Maskit: COVID-19 Resiliency through Computer Vision and Robotics

Allen Mao
University of California, Berkeley
Berkeley, California
allenmao@berkeley.edu

Hans Gundlach
University of California, Berkeley
Berkeley, California
11235hans@berkeley.edu

CONTENTS

I. Introduction

The COVID-19 pandemic has led to, at the time of this writing, almost half a million deaths [1], the worst global recession since the Great Depression [2], and school closures that have affected nearly all of the world's student population [3]. Although reopening efforts have been underway in many parts of the world, their results have been mixed and it is often instead in citizens' own hands to protect themselves from this virus through means such as social distancing, frequent handwashing, and mask wearing. Although the United States Centers for Disease Control and Prevention (CDC) has issued recommendations for citizens to wear masks to reduce exposure to the virus, the choice to wear a mask has unfortunately evolved to become a political question too [4], [5]. Apart from CDC[1] and World Health Organization (WHO)[2] guidelines to wear facial coverings, i.e. masks, in this article we will not discuss the merits of face coverings; this knowledge is assumed. Instead, we propose, describe, and demonstrate *Maskit*, a computer vision and robotics system that keeps business owners and other ordinary citizens safe by blocking people who fail to wear a mask while letting people who do through entrances.

A. Code Source

Our work is found at https://github.com/hansgundlach/FaceMaskDetectionRasPi. It is based on earlier an earlier repository by AIZoo Tech[3]. We thank AIZoo Tech for releasing their models to the public with the MIT License. Throughout this paper, we detail our original contributions on top of AIZoo Tech's technology to enhance public safety during a worldwide pandemic using computer vision and robotics.

II. Purpose & Motivation

A. Meet the Team

The team: Allen to the left, Hans to the right

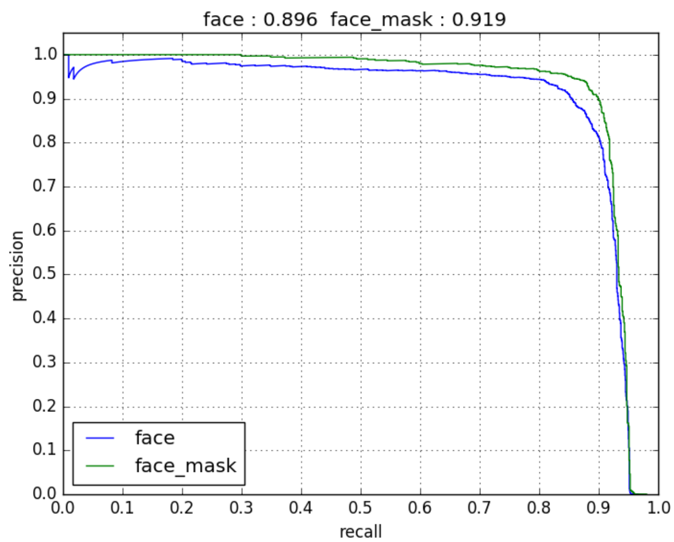
1) Allen: Allen is a Computer Science major at the University of California, Berkeley. His interests focus on Natural Language Processing and information extraction, as well as general software development.

[1]https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/diy-cloth-face-coverings.html
[2]https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public/when-and-how-to-use-masks
[3]https://github.com/AIZOOTech/FaceMaskDetection

2) Hans: Hans is

B. Previous Work on Mask Identification with Computer Vision

In the past several months, several organizations, for example Didi Chuxing[4] and AI Zoo Tech[5] have released open-source pre-trained neural network models on multiple platforms (Keras, Tensorflow, and Caffe) that classify faces within a photo or video as mask-wearing or not. Both models perform spectacularly as the Didi model was trained on a dataset of over 200,000 faces while the AIZoo model consists of 24 convolutional layers trained on approximately 8000 images from WIDER Face and MAFA datasets. Accordingly, the Didi model achieves at least 98% accuracy and the AIZoo model has a 0.896 ROC AUC (*receiver operating characteristic area under the curve*) for face detection and 0.919 ROC AUC for face mask detection, as seen in their curve below.



