IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 8, Issue 2, April - May, 2020
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

RASPBERRY PI BASED COVID-19 FACE MASK DETECTOR WITH OPENCV, KERAS/TENSORFLOW AND DEEP LEARNING

[1] Kalangi Balasubramanyam1, [2] S N S Krishna Kanth2, [3] Dontabhaktuni Jayakumar3

[1] Assistant professor, Department of Electronics and Communication Engineering, Swarna Bharathi Institute of Science & Technology, Khammam, Telangana-507002.

[2] Assistant professor, Department of Electrical and Electronics Engineering, Khammam institute of technology and science, Khammam, Telangana.

[3] Assistant professor, Department of Electronics and Communication Engineering, Marri Laxman Reddy Institute of Technology and Management, Hyderabad- 5000433. Telangana.

, ,

Abstract

In this paper a Raspberry Pi implementation of COVID-19 face mask detection has been proposed. This is implemented using Python Programming with OpenCV Library and keras/TensorFlow (An end-to-end open source machine learning platform). The main Idea of this Implementation is to detect the persons without wearing a Face mask in Public places. The system takes images of people, analyse, detect and recognize human faces with mask and without mask using image processing algorithms. The system can serve as a security or surveillance system in public places like Malls, Universities, and airports. It can detect and recognize persons without a face mask in Public places. We'll use Python script to train a face mask detector and review the results

Keywords— COVID-19, Face mask detection, face recognition, raspberry Pi, Python, OpenCV, ***Keras/TensorFlow.***

I. INTRODUCTION

The main Idea behind this paper is to overcome the Risk of Spread of COVID-19 to prevent infection and to slow down the transmission of COVID-19 we need to maintain at least 1 metre distance with people coughing or sneezing, need to avoid touching our face, so we need to cover our mouth and nose when coughing or sneezing [10]. So every person need to wear a Mask to cover face with a Face Mask. So wearing Face mask is mandatory in this COVID-19 outbreak [9].

Computer-based face detection and recognition systems are rapidly spreading in various sectors. The goal of this research is to build an embedded system that can detect and recognize persons without face mask using image-processing techniques. Practically, this idea can be implemented in large places to avoid speeding of COVID-19 virus. The benefits of this system are:

1. Expand the desired microcontroller capabilities.
2. Implementing Machine Learning algorithms on microcontrollers and observing results.
3. Implement the face mask detection and recognition of persons without face mask algorithms to run over the microcontroller.

II.RELATED WORK

In [1], a face recognition system using Raspberry Pi was developed. Here author used OpenCV for Face recognition. In [2], A Real time face detection and Face recognition using OpenCV on Raspberry Pi was developed. In [3], a Face mask detection System was Implemented using Python Programming, OpenCV, Keras and Tensor Flow here the total process is implemented on Linux System but in this Paper by taking these references we designed a Microcontroller based Embedded system to detect a Face Mask using Raspberry pi.

III.RESEARCH METHODOLOGY AND SOLUTION

The purpose of this paper is mainly to implement the face mask detection and recognition of faces without mask. The implementation is done by using Raspberry Pi with camera for capturing images and python programming using OpenCV [4] library image processing is performed. Keras and TensorFlow are used for implementing Machine Learning algorithm.

IV.HARDWARE IMPLEMENTATION

The block diagram in Figure1 consists of a pi camera or web camera that will capture and forward frames to the Raspberry Pi, the Raspberry Pi then will detect and crop the

faces in this frame and By using Python and TensorFlow it is Processed, we can observe the Result of machine learning algorithm on a Monitor connected to Raspberry pi.

