectors are (x,y) locations of each pixel in the region.
- Assign each pixel to one of k new regions.

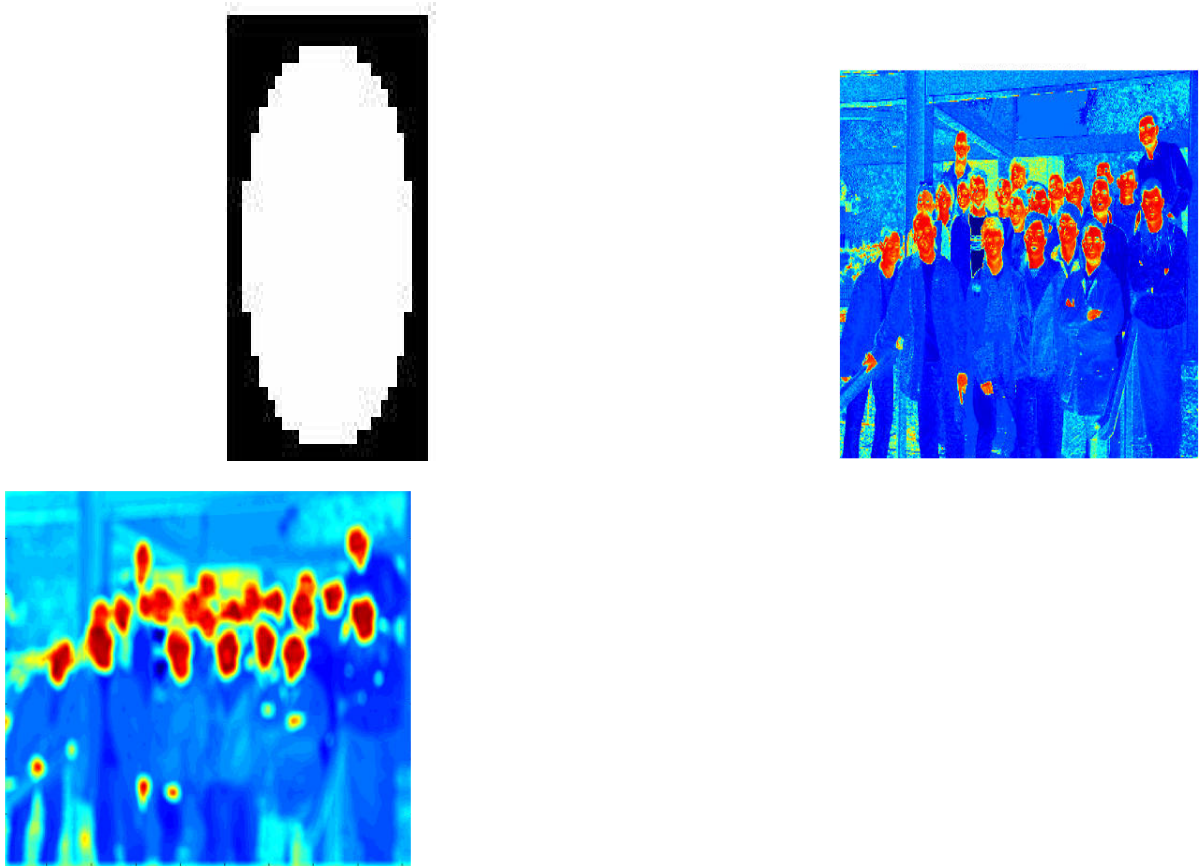


Fig 5.1 ISOLATING FACE SHAPES

5.4 MODULE IMPLEMENTATION

5.4.1 DATA PREPROCESSING:

Data processing is the method of processing the data accordingly. In this proposed project Importing opencv and loading the folder names and the dataset has two categories namely with mask and without mask. Different color, orientations images will be present so converting the image to greyscale image and resizing the image into 100 x 100. The next step that to be followed is normalizing the data, once the data is normalized the data is to be saved and the target which will contain the face with and without mask will also be saved.

5.4.2 TRAINING THE CNN:

The next step to be followed is to train the CNN(Convolutional Neural Network), it is a class of deep learning neural networks. CNN represents a huge breakthrough in image recognition. They are most commonly used to analyze visual imagery and are

frequently working behind the scenes in image classification. The proposed system has two convolutional layers having 200@3x3 and 100@3x3 layers. The layers are flattened and finally it is connected with dense layer of 50 layers. Finally, the output dense layer has two neurons that is with and without mask. The next step is to load the data. The first CNN layer followed by Relu and Max pooling layers and similarly for second layer. Since there are two categories that are going to be used categorical_crossentropy as loss function and adam optimizer. The next step is to split that data for training and testing, and then to save the best model per each epoch.