C. Our Solution

Maskit consists of two parts: a Raspberry Pi and a server. We connect a Pi Camera onto the Raspberry Pi such that the Raspberry Pi would be able to take periodic photos. The Raspberry Pi then sends these images to the server via a POST request. The server then runs the mask detection model on the incoming image and for each face that it identifies, classifies the face as with mask or without mask. These classifications are sent back to the Raspberry Pi as the response to the POST request. After the Raspberry Pi receives these classifications, if and only if all faces identified wear a mask, then it will activate the servo to open access. In our prototype, we model this situation by raising a welcome flag, though this can be generalized to opening store front doors, for example.

III. Step by Step Usage Instructions

A. Raspberry Pi Prerequisites

1) Materials: We used a Raspberry Pi Model B (700-MHz processor, 512 MB RAM). The physical limitations of this Raspberry Pi version are discussed later. In addition to the Raspberry Pi and its regular accessories, i.e. keyboard, power input, etc., we used:

1) HiTEC HS-5625MG servo [6]

[4]https://github.com/didi/maskdetection
[5]https://github.com/AIZOOTech/FaceMaskDetection
[6]https://hitecrcd.com/products/servos/sport-servos/digital-sport-servos/hs-5625mg/product

2) Pi Camera module [7]
3) 6 Volt DC input source. In our case, we put 4 AA batteries into a 9 volt battery case and used aligator clips to join the two empty battery terminals.
4) a breadboard (optional but makes setup nicer)

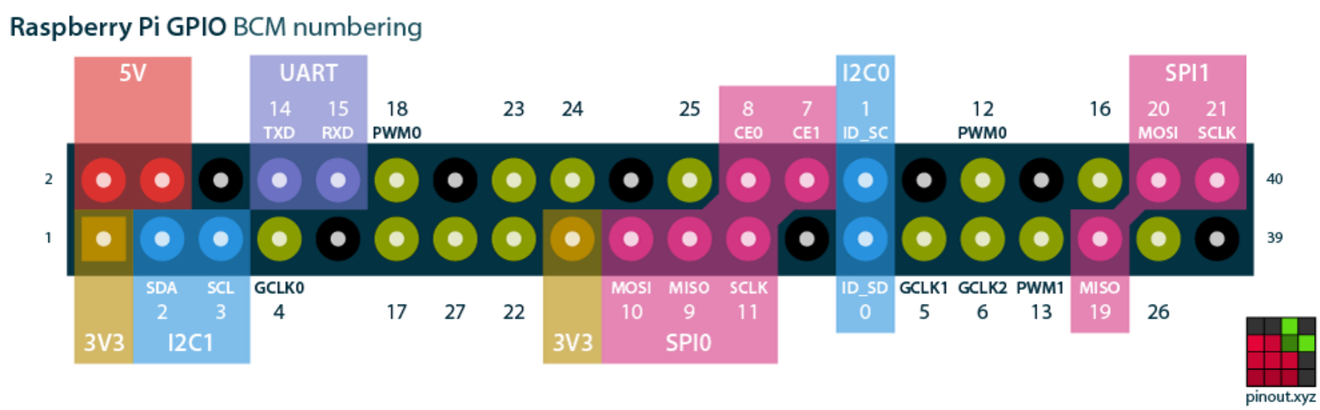
Our setup should work on the latest version of Raspberry Pi OS (formerly known as *Raspbian*). We will not go in depth on the Raspberry Pi settings on how to enable the Pi Camera or GPIO interfacing (i.e. `sudo raspi-config`, then enable both GPIO access and camera module), but links are provided in footnotes[8][9].

We installed the `picamera` and `wiringpi` with `python pip` for camera and GPIO pin interfacing, respectively, in Python.

In the next section, we describe how to connect the servo to the Raspberry Pi.

2) GPIO configuration: We use the Raspberry Pi's GPIO pins to control the servo. However, the servo cannot draw too much current from the Raspberry Pi without breaking it. As a result, the red servo wire must be connected directly to the 6 volt V_{in}. The black servo wire is connected to the ground. Make sure that the Raspberry Pi ground pin is also connected to this ground. The yellow wire is connected to PWM0 (*pulse-width modulation 0*), or physical pin 12[10].