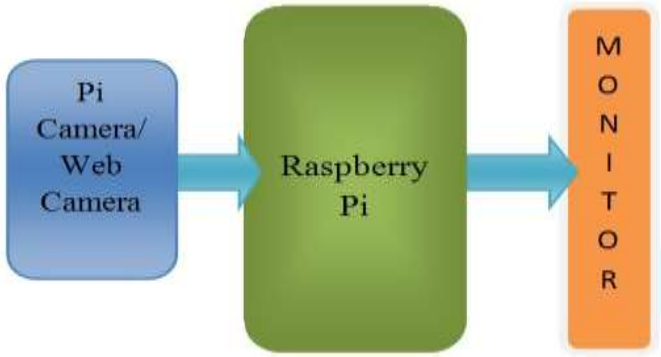

Figure 1. General System Block Diagram.

The system works as follows:

- A camera is connected to the Raspberry Pi will stream live video or Captures Images.
- Faces of people should be detected by Python Script using OpenCV, and Process them based on the Script. This is done by implementing detection algorithm on the Raspberry Pi.
- By using the sample images we train the system by providing image datasets of with mask and without mask
- We apply each image to this trained classifier this algorithm compares the input image with the trained dataset and we get result on screen.
- If the face mask is detected then it can be shown by green mark otherwise a red mark is shown on Screen.

V.SOFTWARE IMPLEMENTATION

In this paper, enforced two-phase COVID-19 mask detector, particularisation however our computer vision/deep learning pipeline are enforced. From there, we have a tendency to review the dataset we have a tendency to victimisation to train our custom mask detector. To implement this we have a tendency to use Python script to train a mask detector on our dataset victimisation using Keras and Tensor Flow. We use this Python script to train a mask detector and review the results. With this trained COVID-19 mask detector, we proceed to implement 2 Python scripts used to:

1. Detect COVID-19 face masks in pictures
2. Detect face masks in period video streams

VI.SYSTEM IMPLEMENTATION

The total system is divided into two parts In order to train a custom face mask detector as shown in figure 2, each with its own respective sub-steps that is:

1. **Training:** In this Stage we focus on loading our face mask detection set of Images from memory and then we need to train a model using TensorFlow on these set of Images, and then serializing the face mask detector to storage.

2. **Deployment:** After training face mask detector, we then move on to loading the mask detector, performing face detection, and then classifying each face as **withmask** or **withoutmask**

In the First Phase By loading facemask dataset we train the face mask classifier by using keras/TensorFlow and then we serialize the face mask classifier to memory.

In the Second Phase we load this classifier from memory and now we apply image or video stream as an input. From this it extracts each face ROI. By applying face mask classification to each face ROI we can determine ‘withmask’ or ‘withoutmask’.

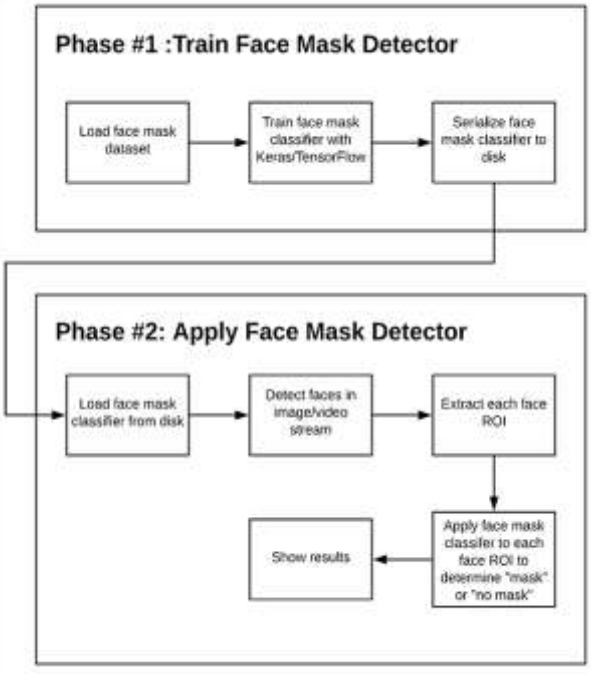

Figure 2**:** System Implementation of COVID-19 face mask detector.

Let’s take a look at the dataset we using to train our COVID-19 face mask detector.