5.4.3 DETECTING MASK AND GENERATING ALERTS:

This is the final step that is to be followed, in this the first step is to load the data model and using the cascade classifier to detect faces and using opencv to capture the faces from the video and for verification, and in the proposed topic the best model obtained is having the accuracy of 0.9275. By optimizing the dataset and algorithm, the accuracy can be improved.

CHAPTER 6

RESULTS AND DISCUSSION

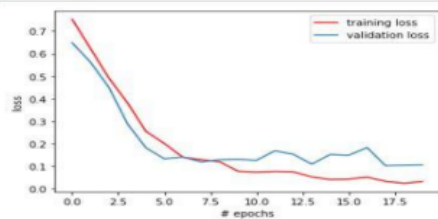
As a result of this system getting introduced, it will act as a bridge to minimize the spread of covid-19 in areas equipped with CCTV's. It will be easy for police to monitor people which will further result in taking quick actions.

```
In [5]: checkpoint = ModelCheckpoint('model-{epoch:03d}.model', monitor='val_loss', verbose=0, save_best_only=True, mode='auto')
history=model.fit(train_data, train_target, epochs=20, callbacks=[checkpoint], validation_split=0.2)
```

```
Epoch 1/20
31/31 [=====] - 50s 2s/step - loss: 0.8820 - accuracy: 0.4939 - val_loss: 0.6478 - val_accuracy: 0.645
2
INFO:tensorflow:Assets written to: model-001.model\assets
Epoch 2/20
31/31 [=====] - 47s 2s/step - loss: 0.6284 - accuracy: 0.6411 - val_loss: 0.5609 - val_accuracy: 0.729
8
INFO:tensorflow:Assets written to: model-002.model\assets
Epoch 3/20
31/31 [=====] - 49s 2s/step - loss: 0.5296 - accuracy: 0.7644 - val_loss: 0.4487 - val_accuracy: 0.754
0
INFO:tensorflow:Assets written to: model-003.model\assets
Epoch 4/20
31/31 [=====] - 49s 2s/step - loss: 0.3850 - accuracy: 0.8272 - val_loss: 0.2881 - val_accuracy: 0.911
3
INFO:tensorflow:Assets written to: model-004.model\assets
Epoch 5/20
31/31 [=====] - 51s 2s/step - loss: 0.2614 - accuracy: 0.9120 - val_loss: 0.1621 - val_accuracy: 0.935
5
INFO:tensorflow:Assets written to: model-005.model\assets
Epoch 6/20
31/31 [=====] - 50s 2s/step - loss: 0.2061 - accuracy: 0.9293 - val_loss: 0.1332 - val_accuracy: 0.947
6
INFO:tensorflow:Assets written to: model-006.model\assets
Epoch 7/20
31/31 [=====] - 47s 2s/step - loss: 0.1497 - accuracy: 0.9521 - val_loss: 0.1413 - val_accuracy: 0.939
9
Epoch 8/20
31/31 [=====] - 49s 2s/step - loss: 0.1279 - accuracy: 0.9656 - val_loss: 0.1197 - val_accuracy: 0.939
9
INFO:tensorflow:Assets written to: model-008.model\assets
Epoch 9/20
31/31 [=====] - 46s 1s/step - loss: 0.1248 - accuracy: 0.9556 - val_loss: 0.1290 - val_accuracy: 0.943
5
Epoch 10/20
31/31 [=====] - 33s 1s/step - loss: 0.0880 - accuracy: 0.9733 - val_loss: 0.1313 - val_accuracy: 0.935
5
Epoch 11/20
31/31 [=====] - 33s 1s/step - loss: 0.0715 - accuracy: 0.9742 - val_loss: 0.1267 - val_accuracy: 0.947
6
Epoch 12/20
31/31 [=====] - 35s 1s/step - loss: 0.0725 - accuracy: 0.9727 - val_loss: 0.1692 - val_accuracy: 0.907
3
Epoch 13/20
31/31 [=====] - 36s 1s/step - loss: 0.0899 - accuracy: 0.9694 - val_loss: 0.1532 - val_accuracy: 0.931
```

FIG 6.1 EPOCH VALUES

```
In [6]: from matplotlib import pyplot as plt
plt.plot(history.history['loss'],'r',label='training loss')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



```
In [7]: plt.plot(history.history['accuracy'],'r',label='training accuracy')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

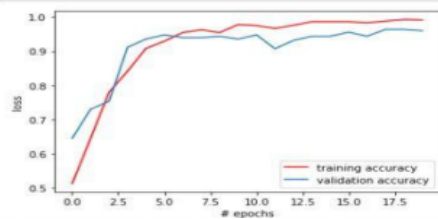


Fig 6.2 loss and accuracy plot

The figures 6.1 and 6.2 represents covid-19 face mask detector training accuracy/loss curves demonstrates high accuracy and little signs of overfitting on the data. The model is now ready to apply knowledge of computer vision and deep learning using python, opencv, tensorflow .to perform face mask detection.