See below for the Raspberry Pi pinout[11]:



B. Setting up the server

1) Install the dependencies found in the `environment.yml` file[12].
2) On terminal, run `conda activate maskdetect` to load the environment.
3) Run `python simple_server.py`. The server will be hosted on port 8000.
4) The server will now be listening for POST requests.
5) Run `hostname -I` to get host IP Address. We will need this value in the next step.

C. Running it on R Pi

To identify the server, the Raspberry Pi must know the server's IP address. In the source below, the IP address from aboved must be replaced into line 9 below (`upload_img_post.py`).

```
1 # upload_img_post.py
2 import os
```

[7]for setup instructions, see https://www.raspberrypi.org/documentation/usage/camera/
[8]https://www.raspberrypi.org/documentation/configuration/camera.md
[9]https://pimylifeup.com/raspberry-pi-gpio/
[10]See https://pinout.xyz/ for Raspberry Pi pin information
[11]Diagram by https://pinout.xyz/
[12]https://github.com/hansgundlach/FaceMaskDetectionRasPi/blob/master/environment.yml

```
3 import requests
4 import json
5 import wiringpi
6 from io import BytesIO
7 from time import sleep
8 from picamera import PiCamera
9 url = 'http://192.168.0.31:8000' # replace with actual IP address
10 wiringpi.wiringPiSetupGpio()
11 wiringpi.pinMode(18, wiringpi.GPIO.PWM_OUTPUT)
12 wiringpi.pwmSetMode(wiringpi.GPIO.PWM_MODE_MS)
13 wiringpi.pwmSetClock(192)
14 wiringpi.pwmSetRange(2000)
15 delay_period= .001
16
17
18 stream = BytesIO()
19 camera = PiCamera()
20 camera.start_preview()
21 sleep(2)
22 for i in list(range(2)):
23    camera.capture(stream, 'jpeg')
24    stream.seek(0)
25    data = stream.read()
26    r = requests.post(url,data=data)
27    lst_str = r.content.decode('utf-8')
28    lst = json.loads(lst_str)
29    all_masked = sum([x[0] for x in lst])
30    if (all_masked == 0):
31       for pulse in range(50, 250, 1):
32          wiringpi.pwmWrite(18, pulse)
33          sleep(delay_period)
34       sleep(3)
35       for pulse in range(250, 50, -1):
36          wiringpi.pwmWrite(18, pulse)
37          sleep(delay_period)
38
39    stream = BytesIO()
40    sleep(5)
```

After installing other prerequisites (e.g. the `requests` library[13]), run the client on the Raspberry Pi with the command `python upload_img_post.py`. As the program is currently, the Raspberry Pi takes two images with 5 seconds in between. To adjust this frequency, modify line 22. For perpetual monitoring, it is permissble to add an infinite loop, e.g. `while True`.

D. A Preview

We explain a frame from our demonstration video below. The monitor to the left is connected to the Raspberry Pi while the monitor to the right is connected to the desktop. We are displaying video for

[13]https://requests.readthedocs.io/en/master/

demonstration purposes; in widescale implementation, both the server and the Raspberry Pi could both be command-line only, i.e. without X-windows. As we see, the server has identified Allen's face as masked and the Raspberry Pi servo has responded accordingly by raising the flag.

Allen's mask recognized

IV. Difficulties and Challenges

A. Getting Models to Work

This project was an excellent learning experience for us to learn how to load trained models and use them for interesting applications. We were very fortunate to have found the models by AIZoo ([14]) as these models were not only more or less ready out of the box, but were available on through give of the mainstream deep learning frameworks: PyTorch, TensorFlow, Keras, MXNet, and Caffe. They were in fact not our original attempts as we had initally unsuccessfully tried to compile Didi's models [15]. After that failed, Allen tried to translate some of the C++ code into Python for a better understanding of what was going on, but the short timeline of Hackathons meant that we explored other options and we thus came upon AIZoo's contribution.

