Face Mask Recognition using AI for Computer Vision

Junghwan Kim
Master's Student
Department of Computer Science
University of Massachusetts

Mohammad Arif Ul Alam, Ph.D.
Assistant Professor
Department of Computer Science
University of Massachusetts

***Abstract*— COVID-19 is a severe acute respiratory syndrome disease. Unfortunately, it is highly contagious, and the global spread is accelerating. To prevent such high infectivity, the only way to prevent the virus from spreading is wearing a mask. In these days, wearing a mask has become a necessity, not an option. However, someone ignore quarantine rules and do not wear masks or cover their nose. We need to monitor whether to wear a mask. This application allows people to easily, quickly and accurately find someone who does not wear a mask or cover nose using a smartphone everyone has.**

Keywords*— *deep learning, artificial intelligence, machine learning, object detection, tersorflow lite, computer vision, opencv, keras

I. Introduction

COVID-19 is a severe acute respiratory syndrome disease transmitted by the SARS-CoV-2 virus, a new type of variant coronavirus. The SARS-CoV-2 virus is a mutated virus of the corona virus and has a similar form to SARS and MERS. However, it is highly contagious, and the global spread is accelerating, and the fatality rate is high. It is a pandemic infectious disease that has been first reported and spread in China since November 2019, and continues to this day in the world, and is a contagious disease that infects both humans and animals. By current standards, it is very easily transmitted like a cold, but it can be said to be an unprecedented infectious disease with a high mortality rate against the elderly. Unlike SARS and MERS, as this situation continues for a long time, it is not an exaggeration that experts call it the second Black Death, the Spanish flu, and it is expected to be recorded as one of the worst infectious diseases that engulfed the entire global village in modern human history.

COVID-19 spreads from person to person primarily through respiratory droplets. Respiratory droplets move through the air when coughing, sneezing, talking or yelling, or singing. These droplets can fall into the mouth or nose of people around you or get into your body when you breathe. To prevent such high infectivity, the only way to prevent the virus from spreading is wearing a mask. A mask is a simple screen that prevents respiratory droplets from reaching others. Research shows that wearing a mask to cover your nose and mouth reduces the number of droplets released. Even if you are not sick, you should wear a mask. This is because studies have shown that even people infected with COVID-19 who have no symptoms (asymptomatic) and those who have no symptoms yet (before onset of symptoms) can transmit the virus to others. If you are infected but have no symptoms, wearing a mask can help protect those around you. Because the spread of COVID-19 occurs primarily between people who are in close contact with each other, wearing a mask is especially important when there are people around you who do not live indoors, and when you cannot distance them more than 6 feet.

Wearing a mask in indoor makes a tremendous contribution to preventing the spread of the virus. Unfortunately, however, some people ignore the quarantine rules and come into the room without wearing a mask. In addition, some people often expose the nose or mouth with a mask that does not cover both the nose and mouth. To catch these people, you need to have a watchdog at the entrance to the building to work. However, direct human monitoring is not effective. There are cases where the watchdog misses a person who is not wearing a mask, and it is difficult to properly monitor when a large number of people gather at once. In addition, since the watchdog has to work in shifts, a lot of labor costs are required.

In this paper, we used a smartphone that everyone possesses to solve this problem. With this smartphone, anyone can quickly and accurately detect whether a mask is worn anytime, anywhere. This application consists of two steps. In the first step, it checks whether a human face is detected, and in the second step, if it is a human face, it checks whether a mask is worn or not. Based on these two steps, the person who wears the mask and the person who does not are finally distinguished and displayed on the smartphone screen in real time. In this application, even a person wearing a mask can be recognized as a human face. In addition, even if several people appear at the same time, this application is processed for each person, and the results are also displayed for each person.

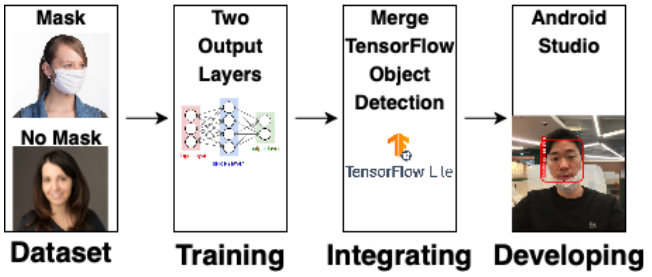

Figure 1: Overall System Architectural Diagram

II. Overall System Architecture

Figure 1 illustrates a schematic diagram of our developed system architecture which consists of four components:

1. Dataset

First of all, we need dataset to train model. We used dataset from GitHub repository. It was created by PyImageSearch reader Prajna Bhandary on GitHub here:

https://github.com/prajnasb/observations/tree/master/experiements/data

This dataset consists of 1,376 images belonging to the following two classes: with_mask and without_mask. with_mask is consisting of 690 images and without_mask is consisting of 686 images. Prajna Bhandary used scrapped web data from Google Image Search with classes that people wearing mask and no mask and trained it. And then she used the model to predict on a webcam feed. My goal is to train a customized deep learning model that detects whether a person is wearing a mask or not.

To create this dataset, Prajna provided the following ingenious solution. First, taking normal images of the face then create a custom computer vision Python script to add face masks to create an artificial (but still practical) dataset. This method is actually much easier than it sounds once you apply a face cover to your problem.

To build a data set of a face wearing a face mask using facial landmarks, we first need to start with an image of a person without a face mask. There we apply face detection to compute the position of the bounding box of the face in the image.

The next step is to apply face detection. Here, we used a deep learning method to perform face detection with OpenCV. Once we know the location of the face, we can extract the region of interest (ROI) of the face. The next step is to extract the facial ROI using OpenCV and NumPy slicing. And from there, we applied facial landmarks to position the eyes, nose, mouth, etc. Then, we use slip to detect the landmarks on the face to know where to put the mask on the face.

This set is consisting of with_mask and without_mask" dataset for Computer Vision technology using Python, OpenCV and TensorFlow based on Keras.

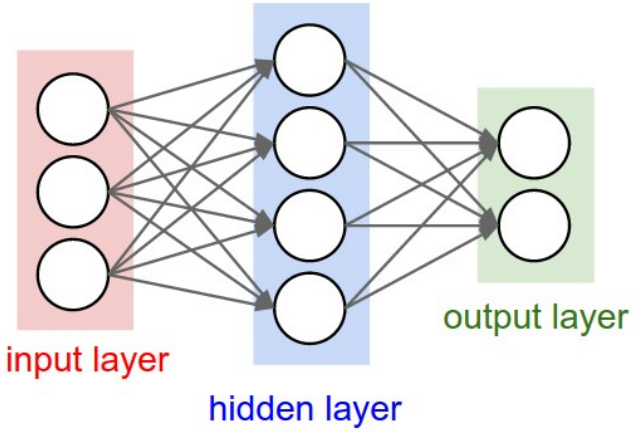

Figure 2: Model Layers

2. Training

Now that we have reviewed our dataset, we want to use Keras and TensorFlow to train a classifier that automatically detects whether a person is actually wearing a mask. It is divided into two classes and needs to be determined, so the classification method should be used. We will use skit-learn(sklearn) for normalization of two class labels, data set segmentation, and printing classification reports. Using sklearn, we divided the typical dataset classification ratio: 80% of training and the remaining 20% of testing.

It uses the MobileNet V2 architecture, which is a very efficient and fast architecture for fast detection to mobile devices. We load the MobileNet V2 using pretrained ImageNet weights.