This model acquires 99 percent accuracy on test data set, looking at the figure the model has little signs of overfitting, with validation loss lower than the training loss. Given results are hopeful that the model will generalize well to images outside training testing dataset.

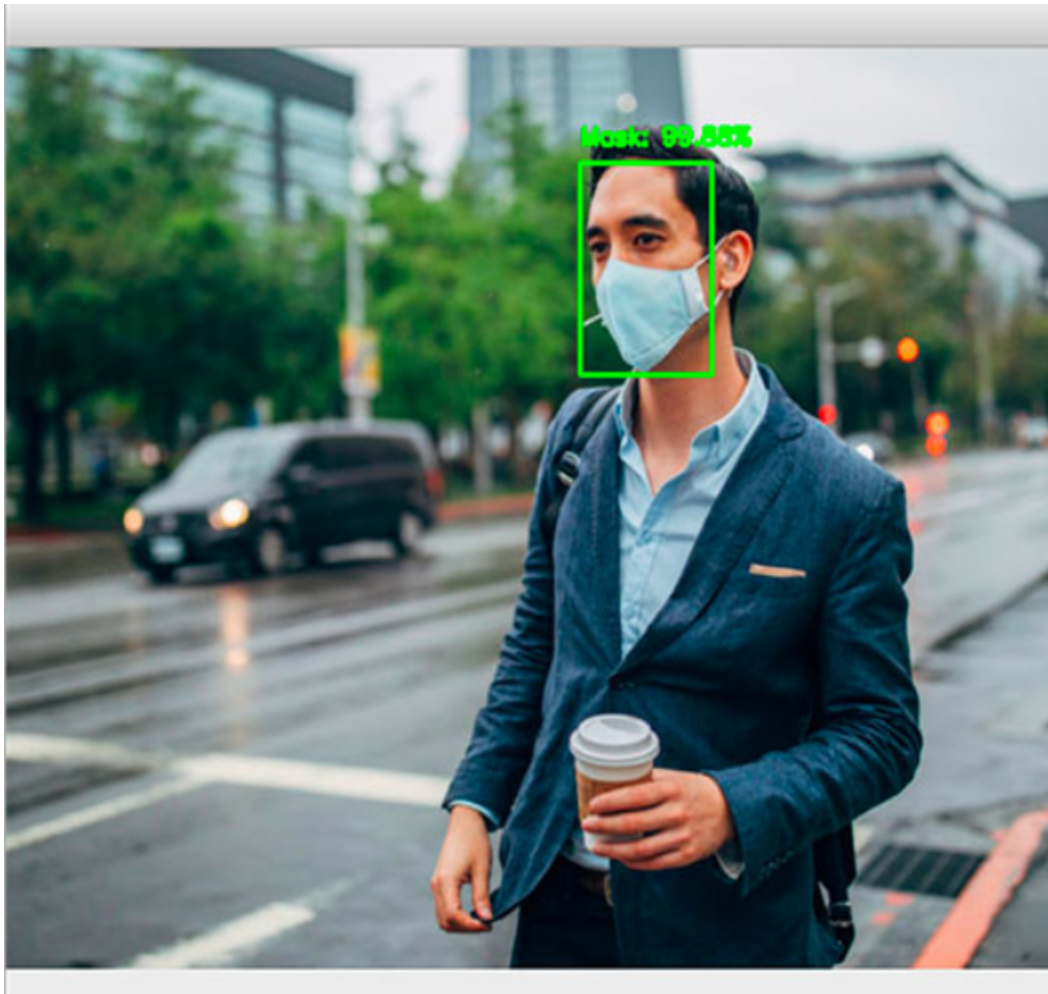


Fig 6.3 masked face

In this figure the man is wearing a COVID-19/Coronavirus face mask in public and the model's computer vision and deep learning method using Python, OpenCV, and TensorFlow/Keras has made it possible to detect the presence of the mask automatically. face mask detector correctly labeled this image as mask.

The face mask detector has recognised this persons face as masked face with accuracy above 99 percentage.

The next image is a person without mask.



Fig 6.3 no mask face

This person face is without mask , not wearing a COVID-19 face mask in this picture. Using Python, OpenCV, and TensorFlow/Keras, the system has correctly detected "No Mask" for this face. face mask detector has correctly predicted no mask.



FIG 6.4 MASK AND NO MASK FACE

Why is it that the system were able to detect the faces of the two gentlemen in the background and correctly classify mask/no mask for them, but could not detect the woman in the foreground?

Keep in mind that in order to classify whether or not a person is wearing in mask, first need to perform face detection — if a face is not found which is what happened in this image, then the mask detector cannot be applied.