B. Hardware Limitations

As was mentioned previously, the Raspberry Pi we used is not the latest piece of technology as it contains merely a 700-MHz processor and 512 MB RAM. As such, setup on the Raspberry Pi was time-consuming. Initially, we had ambitions of facial recognition on the Raspberry Pi itself, but this became impossible as loading tensorflow itself would take almost an entire minute; this process would be more of a hindrance to the general public than any help. As such, we devised the server-client system that we have described thus far.

That said, we would be interested in continuing this project with a newer version of the Raspberry Pi, an improvement that would likely significantly boost current performance and perhaps even allow for mask recognition on the Raspberry Pi itself. We discuss these benefits more in depth later.

C. Remote Work

It goes without saying that work on such a hardware-oriented project is difficult when one teammate lives in Fremont, California and the other in Seattle, Washington. However, now that we have created a working prototype, we are even more proud to have overcome this additional hurdle.

[14] https://github.com/AIZOOTech/FaceMaskDetection
[15] https://github.com/didi/maskdetection

V. Market Evaluation

A. Cost Analysis

As the Raspberry Pi is famous for being a low cost computer, our calculations show that our Raspberry Pi setup can cost even less than $100, depending on servo quality, a very attractive price to business owners. Prices may be even lower when purchasing from sites such as eBay. Refer to the table below for the cost breakdown:

Item	Cost
Raspberry Pi	$35[16]
Pi Camera	$29.95[17]
Servo	$10-50[18]
Batteries and Wiring	$10
Total	$85-125

B. During a Pandemic, Mask Wearing Enforcement is Essential

The conception for Maskit came actually we watched news reports of customers refusing to wear masks and store employees' safety threatened when forced to deal with these customers [19] and we wondered how we could automatically prevent such unruly customers from entering businesses. Just like how sliding doors at many supermarkets only open when motion detectors are activated, we reasoned that it would be interesting to explore an add-on to this concept, one that integrated facial recognition to open only when the customer wears a mask.

VI. Suggested Improvements

A. Additional Features in Public Spaces

MaskIt needs to be fully evaluated in larger public situations to determine its utility in public spaces. More importantly, we need to implement overrides in case Maskit misidentifies customers with masks as not having a mask. Customers need a contactless way to dispute misidentification. We propose some sort of foot pedal that connects to the Raspberry Pi that the customer can use to dispute a classification. A store attendant on the server side can then make the decision and override Maskit.

B. Image Recognition and Processing on Raspberry Pi

We hope to invest in newer Raspberry Pis and explore whether it is possible to run the models within the Raspberry Pi alone with comparable efficiency. This form would significantly reduce costs as customers would no longer have to rely on servers.

C. Image Recognition and Processing on the Cloud

Or the hardware can be made much more streamlined and small and all the image processing can be done on a cloud server instead of a our homemade desktop server. Mask detection can also be done along with other measures to acquire data on COVID-19 symptoms, including temperature detection and heart rate detection. This measurements are effective with just video sources[20].

[19]https://www.youtube.com/watch?v=O8fkVqEZGRA
[20]https://web.stanford.edu/class/cs231a/prev_projects_2016/finalReport.pdf

VII. Conclusion

In conclusion, MaskIT has significant potential in research and commercial settings during the COVID pandemic. Given the lack of curret data on mask usage in practice and the importance of mask in the current public health it is important to create automated systems to tack account of mask usage. Some building may need automated gate systems to scan mask usage while universities will need to monitor the percentage of mask compliance on campuses. Systems like MaskIT will be of increasing public importance in the coming months.

References

[1] S. Fink, "Worst-case estimates for us coronavirus deaths," *New York Times*, 2020.

[2] N. Fernandes, "Economic effects of coronavirus outbreak (covid-19) on the world economy," *Available at SSRN 3557504*, 2020.

[3] R. M. Viner, S. J. Russell, H. Croker, J. Packer, J. Ward, C. Stansfield, O. Mytton, C. Bonell, and R. Booy, "School closure and management practices during coronavirus outbreaks including covid-19: a rapid systematic review," *The Lancet Child & Adolescent Health*, 2020.

[4] J. Brandon, "A doctor explains why 45protective mask," May 2020.

[5] N. Vigdor, "American airlines bans conservative activist who refused to wear a mask," Jun 2020.