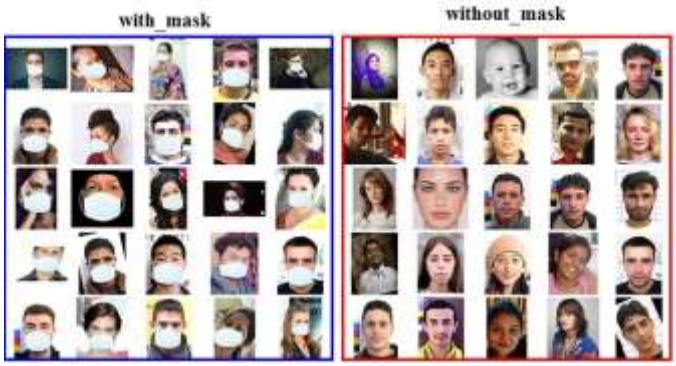

Figure 3**:** A face mask detection dataset consists of "withmask" and "withoutmask" images.

The dataset to build a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV and TensorFlow is shown in figure 3. The above mentioned dataset consists of 1,376 numbers of images belonging to following classes:

- withmask: 690 images
- without mask: 686 images

Our aim is to train a custom deep learning model to detect whether a person is wearing a mask or not. To create this dataset:

1. Normal images of faces are needed to take.
2. Then we need to add masks by creating a custom computer vision Python script, thereby creating an artificial dataset

This method is very easier than it sounds once we apply facial landmarks to the problem. Facial landmarks allow us to automatically conclude the location of facial structures, including:

- Eyes
- Eyebrows
- Nose
- Mouth
- Jaw line

To obtain facial landmarks to build a dataset of faces wearing face masks, we need to first start with an image of a person not wearing a face mask as shown in figure 4.

Figure 4: A photograph of someone not wearing a face.

From there, we apply face detection to compute the bounding box location of the face in the image shown in fig 5:

Figure 5: After applying face detection with OpenCV.

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

Figure 6**:** extract the face ROI with OpenCV and NumPy slicing.

And from there, we apply facial landmarks, allowing us to localize the eyes, nose, mouth, etc. As shown in below fig 7.

Figure 7**:** facial landmarks using dlib

Next, we need an image of a mask (with a transparent background) shown in figure 8.

Figure 8**:** a COVID-19/Corona virus face mask/shield.

This face mask will be overlaid on the original face ROI automatically since we know the face landmark locations. 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 which is shown in figure 9.

Figure 9**:** Face mask is placed on the person's Image

It is not an easy task to tell at an instance that the COVID-19 mask has been applied with computer vision by way of OpenCV and dlib face landmarks [8]. Then we repeat this process for all the input images, thereby creating artificial face mask dataset as shown in Figure 10.

Figure 10: An artificial set of COVID-19 face mask images**.**

The above set will be subset of our "withmask"/"without mask" dataset for face mask detection with computer vision and deep learning using Python, OpenCV, Keras and TensorFlow.

To implement this following Python scripts are used:

- TrainMaskDetector.py: It accepts input dataset and fine-tunes MobileNetV2 upon it to create Mask detector model. A training history plot1.png containing accuracy and loss curves is also produced

- DetectMaskImage.py: It Performs face mask detection in static images

- DetectMaskVideo.py: If we used webcam, this script applies face mask detection to every frame in the video stream

Now we reviewed our face mask dataset, after that use Tensor Flow to train a classifier to automatically finds whether a person is wearing a mask or not. To accomplish this task, we will be fine-tuning the MobileNet V2 architecture, a highly efficient architecture that can be applied to embedded devices with limited computational capacity here I am using Raspberry Pi, Deploying face mask detector to embedded devices could reduce the cost of manufacturing such face mask detection systems, hence why we choose to use this architecture.