The reason that the system cannot detect the face in the foreground is because:

1. It's too obscured by the mask
2. The dataset used to train the *face detector* did not contain example images of people wearing face masks
3. Therefore, if a large portion of the face is occluded, face detector will likely fail to detect the face.

6.1 SUGGESTIONS FOR IMPROVEMENT

from the results sections above, face mask detector is working quite well despite:

1. Having limited training data
2. The `with_mask` class being artificially generated.

To improve face mask detection model further, **should gather actual images (rather than artificially generated images) of people wearing masks**. While artificial dataset worked well in this case, there's no substitute for the real thing. Secondly, should also gather images of faces that may "confuse" the classifier into thinking the person is wearing a mask when in fact they are not — potential examples include shirts wrapped around faces, bandana over the mouth, etc. All of these are examples of something that *could* be confused as a face mask by face mask detector. Finally, should consider training a dedicated two-class *object detector* rather than a simple image classifier.

Our current method of detecting whether a person is wearing a mask or not is a two-step process:

1. step #1: Perform face detection
2. Step #2: Apply face mask detector to each face

The problem with this approach is that a face mask, by definition, obscures part of the face. If enough of the face is obscured, the face cannot be detected, and therefore, the face mask detector will not be applied. To circumvent that issue, should train a two-class object detector that consists of a `with_mask` class and `without_mask` class. Combining an object detector with a dedicated `with_mask` class will allow improvement of the model in two respects. First, the object detector will be able to naturally detect people wearing masks that otherwise would have been impossible for the face detector to detect due to too much of the face being obscured. Secondly, this approach reduces our computer vision pipeline to a single step — rather than applying face detection and *then* our face mask detector model, all we need to do is apply the object detector to give us bounding boxes for people both `With_mask` and `without_mask` in a single

forward pass of the network. Not only is such a method more computationally efficient, it's also more “elegant” and end-to-end.

CHAPTER 7

CONCLUSION

This system can therefore be used in real-time applications which require face-mask detection for safety purposes due to the outbreak of Covid-19. This project can be integrated with embedded systems for application in airports, railway stations, offices, schools, and public places to ensure that public safety guidelines are followed.

7.1 DRAWBACKS:

7.1.1. POOR IMAGE QUALITY LIMITS FACIAL RECOGNITION'S EFFECTIVENESS:

Image quality affects how well facial-recognition algorithms work. The image quality of scanning video is quite low compared with that of a digital camera. Even high-definition video is, at best, 1080p (progressive scan); usually, it is 720p. These values are equivalent to about 2MP and 0.9MP, respectively, while an inexpensive digital camera attains 15MP. The difference is quite noticeable.

7.1.2. SMALL IMAGE SIZES MAKE FACIAL RECOGNITION MORE DIFFICULT

When a face-detection algorithm finds a face in an image or in a still from a video capture, the relative size of that face compared with the enrolled image size **affects how well the face will be recognized**. An already small image size, coupled with a target distant from the camera, means that the detected face is only 100 to 200 pixels on a side. Further, having to scan an image for varying face sizes is a processor-intensive activity. Most algorithms allow specification of a face-size range to help eliminate false positives on detection and speed up image processing.

7.1.3. DIFFERENT FACE ANGLES CAN THROW OFF FACIAL RECOGNITION'S RELIABILITY

The relative angle of the target's face influences the recognition score profoundly. When a face is enrolled in the recognition software, usually multiple angles are used (profile, frontal and 45-degree are common). Anything less than a frontal view affects the algorithm's capability to generate a template for the face. The more direct the image (both enrolled and probe image) and the higher its resolution, the higher the score of any resulting matches.

7.1.4. DATA PROCESSING AND STORAGE CAN LIMIT FACIAL RECOGNITION TECH:

Even though high-definition video is quite low in resolution when compared with digital camera images, it still occupies significant amounts of disk space. **Processing every frame of video is an enormous undertaking**, so usually

only a fraction (10 percent to 25 percent) is actually run through a recognition system. To minimize total processing time, agencies can use clusters of computers. However, adding computers involves considerable data transfer over a network, which can be bound by input-output restrictions, further limiting processing speed. Ironically, humans are vastly superior to technology when it comes to facial recognition. But humans can only look for a few individuals at a time when watching a source video. A computer can compare many individuals against a database of thousands.

REFERENCE

- [1] J Tang,Xia Hu,Huiji Gao,Huan Liu," Unsupervised Feature Selection for Multi-View Data in Social Media",Proceedings of the ACM international conference on Information and Knowledge management (2018),pp.1673-1678.

