EECS 442 Project: MasKlassifier

Shiqi Sheng
University of Michigan
ssheng@umich.edu

Ashutosh Bhown
University of Michigan
asbhown@umich.edu

Adityasai Koneru
University of Michigan
adityask@umich.edu

Milan Gupta
University of Michigan
milagu@umich.edu

Vishnu Murthy
University of Michigan
vrmurthy@umich.edu

Abstract

This project is a deep learning classifier that can determine if a person is wearing a face mask correctly or not when shown an image of said person. This classifier contains three sub-classifiers, one for detection of a visible nose, one for a detection of a mouth, and one for detection of a mask. These three outputs are combined to determine if a mask is worn correctly, covering the mouth and nose.

1. Introduction

The COVID-19 pandemic took the world by surprise in the first months of 2020, forcing the entire world to adapt to an isolative way of life to avoid the spread of a new, highly contagious, and lethal coronavirus that transmits through viral particles that are absorbed through the eyes and nose. An invaluable part of curtailing transmission of the COVID-19 virus is wearing a mask over the nose and mouth. This keeps potentially viral particles restrained, minimizing infection risk in both private and public through interactions between people who are not in each others' households.

1.1. The Problem

Although most established public health organizations both nationally and internationally (including the WHO and CDC) highly recommend wearing masks in public, research done in August 2020 by Dr. Deborah Cohen *et al*. as shown in [2] found that 49% of people Philadelphia in commercial corridors were wearing their masks incorrectly, if at all.

This poses a huge problem as the pandemic rages on to this day, over a year after it began. Unless the entire population contributes to public health and safety by following current guidelines, everyone is put at a significantly higher risk of contracting COVID-19 and spreading it further. Thus, most businesses and establishment chains that host their patrons in indoor spaces require masks at all times within their buildings. However, it is all too easy for customers to wear their masks incorrectly by leaving their nose, mouth, or both exposed, or even remove their masks completely.

1.2. The Solution

The only current line of defense to prevent these consumers from putting an entire building's occupants at risk is for employees to notice these rule-breakers and initiate the appropriate resolution process. However, this is not a guaranteed nor efficient solution, as employees have other duties to attend to, and it is possible to avoid employees to begin with. This system evidently leaves much to be desired; if there was some sort of automated method of detecting whether someone was not wearing their mask correctly, and employees could be alerted to deal with these situations promptly, it would promote a significantly safer environment in businesses for everyone.

Thankfully, deep learning technology has permitted developers like Adrian Rosebrock to be able to do just this [4]. He developed a program to recognize masks in real-time video feeds. APIs such as PyTorch, OpenCV, and TensorFlow allow streamlined construction of deep learning networks that can utilize image manipulation and convolution to learn features through spatial patterns and then learn to predict classes when presented with unlabeled examples. Many of these current models are binary in detecting either mask or no mask. Thus, these models fail to account for incorrectly worn masks.

Using PyTorch, we created three convolutional neural networks to determine if a person is wearing their mask correctly. The three networks detect noses, mouths, and masks. If a mask is detected and both nose and mouth are not visible, the system will label the face as correctly masked. If no mask is detected, it will be labelled as maskless. Otherwise, it will be flagged as incorrectly masked.

2. Approach

The deep learning classifier we created takes in an image of a face and determines if that person is wearing their mask correctly. This was implemented in Python, largely using the PyTorch Application Programming Interface. It consists of four components: A neural network used to determine if a nose is visible in an image (this will be referred to as the Nose Classifier), a neural network used to determine if a mouth is visible in an image (this will be referred to as the Mouth Classifier), a neural network to determine if a mask is visible (this will be referred to as the Mask Classifier), and a driver program to consolidate the outputs of all of these individual feature classifiers into a determination of whether the subject is correctly wearing a mask, covering their nose and mouth with their mask, or incorrectly wearing their mask, or not wearing a mask at all.

2.1. Dataset

The dataset selected for this project is called MaskedFace-Net and consists of 133,783 images of both Correctly Masked and Incorrectly Masked faces [1]. A correctly masked image is defined as a mask covering both the nose and mouth of the face. Incorrectly worn masks fail to cover either one or multiple of these facial features. However, this dataset fails to include faces with no mask worn.

The MaskedFace-Net dataset is based on the dataset Flickr-Faces-HQ (FFHQ) which contains 70,000 images of human faces [3]. MaskedFace-Net features these faces with realistic computer generated masks placed on the faces. In order to access images without any masks, we defaulted to the original FFHQ dataset.

Images in both datasets are 1024 by 1024 pixels.

2.2. Data Preprocessing

Before even considering the procedure that turns the input of an image of a face into classification of whether they are correctly masked or not, the very first thing to consider is augmentation of the actual dataset itself to potentially improve efficiency and performance, which in this case consisted of color images of resolution 1024 pixels by 1024 pixels, with each image containing a face with either a correctly worn mask, an incorrectly worn mask, or no mask at all. Two augmentations were made to the images from this dataset used to train and test the system: resizing and greyscaling.

