Face Mask Detection

Rohith Reddy Bollareddy, Brijesh Rao Pamara, Blessy Kuriakose, Amrutha Veeramachaneni

Introduction

The motive of this project is to encourage mask usage by bringing attention to whether a mask is worn correctly on one's face or not. (This is under the same principle as the use of speed monitoring displays to encourage drivers to reduce their speed in select locations.) The goal is to report this information to those to whom this is a concern, which, for the present purposes of this project, will be the individual themselves. The significance is in the present condition with the pandemic: Mask usage is an effective step in protecting ourselves and each other from the spread of the virus. The objective is to have more people properly use masks in the setting of implementation.

The completed project is in a model that can determine (classify) if a mask is being properly worn by an individual in an image. There is also a separate code intended to home in on a person's face from a live camera feed. These two parts have not been integrated at this time, however further work may look at doing this integration so that the model can be used in real settings.

Background

There are multiple implementations of face mask detection that are referenced in the literature. One literature is the "Review on Literature Survey of Human Recognition with Face Mask." This one turns its attention to recognizing faces while a face mask is present, however many of its methods for face and mask detection remain relevant to our project. It refers to multiple methods of evaluation/feature extraction such as sift and Hog, as well as some novel variants. It also contains references to other texts that take to a simple mask detection model, however many of these models do not take a feature-extraction approach but use (black-box) descriptors for classification.

From research we have found some possible sources of error or misfunction. These include different types of masks with different dimensions and shapes which may overlap with our region of interest; occlusions like eyeglasses, hats, and headphones which increase noise in the images; other external obstacles that can cover the face; orientation of the face in the image being analyzed; and manipulated or otherwise incompatible images in the training and testing dataset. While not all of these can be eliminated as a factor, especially in real-life implementation, we aim to consider the effects of these in the outputs of the classifier.

Model Architecture and Workflow

In **Figure 1** (below) shows a diagram of the architecture of the project:

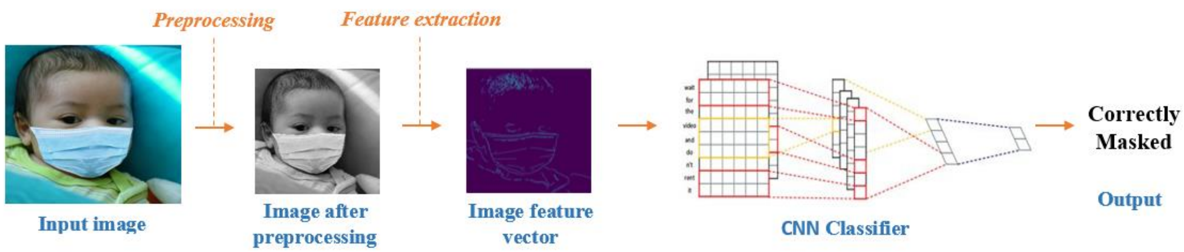

Figure 1: Architecture Diagram. The input image is preprocessed and then the features are extracted to get the feature vector. This is then passed on to the CNN Classifier to determine whether the person in the image is correctly masked or not.

As the diagram shows, the image is first preprocessed for uniformity and to keep the content relevant for the feature extraction that follows. (More on preprocessing under ***Analysis of Dataset*** section). From there three edge features are extracted. (More on feature extraction under) these are what gets inputted to the CNN Classifier to attain an output. (The particular CNN we used is ResNet-18.)

In **Figure 2** (below) shows a diagram of the workflow of the project:

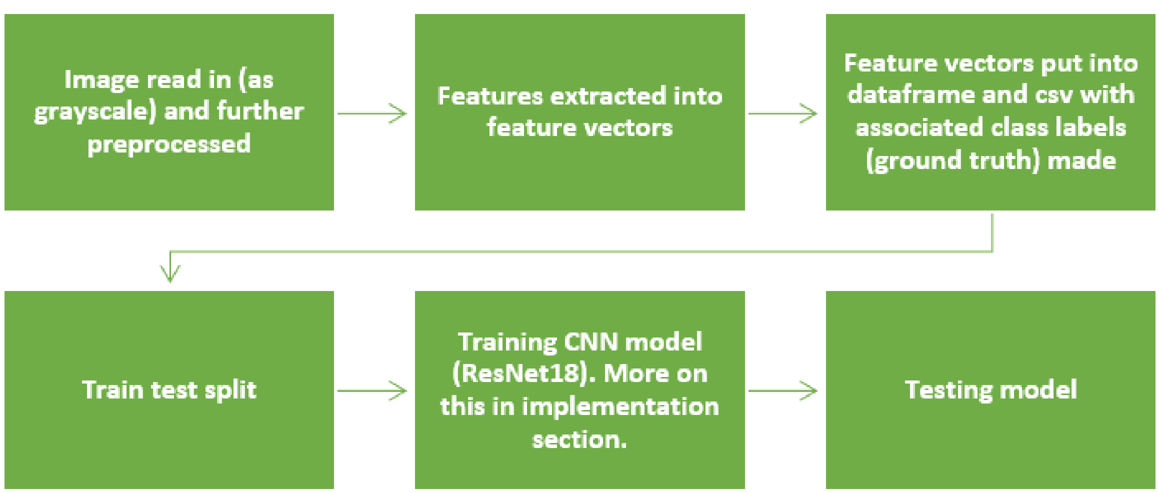

Figure 2: Workflow Diagram. This diagram outlines the basic process in creating and testing the model. More details on the training can be found in the ***Implementation*** section. Details on the images used can be found in the following section, titled ***Dataset***.