- [2] Haytham Elzhazel and Alex Auseem, "Unsupervised Feature Selection with ensemble learning ", Machine Learning ,Springer (2019).
- [3] Mohak Shah, Mario Marchand, and Jacques Corbeil, " Feature Selection with Conjunctions of Decision Stumps and Learning ", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 34, NO. 1,(2018) pp.174-186
- [4] Qingzheng XU, Lei WANG, Baomin HE, Na WANG," Modified Opposition-Based Differential Evolution for Function Optimization", Journal of Computational Information Systems 7:5 pp.1582-1591.
- [5] J Tang,Xia Hu,Huiji Gao,Huan Liu," Unsupervised Feature Selection for Multi-View Data in Social Media",Proceedings of the ACM international conference on Information and Knowledge management (2015),pp.1673-1678
- [6] Haytham Elzhazel and Alex Auseem, "Unsupervised Feature Selection with ensemble learning ", Machine Learning ,Springer (2013).
- [7] Mohak Shah, Mario Marchand, and Jacques Corbeil, " Feature Selection with Conjunctions of Decision Stumps and Learning ", IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 34, NO. 1,(2012) pp.174-186.
- [8] Qingzheng XU, Lei WANG, Baomin HE, Na WANG," Modified Opposition-Based Differential Evolution for Function Optimization", Journal of Computational Information Systems 7:5 (2011) pp.1582-1591.
- [9] Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C....Ghemawat S. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.
- [10] Ahmed I., Ahmad M., Rodrigues J.J., Jeon G., Din S. A deep learning-based social distance monitoring framework for COVID-19. *Sustainable Cities and Society*. 2020

- [11] Alom M.Z., Taha T.M., Yakopcic C., Westberg S., Sidike P., Nasrin M.S....Asari V.K. 2018. The history began from alexnet: A comprehensive survey on deep learning approaches. arXiv preprint arXiv:1803.01164.
- [12] Anisimov D., Khanova T. Towards lightweight convolutional neural networks for object detection. 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS); IEEE; 2017. pp. 1–8. August.
- [13] Chen D., Ren S., Wei Y., Cao X., Sun J. *European conference on computer vision*. Springer; Cham: 2014. Joint cascade face detection and alignment; pp. 109–122. September.
- [14] Chen D., Hua G., Wen F., Sun J. *European conference on computer vision*. Springer; Cham: 2016. Supervised transformer network for efficient face detection; pp. 122–138. October.
- [15] Ge S., Li J., Ye Q., Luo Z. Detecting masked faces in the wild with lle-cnns. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017:2682–2690.
- [16] Ge X.Y., Pu Y., Liao C.H., Huang W.F., Zeng Q., Zhou H....Chen H.L. Evaluation of the exposure risk of SARS-CoV-2 in different hospital environment. *Sustainable Cities and Society*. 2020;61
- [17] Ghiasi G., Fowlkes C.C. Occlusion coherence: Localizing occluded faces with a hierarchical deformable part model. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014:2385–2392.
- [18] Gulcehre C., Moczulski M., Denil M., Bengio Y. *International conference on machine learning*. 2016. Noisy activation functions; pp. 3059–3068. June.
- [19] Hahnloser R.H., Sarpeshkar R., Mahowald M.A., Douglas R.J., Seung H.S. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*. 2000;405(6789):947–951.

- [20] He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016:770–778.
- [21] Huang C., Ai H., Li Y., Lao S. High-performance rotation invariant multiview face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2007;29(4):671–686.
- [22] Jia Y., Shelhamer E., Donahue J., Karayev S., Long J., Girshick R....Darrell T. Caffe: Convolutional architecture for fast feature embedding. *Proceedings of the 22nd ACM international conference on multimedia*. 2014:675–678. November.
- [23] Jones P., Viola P., Jones M. University of Rochester; Charles Rich: 2001. Rapid object detection using a boosted cascade of simple features.
- [24] Jun B., Choi I., Kim D. Local transform features and hybridization for accurate face and human detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2012;35(6):1423–1436.
- [25] Kim T.H., Park D.C., Woo D.M., Jeong T., Min S.Y. *International conference on intelligent science and intelligent data engineering*. Springer; Berlin, Heidelberg: 2011. Multi-class classifier-based adaboost algorithm; pp. 122–127. October.
- [26] Klare B.F., Klein B., Taborsky E., Blanton A., Cheney J., Allen K....Jain A.K. Pushing the frontiers of unconstrained face detection and recognition: larpa janus benchmark a. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015:1931–1939.
- [27] Klare B.F., Klein B., Taborsky E., Blanton A., Cheney J., Allen K....Jain A.K. Pushing the frontiers of unconstrained face detection and recognition: larpa janus benchmark a. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015:1931–1939.
- [28] Kumar P., Hama S., Omidvarborna H., Sharma A., Sahani J., Abhijith K.V....Tiwari A. Temporary reduction in fine particulate matter due to ‘anthropogenic

emissions switch-off' during COVID-19 lockdown in Indian cities. *Sustainable Cities and Society*. 2020;62

[29] Lawrence S., Giles C.L., Tsoi A.C., Back A.D. Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*. 1997;8(1):98–113.