In addition to resizing and greyscaling, we created an additional dataset called the "Translations" dataset.

2.2.1 Resizing

The images used in this dataset were a default size of 1024 pixels by 1024 pixels, which provides a very high resolution. Although this may provide more image clarity, it does not translate well into being processed by a neural network, as it takes significantly more time to process more complex images. As the classifier depends on three neural networks for final output, using these high definition images would be extremely slow. To speed this process up, each image used was resized to 256 pixels by 256 pixels. This makes images significantly easier to process for the neural networks and also works to reduce some noise in the image.

2.2.2 Greyscaling

This dataset was created only using one form of mask – blue surgical masks. Although these are common, there are many other types of masks in use today, and it would thus be a possible danger to train on these types of masks, as recognition of the distinctive blue color of these masks may create a shortcut for the Mask Classifier to prematurely classify a positive instance of a mask and/or restrict potential instances of masks to have the color blue. This would not be able to translate well to the real world, as application to another type of mask (i.e. a black cloth mask) could potentially result in a false negative from the Mask Classifier.

To work against this, we chose to greyscale images that would be used for training and testing the classifier, such that the possibility of the networks (the Mask Classifier in particular) learning shortcuts through the colors of the masks or anything unnecessary in the image would be reduced. This reduces the image from three channels (R, G, B) to one channel (greyscale).

2.2.3 Augmented Dataset with Translations

Our dataset primarily consists of images with faces located in the center of the image. Although this is ideal in terms of consistency, this may not reflect real-world scenarios as heads can be shifted and tilted. We want to make sure our algorithm is more generalizable and robust. Thus, we created a separate dataset to reflect this. For each greyscaled/resized image, we performed 4 shifts (up/left, up/right, down/left, down/right each by 15% of the height and width of the image), 20° clockwise rotation, and 20° counterclockwise rotation. Each image would produce a series of 6 additional images, one for each of the aforementioned translations. Due to computational limits, 2 resultant images out of the 7 candidate images (6 translations and the original) were randomly chosen to be in the dataset. Augmenting our dataset with translated images allows for a model with greater semblance to real-world scenarios.

2.3. Network Architectures and Specifications

2.3.1 Classifier Architecture

The Nose Classifier, Mouth Classifier, and Mask Classifier were all implemented using PyTorch. The same architecture was used across all three neural networks to maintain consistency and make debugging easier. The architecture is as follows:

1. Max CNN Block I:
   (a) Convolutional 2D Layer (input size, hidden layer size, 3x3 kernel size)
   (b) Batch Norm 2D Layer (hidden layer size)
   (c) ReLU Activation layer
   (d) Max Pool 2D Layer (2x2 kernel size, 2x2 stride)
2. Avg CNN Block:
   (a) Convolutional 2D Layer(input size, hidden layer size, 3x3 kernel size)
   (b) Batch Norm 2D Layer (hidden layer size)
   (c) ReLU Activation layer
   (d) Average Pool 2D Layer (2x2 kernel size, 2x2 stride)
3. Max CNN Block II:
   (a) Convolutional 2D Layer(input size, hidden layer size, 3x3 kernel size)
   (b) Batch Norm 2D Layer (hidden layer size)
   (c) ReLU Activation layer
   (d) Max Pool 2D Layer (2x2 kernel size, 2x2 stride)
4. Fully Connected Layer (flattened hidden layer output size, output size)

2.3.2 Classifier Hyperparameters

Each neural network also contains the following hyperparameters:
Input size = 1
Hidden Layer Size = 32
Output Size = 2
Batch Size = 64
Learning Rate = 0.01
Weight Decay = 0.0001
Number of Training Epochs = 10

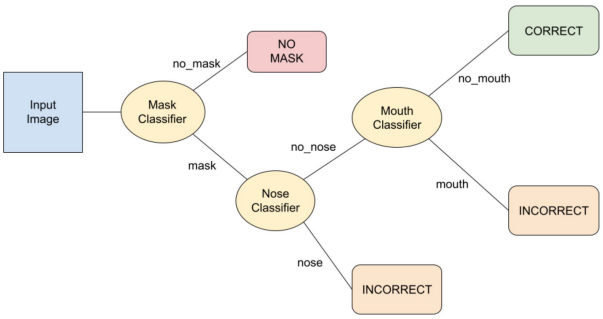
2.4. Final Output Consolidation Algorithm

Once all three neural network sub-classifiers have their outputs, the final step is to aggregate the information from all three networks and determine if the person in the original input image is wearing their mask correctly, incorrectly, or not wearing a mask at all. This is done in essentially several conditional statements:
If a visible nose is not detected, a visible mouth is not detected, and a mask is detected, the image is classified as "correctly masked".
If no mask is detected at all, the other two inputs are ignored and the image is classified as "no mask".
In any other combination of outcomes (mask/nose visible/mouth not visible, mask/nose not visible/mouth visible, mask/nose visible/mouth visible), the classifier labels the image as "incorrectly masked".