With our data and model architecture ready to be fine-tuned, we can now compile and train our face mask detection network. Adam Optimizer was used for the compilation method, and our model was compiled using categorical cross entropy. If you build this training script with 2 classes, we must use categorical cross entropy.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 180, 180, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 180, 180, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 90, 90, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 90, 90, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3965056
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 3,988,898
Trainable params: 3,988,898
Non-trainable params: 0
```

Figure 3: Model Summary

As shown Figure 3, we trained 15 epochs with 100 iterations using 11 batch size. When training is complete, the resulting model is evaluated on the test set. And then it plots the accuracy and loss curves graph.

3. Integrating

In order to use the model, we have already created, we first need to integrate it with TensorFlow Examples and Google ML Kit. Among several examples, we used the TensorFlow Lite Object Detection Android Demo.

https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android

The original code works based on a single model trained on the COCO data set and computes the result only once. For this app, we need to implement two-step detection. The first step is to detect if it is a person's face, and the

second step is to check if the mask is detected if the person's face is correct. In the first step, face detection, we will use Face Detection from Google ML Kit.

https://developers.google.com/ml-kit/vision/face-detection/android

We opened the TensorFlow Object Detection Android Example project into Android Studio and added the ML Kit dependency. Once the project is load it, we can use Face Detector to perform detector activities. Face detectors are created with options that prioritize performance over other features. The original app defines two bitmaps. We will define two additional bitmaps (vertical Bmp and face Bmp) for processing. The first is to rotate the input frame in portrait mode for devices with the sensor positioned horizontally. And the face Bmp bitmap draws all detected faces and cuts the detected position. When cropped and turned off, it is resized to 255 x 255 pixels and used as input to the MobileNet V2 model. ToCropTransform transforms the coordinates from the original bitmap to the clipped bitmap space, and CropToFrameTransform transforms it in the opposite direction.

When a human face is detected, the original frame is drawn on a vertical Bmp bitmap to continue the second step, mask detection. For each face detected, the bounding box is searched and mapped from the cropped space to the original space. Face cropping is done by converting the vertical bitmap to the origin of the face and resizing the face bounding box size to match 255 x 255 pixels. Finally, the mask detector is called.

4. Developing

Before using TensorFlow model, we must convert it into TensorFlow Lite model for using at Android device. There is converter using Python. It is the easiest way to convert. Once we integrated TensorFlow Object Detection Android Application and Google ML Kit Face Detection, we develop mask detection part.

In addition, in the configure of detector activity section, we need to adjust the necessary parameters to suit our model requirements. We set the input size of the model for fitting to TensorFlow Lite. We need to point to the mask detector file. We can also create label map text files in assets folder with class names with_mask and without_mask.

Then we can modify the recognize method of the TensorFlow Lite model class. The output labels are hard-coded, but we can be easily retrieved from the label mapping text file. Finally, you can slightly modify the recognition class and multi-box tracker class by adding color to the recognition result. We put visual information into the model we trained and receive the result. If the result is lower than the threshold, I set it to green for "Mask", red for "NO MASK", and blue for "Unknown".

III. Experimental Evaluation

To evaluate this project, we need to evaluate trained model first.

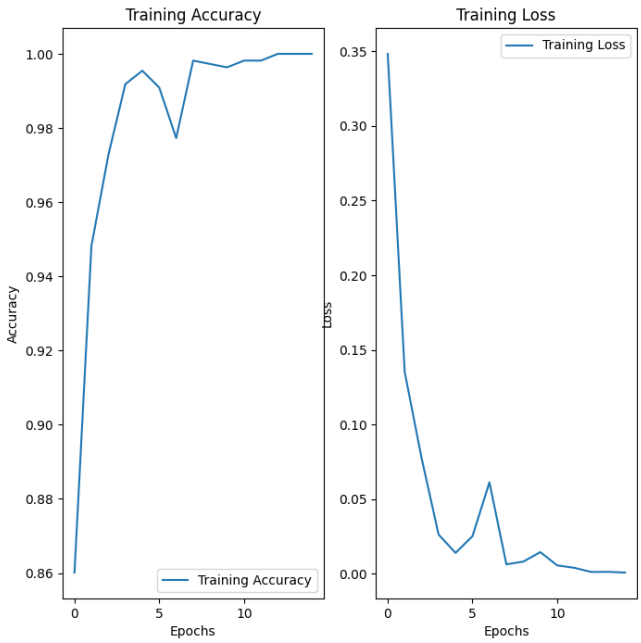

Figure 4: Training Accuracy and Loss

Looking at Figure 4, we can see that the validation loss is lower than the training loss and there are few signs of overfitting.

Result	**Value**	**Unit**
Model Type	sequential	N/A
Training Loss	0.0013	0.0 = 0% 1.0 = 100%
Training Accuracy	1.0000	
Validation Loss	0.3112	
Validation Accuracy	0.9564	

We used the following techniques to improve this model a bit better.

1. Batch Normalization
2. Dropout (Training Rate: 0.7, Evaluation Rate: 1.0)
3. Xavier Initialization (For weight initialization)
4. Leaky ReLU (LReLU) instead of Sigmoid (Prevent the vanishing gradient problem)
5. Adam Optimizer instead of Gradient Descent Optimizer (The Gradient Descent Optimizer uses fixed learning rate for whole network weight)
6. Data Augmentation

Given these results, we hope that our model creates well to images outside the training and test set. As results, this model should be works great to classify wearing a mask or not.

The next step is our trained model can distinguish between wearing a mask person and not wearing a mask person.

Figure 5: Test Pictures

As shown Figure 5, to test our trained model, we prepared two pictures: /Python/test/1.jpg (Mask) and /Python/test/2.jpg (No Mask). We implemented test program using Python.