Train_mask_detector.py file, imports for our training script may look intimidating to our set of TensorFlow. Imports allow for:

- Data augmentation
- Loading the MobilNetV2 classifier (we will fine-tune this model with pre-trained ImageNet weights)
- Building a new fully-connected (FC) head
- Pre-processing
- Loading image data we use scikit-learn (sklearn) for binarizing class labels, segmenting our dataset, and printing a classification report.

Fine-tuning setup is a three-step process:

1. Load MobileNet with pre-trained ImageNet weights, leaving off head of network (Lines 88 and 89)
2. Construct a new FC head, and append it to the base in place of the old head (Lines 93-102)
3. Freeze the base layers of the network (Lines 106 and 107). The weights of these base layers will not be updated during the process of back propagation, whereas the head layer weights will be tuned.

Fine-tuning is a strategy always recommends establishing a baseline model while saving considerable time. With our data prepared and model architecture in place for fine-tuning,

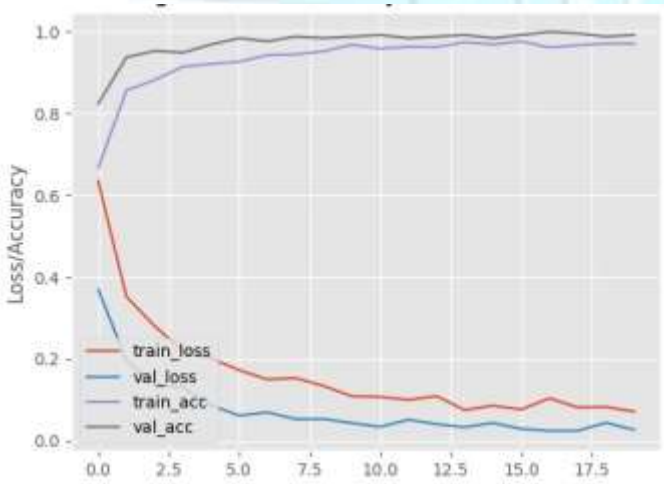

Figure 11(a): face mask detector training accuracy/loss curves

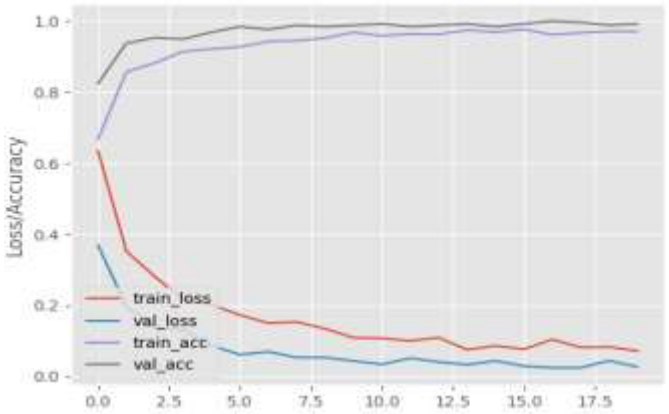

Figure 11(b): face mask detector training accuracy/loss curves

The above figures 11(a) and (b) shows COVID-19 face mask detector training accuracy/loss curves demonstrate high accuracy and little signs of over fitting on the data. We're now ready to apply our knowledge of computer vision and deep learning using Python, OpenCV, and TensorFlow to perform face mask detection. We can see, we are obtaining ~99% accuracy on our test set. Looking at Figure 10, we can see there are little signs of over fitting, with the validation loss lower than the training loss given these results, we are hopeful that our model will generalize well to images outside our training and testing set.

VII.RESULTS

Implementing our COVID-19 face mask detector for images with OpenCV that is our face mask detector is trained, let's learn how we can:

1. Load an input image from disk
2. Detect faces in the image
3. Apply our face mask detector to classify the face as either with_mask or without_mask

Figure 12: Detecting presence of the mask automatically

The Figure 12 shows is this man wearing a COVID-19 face mask in public? Computer vision and deep learning method using Python, OpenCV, and TensorFlow has made it possible to detect the presence of the mask automatically. As we can see, our face mask detector correctly labeled this image as Mask.