[30] LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998;86(11):2278–2324.

[31] Li H., Lin Z., Shen X., Brandt J., Hua G. A convolutional neural network cascade for face detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015:5325–5334.

[32] Li Y., Sun B., Wu T., Wang Y. *European conference on computer vision*. Springer; Cham: 2016. Face detection with end-to-end integration of a convnet and a 3d model; pp. 420–436. October.

APPENDIX

A SOURCECODE

TRAINING

USAGE

python train_mask_detector.py --dataset dataset

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 argparse

import os

# construct the argument parser and parse the arguments

ap = argparse.ArgumentParser()

ap.add_argument("-d", "--dataset", required=True,

help="path to input dataset")

ap.add_argument("-p", "--plot", type=str, default="plot.png",

help="path to output loss/accuracy plot")

ap.add_argument("-m", "--model", type=str,

default="mask_detector.model",

help="path to output face mask detector model")

args = vars(ap.parse_args())

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

# and batch size

INIT_LR = 1e-4

EPOCHS = 20

BS = 32

# 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...")

imagePaths = list(paths.list_images(args["dataset"]))

data = []

labels = []

# loop over the image paths
```

```
for imagePath in imagePaths:

    # extract the class label from the filename

    label = imagePath.split(os.path.sep)[-2]

    # load the input image (224x224) and preprocess it

    image = load_img(imagePath, target_size=(224, 224))

    image = img_to_array(image)

    image = preprocess_input(image)

    # update the data and labels lists, respectively

    data.append(image)

    labels.append(label)

# convert the data and labels to NumPy arrays

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

labels = np.array(labels)

# perform one-hot encoding on the labels

lb = LabelBinarizer()

labels = lb.fit_transform(labels)

labels = to_categorical(labels)

# partition the data into training and testing splits using 75% of

# the data for training and the remaining 25% for testing

(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(args["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(args["plot"])
```

DETECT MASK IN IMAGE

USAGE

```
# python detect_mask_image.py --image examples/example_01.png
```

```
# 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
```

```
import numpy as np
```

```
import argparse
```

```
import cv2
```

```
import os
```

```
# construct the argument parser and parse the arguments
```

```
ap = argparse.ArgumentParser()
```

```
ap.add_argument("-i", "--image", required=True,
help="path to input image")

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"])

net = cv2.dnn.readNet(prototxtPath, weightsPath)

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

model = load_model(args["model"])

# load the input image from disk, clone it, and grab the image spatial
# dimensions
image = cv2.imread(args["image"])

orig = image.copy()

(h, w) = image.shape[:2]
```

```
# construct a blob from the image

blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
(104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections

print("[INFO] computing face detections...")

net.setInput(blob)

detections = net.forward()

# 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 = 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)

# pass the face through the model to determine if the face
# has a mask or not

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

# 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(image, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)

cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)

# show the output image

cv2.imshow("Output", image)

cv2.waitKey(0)

```

DETECT MASK IN VIDEO

```
# USAGE
```

```
# 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]
        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)

# 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)

    # 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()

```

B SCREENSHOTS

```
In [5]: checkpoint = ModelCheckpoint('model-{epoch:03d}.model', monitor='val_loss', verbose=0, save_best_only=True, mode='auto')
history=model.fit(train_data, train_target, epochs=20, callbacks=[checkpoint], validation_split=0.2)
```