```
This image most likely belongs to with_mask with a 99.99 percent confidence.

Process finished with exit code 0
```

Figure 6: Test Mask Picture

```
This image most likely belongs to without_mask with a 100.00 percent confidence.

Process finished with exit code 0
```

Figure 7: Test No Mask Picture

The Figure 6 is result of test program when the test picture is 1.jpg (Mask). The test program tells it is 99.99% confidence of with_mask. The Figure 7 is result of test program when the test picture is 2.jpg (No Mask). The test program tells it is 100% confidence of without_mask. As a result, it works accurately.

The final step is our Android application can really detect whether to wear a mask.

As mentioned above, there are three results which are displayed on out Android application.

Result	Condition
Mask	with_mask > without_mask
NO MASK	with_mask <= without_mask
Unknown	Detect human face, but cannot recognize a mask

"Unknown" result can be occurred under the following conditions.

1. When the background around the face is like the skin color
2. When exposed to direct sunlight
3. When there is a shadow and only the shape of the face border can be recognized
4. When backlit in picture

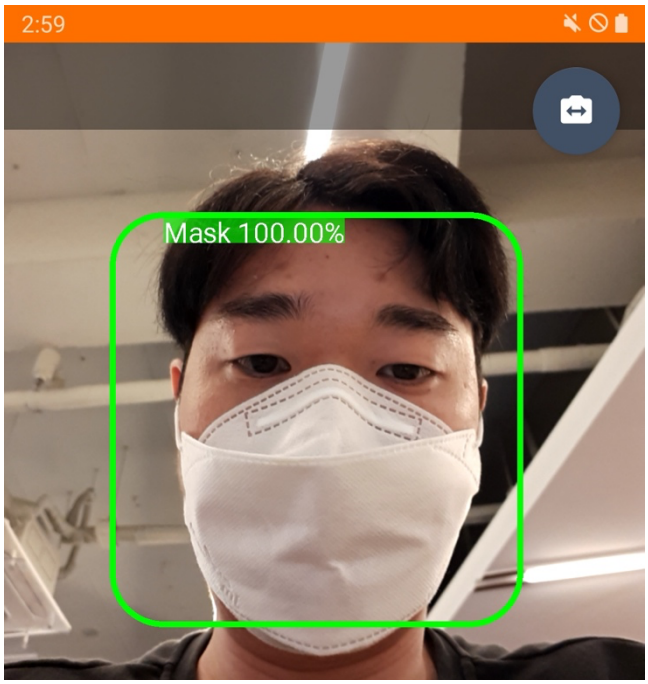

Figure 8: Fully Wearing Mask Person

As shown Figure 8, person is wearing a mask fully. The mask covers his nose and his jaw. Our application correctly labeled this image as "Mask". We tried another; this is a person not wearing a face mask.

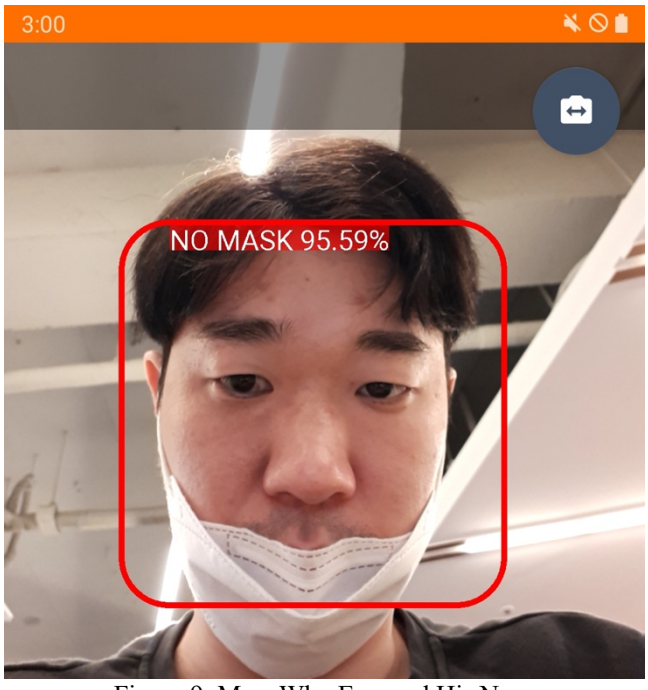

Figure 9: Man, Who Exposed His Nose

As shown Figure 9, person is wearing a mask partially. The mask does not cover his nose. It only covers his jaw. Our application has correctly predicted "NO MASK".

In most common situations, this app works fine. However, under some special circumstances, this app has limitations in operating abnormally.

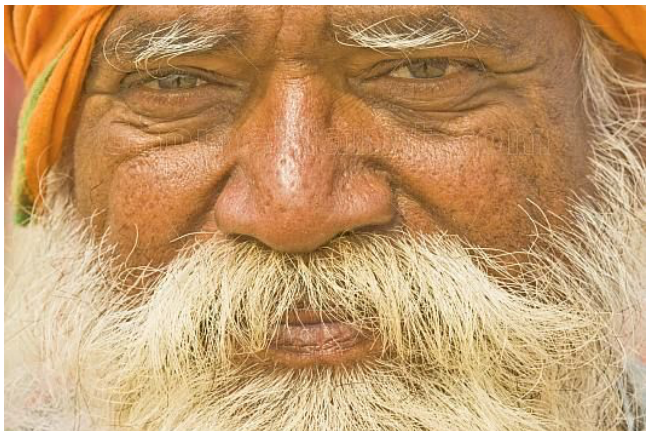

Figure 10: Beard Person

First, beard guys are recognized as masked as shown Figure 10. He exposed his mouth partially, however our model cannot detect correctly. It will be possible to retrain with additional beard guys to find them as “No Mask”.