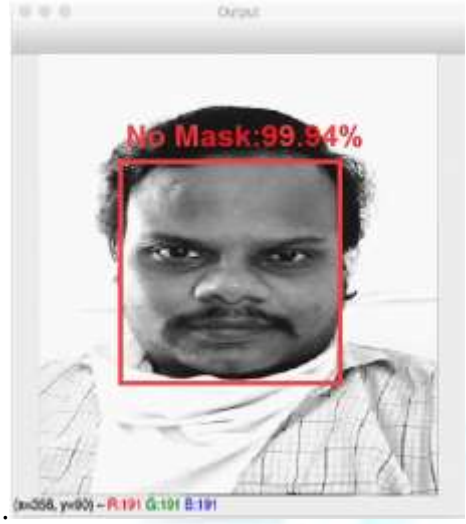

Figure 13: Person not wearing a COVID-19 face mask

From the above figure 13 we can say that the face mask detection system has correctly detected "No Mask".

Figure 14: Face Mask Detector Failed to detect person with Mask

In some cases this algorithm will not give correct result for an example let us consider above figure 14, here the model was able to detect the faces of the two gentlemen in the background and correctly classify mask/no mask for them, but it fails to detect the woman in the foreground. In order to classify whether a person is wearing a mask or not, we first need to perform face detection. If a face is not found then the mask detector cannot be applied. The above image falls under this category.
The reasons for not detecting the face in the foreground are:

1. It is too obscured by the mask
2. The dataset used to train the face detector did not contain example images of people wearing face masks

Therefore, if a large portion of the face is covered by mask, our face detector will likely fail to detect the face.

VIII.CONCLUSION

As we can see from the results sections above, our face mask detector is working quite well despite:

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

To improve our face mask detection model further, we should gather actual images (rather than artificially generated images) of people wearing masks. While our artificial dataset worked well in this case, there's no substitute for the real thing.

Secondly, we should also gather images of faces that may "confuse" our classifier into thinking the person is wearing a mask when in fact they are not — potential examples include shirts wrapped around faces, hankie over the mouth, etc.

All of these are examples of something that could be confused as a face mask by our face mask detector. Finally, you 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 our 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, we should train a two-class object detector that consists of a withmask class and without mask class. Combining an object detector with a dedicated withmask 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 withmask and withoutmask in a single forward pass of the network.

REFERENCES

[1] Face Recognition Using Raspberry Pi By Biswajit Das an Article in Electronics for u.com Online technical documentation. Available at: https://www.electronicsforu.com/electronicsprojects/face-recognition-using-raspberry-pi.

[2] Real Time Face Recognition with Raspberry Pi and OpenCV ByAswinth Raj Online technical documentation. Available at: https://circuitdigest.com/microcontrollerprojects/raspberry-pi-and-opencv-based-face-recognition-system.

[3] COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-kerastensorflow-and-deep-learning/.

[4] T. Shakunaga , and K. Shigenari,"Decomposed Eigen-Face for face recognition under various lighting conditions" . IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.

[5] FaceDetection using Haar Feature-based Cascade Classifiers in OpenCV. Online technical documentation. Available at https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html.

[6] Ping Hsin Lee, Vivek Srinivasan, and Arvind Sundararajan. Face Detection, Final Year Project, Stanford University, 2014.

[7] Local Binary Pattern Histograms in OpenCV (Open Source Computer Vision Library) Online technical documentation. Available at https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms-in-opencv.

[8] Detect Multi Scale Function in OpenCV (Open Source Computer Vision Library). Online technical documentation. Available at https://docs.opencv.org/trunk/d1/de5/classcv_1_1CascadeClassifier.html.

[9] Corona virus disease (COVID-19) advice for the public: When and how to use masks: https://www.who.int/emergencies/diseases/novelcoronavirus-2019/advice-for-public/when-and-how-to use-masks.

[10] Centers for disease control and prevention (CDC) an article on How to protect yourself & others: https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/prevention.html.