```
Epoch 1/20
31/31 [=====] - 50s 2s/step - loss: 0.8820 - accuracy: 0.4939 - val_loss: 0.6478 - val_accuracy: 0.645
2
INFO:tensorflow:Assets written to: model-001.model/assets
Epoch 2/20
31/31 [=====] - 47s 2s/step - loss: 0.6284 - accuracy: 0.6411 - val_loss: 0.5609 - val_accuracy: 0.729
8
INFO:tensorflow:Assets written to: model-002.model/assets
Epoch 3/20
31/31 [=====] - 49s 2s/step - loss: 0.5296 - accuracy: 0.7644 - val_loss: 0.4487 - val_accuracy: 0.754
0
INFO:tensorflow:Assets written to: model-003.model/assets
Epoch 4/20
31/31 [=====] - 49s 2s/step - loss: 0.3850 - accuracy: 0.8272 - val_loss: 0.2881 - val_accuracy: 0.911
3
INFO:tensorflow:Assets written to: model-004.model/assets
Epoch 5/20
31/31 [=====] - 51s 2s/step - loss: 0.2614 - accuracy: 0.9120 - val_loss: 0.1821 - val_accuracy: 0.935
5
INFO:tensorflow:Assets written to: model-005.model/assets
Epoch 6/20
31/31 [=====] - 50s 2s/step - loss: 0.2061 - accuracy: 0.9293 - val_loss: 0.1332 - val_accuracy: 0.947
6
INFO:tensorflow:Assets written to: model-006.model/assets
Epoch 7/20
31/31 [=====] - 47s 2s/step - loss: 0.1497 - accuracy: 0.9521 - val_loss: 0.1413 - val_accuracy: 0.939
5
Epoch 8/20
31/31 [=====] - 49s 2s/step - loss: 0.1279 - accuracy: 0.9656 - val_loss: 0.1197 - val_accuracy: 0.939
5
INFO:tensorflow:Assets written to: model-008.model/assets
Epoch 9/20
31/31 [=====] - 46s 1s/step - loss: 0.1248 - accuracy: 0.9556 - val_loss: 0.1290 - val_accuracy: 0.943
5
Epoch 10/20
31/31 [=====] - 33s 1s/step - loss: 0.0880 - accuracy: 0.9733 - val_loss: 0.1313 - val_accuracy: 0.935
5
Epoch 11/20
31/31 [=====] - 33s 1s/step - loss: 0.0715 - accuracy: 0.9742 - val_loss: 0.1267 - val_accuracy: 0.947
6
Epoch 12/20
31/31 [=====] - 35s 1s/step - loss: 0.0725 - accuracy: 0.9727 - val_loss: 0.1692 - val_accuracy: 0.907
3
Epoch 13/20
31/31 [=====] - 36s 1s/step - loss: 0.0899 - accuracy: 0.9694 - val_loss: 0.1532 - val_accuracy: 0.931
```

FIG 7.1 EPOCH

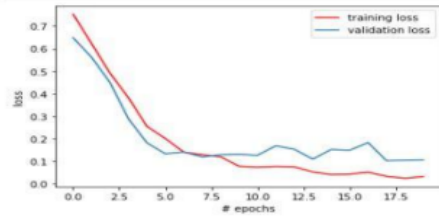
```
In [8]: print(model.evaluate(test_data, test_target))
```

```
5/5 [=====] - 1s 210ms/step - loss: 0.1829 - accuracy: 0.9275
[0.18294985592365265, 0.9275362491607666]
```

FIG 7.2 ACCURACY TEST