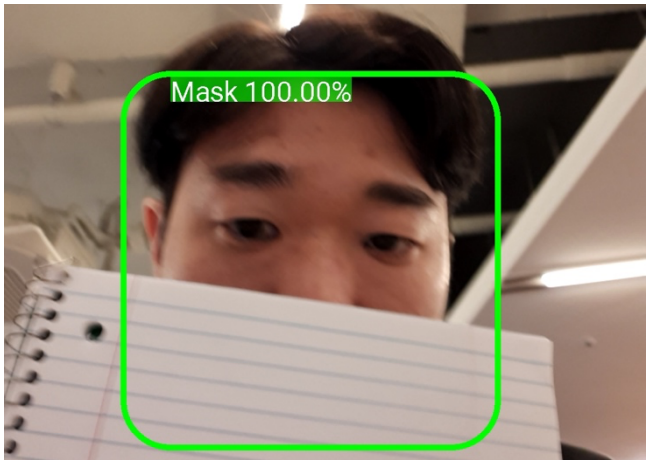

Figure 11: Man, Who Covered His Face Using Paper

Second, people cover his face using paper without mask, our application is also recognized as masked as shown Figure 11. Even he does not wear a mask. Because our model always finds person’s nose and mouth. We need to improve more details to recognize covered material is face mask clearly.

IV. Conclusion

The recent coronavirus outbreak has brought people around the world to new challenges. In the context of this uncertainty, this application that detects a face mask on a smartphone can play our part by contributing to the fight against this disease. Computer vision and deep learning methods using Python, OpenCV, and TensorFlow and Keras have enabled automatic detection of the presence of masks. It is a great opportunity to serve mankind by utilizing these technologies.

Solutions that are within the reach of everyone can help control the use of face masks. And today, everyone has a smartphone, so perhaps a mobile application will help. It can be used immediately by installing it at the entrance of a restaurant or store, or at the entrance of any building.

V. References

[1] Prajna Bhandary. (2020, April). Mask Classifier Dataset: (https://github.com/prajnasb/observations)

[2] Google. (2021, Feburary) TensorFlow Lite Object Detection Android Demo: (https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android)

[3] Google. (2021, April) Detect faces with ML Kit on Android: (https://developers.google.com/ml-kit/vision/face-detection/android)

[4] Das, Arjya, Mohammad Wasif Ansari, and Rohini Basak. "Covid-19 Face Mask Detection Using Tensorflow, Keras and OpenCV." 2020 IEEE 17th India Council International Conference (INDICON) (2020). Print.

[5] "COVID19 Face Mask Detection Using Deep Learning." COVID19 Face Mask Detection Using Deep Learning - MATLAB Central. 2021. (https://kr.mathworks.com/matlabcentral/fileexchange/76758-covid19-face-mask-detection-using-deep-learning)

[6] Gupta, Ayush. “A Comprehensive Guide on Deep Learning Optimizers.” Analytics Vidhya, 2021. (https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/)