2.5. Visualizations of System

Illustration of Image Preprocessing:

Illustration of Classifier Pipeline:

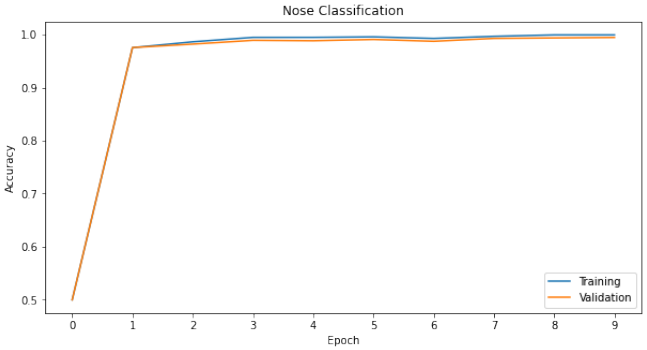


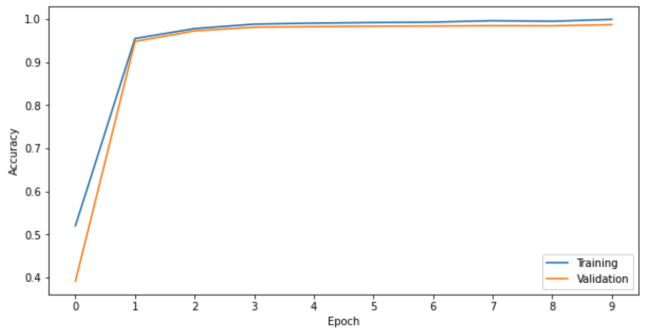
3. Experiments

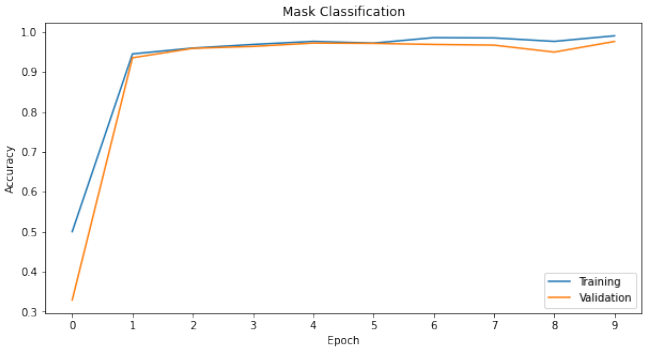
3.1. Training Process

As stated previously, our model is a compilation of three classifiers: Nose Classifier, Mouth Classifier, and Mask Classifier. In our training process, we decided to train each of these classifiers separately before combining the trained models. Using the preprocessed MaskedFace-Net images, we trained all three classifiers for 10 epochs each. We also class balanced our data for each of the three models by training on an equal number of images per class.

Below you can see the training and validation accuracy plots for each of the classifiers:







The final testing accuracies for Nose Classifier, Mouth Classifier, and Mask Classifier are **99.45%**, **98.67%**, and **97.6%**, respectively. All three of these classifiers perform significantly better than the accuracy of a random classifier (50%).

3.2. Combined Model

The final model uses the outputs of all three intermediary classifiers to make a ternary decision. If a mask is not detected by the Mask Classifier, the model will identify the individual in the image as not wearing a mask. If a mask is detected but either the Nose Classifer or Mouth Classifier detect their respective facial feature, the model will output a classification of an incorrectly worn mask. The model only classifies a correctly work mask if a mask is detected without detecting a nose or mouth.

In all, our designed model makes a ternary decision: no mask, incorrectly worn mask, and correctly worn mask. Here is an example classification on an incorrectly masked individual:

```
Nose Classification: nose
Mouth Classification: no_mouth
Mask Classification: mask
Classification: incorrect
```

Based on our final tests on the MaskedFace-Net dataset, our model has an accuracy of **79.15%**.

Since most of the faces in those images were centered, we used our translation dataset to test the robustness of our model. When testing on this augmented dataset, our model had an accuracy score of **58.13%** which is still significantly better than a random ternary choice (33%).

4. Implementation

In terms of our classifiers, our primary goal for this portion was to design a convolutional neural network architecture which would be effective. In order to do so, we used various built-in functionality from PyTorch and Numpy.

We started by trying to create our own driver for the classifiers but ran into several difficulties due to lack of experience. Since we primarily wanted to focus on the model architecture, we started by modifying the driver code from Homework 5 of EECS 442 to maximize our time. Once we had designed three viable classifiers from scratch, we designed our own script to combine these three classifiers into one cohesive model (as seen in classifier.py).

References

[1] cabani. Maskedface-net dataset. 2020. 2

[2] B. H. D. Cohen, T. MacKenzie and S. Williamson. Studio ludo somad report, 2020. 1

[3] N. R. Labs. Flickr-faces-hq dataset. 2019. 2

[4] A. Rosebrock. Covid-19: Face mask detector with opencv, keras/tensorflow, and deep learning, 2020. 1