```
In [6]: from matplotlib import pyplot as plt

plt.plot(history.history['loss'],'r',label='training loss')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



```
In [7]: plt.plot(history.history['accuracy'],'r',label='training accuracy')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```

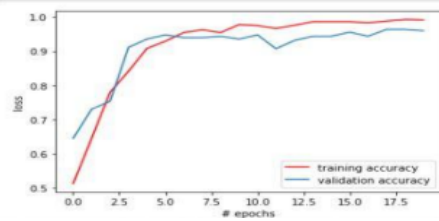


FIG 7.3 ACCURACY PLOT

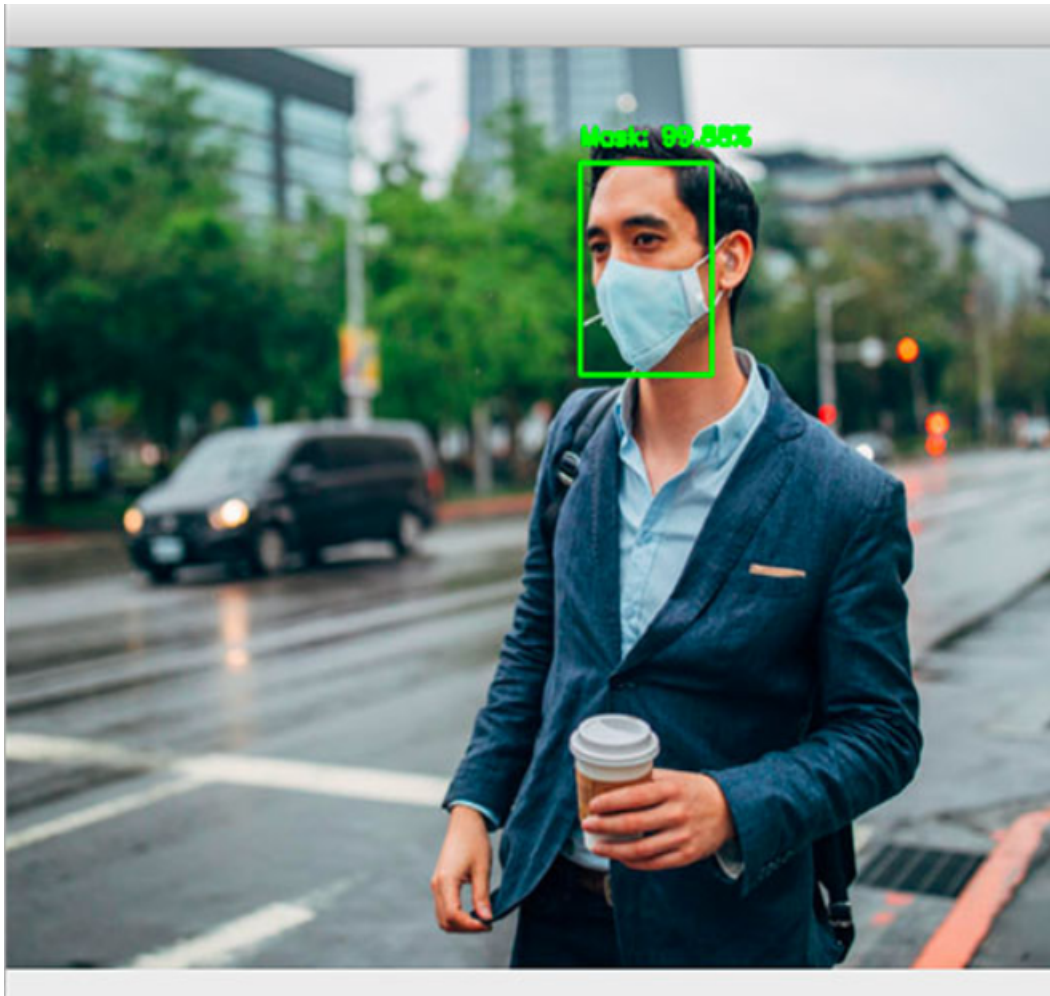


FIG 7. 4 MASKED FACE

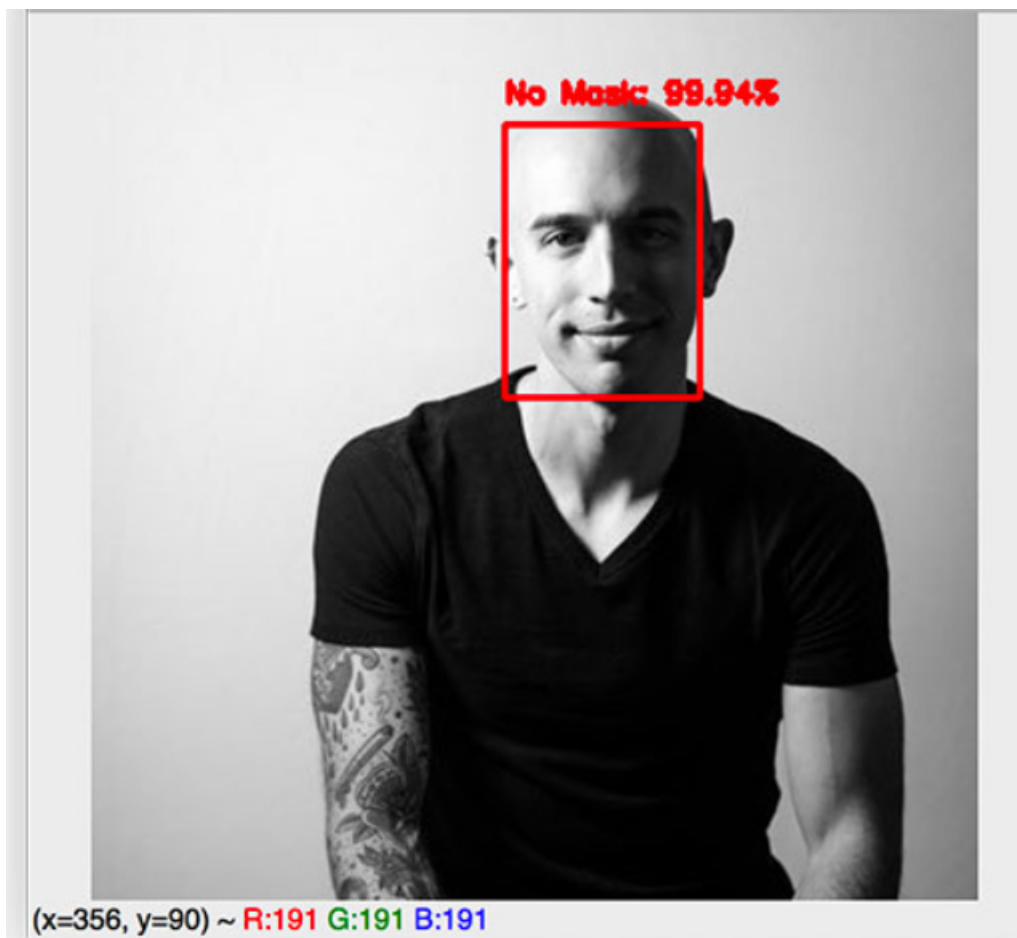


FIG 7.5 NO MASK FACE