Dataset

The data used to train the model is the features extracted from a database of images called MaskedFace-Net; Here is a link to the repository: https://github.com/cabani/MaskedFace-Net. This database contains a total of 133,783 computer altered images of faces that are classified as being correctly masked or incorrectly masked in their different variations: chin uncovered, nose uncovered, chin and nose uncovered. In this project, we only focused on binary classification, and we only needed to use a portion of the available images: a total of 33,721 images of correctly masked faces and 33,588 images of incorrectly masked faces. More details on where the original images came from (before alteration) are given within the repository.

Samples of the images are shown below in **Figures 3** and **4**:

Figure 3: Sample images. The left image shows a woman with the mask on correctly. The right image shows an image with the mask on incorrectly (nose is uncovered).

Figure 4: More sample images. The leftmost image shows a girl with a mask on incorrectly (mouth and nose uncovered). The middle image shows a man with a mask on incorrectly (chin uncovered). The rightmost image shows a man with the mask on correctly.

These images are first preprocessed, as outlined in the following section, and then a total of three features are extracted: *Sobel edge detection*, *Canny edge detection*, and *Laplacian of Gaussian* (a blob detector). This information is what is then used for training and testing.

Analysis of Dataset

After examining the dataset, the preprocessing stage is next. This happened in three stages, each of which are shown in **Figure 5**, below. The steps taken include turning the image into a greyscale, resizing them to a reduced size, and denoising them via an inbuilt function.

Figure 5: Image preprocessing. Shown is a sample of how an image is read in as a grayscale and then resized to a smaller size (224x224 pixels). From there it is denoised to get rid of excess details.

After preprocessing the three edge features mentioned previously are retrieved from the image and put into a data frame for use with the model. **Figure 6**, below shows how each of these features look in image form.

Figure 6: Features extracted. These show the features extracted from a sample image *(same as used in* ***Figure 5****)*. From left to right these show the edges detected by Sobel edge detection, Canny edge detection, and Laplacian of Gaussian, respectively.

These images contain information on the placement of the mask which can be compared with the outline of the face. These turned into vector forms are put into a data frame. At the same time this is being done, a truth value representative of if the mask is on correctly or not used is added as a 'feature' of the data frame. The original dataset source has the images split into these classes already, so the images from one file are known to be improperly masked while the other is properly masked. The program preprocesses and extracts feature from the images in one class file at a time, so it is already known what class is being added to the data frame. These are what goes into the implementation of the model.

Implementation

Here out the model is trained. The samples within the data frame formed in the previous section is split into training, validation, and testing sets with 60% going to training and 20% each going to validation and training.

From there, a model was made for each feature vector. Each model was made by training ResNet-18 over 15 epochs with a batch size of 128. Adam Optimizer was used with a learning rate of 0.001. The results of this implementation can be seen in the next section.

Results and conclusions

The following figure (**Figure 7**) shows the confusion matrix attained after testing the model:

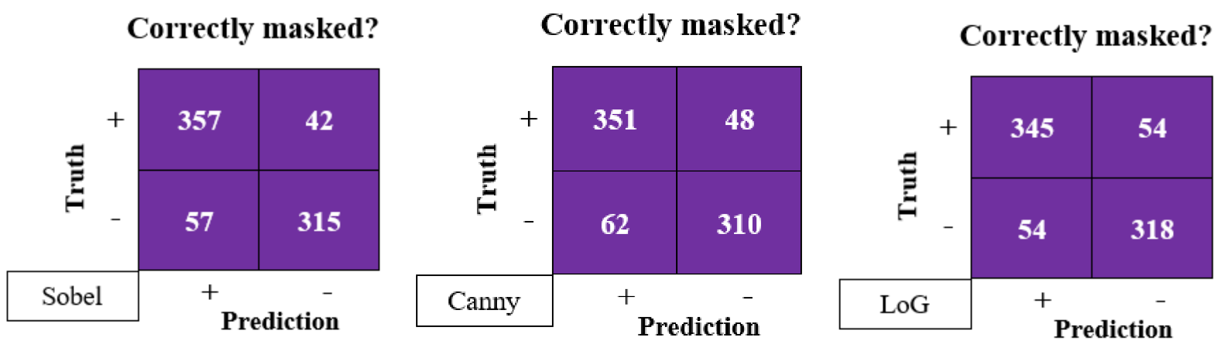

Figure 7: Confusion matrixes. This matrix shows the result of testing the models on the testing dataset. (“LoG” stands for Laplacian of Gaussian.)

The accuracies and F1 scores of the models are as follows:

```
Sobel Edge Detection >>  Accuracy: 0.8716   F1 Score: 0.8716
Canny Edge Detection >>  Accuracy: 0.8573   F1 Score: 0.8573
Laplacian of Gaussian >> Accuracy: 0.8599   F1 Score: 0.8599
```

We can note that the accuracies and F1 scores are near equivalent to each other since there was no significant class imbalance.

The model with the highest accuracy is the one using feature vectors from Sobel Edge Detection.

Project Management

The objective in the finalized code is to tell if the person in the image fed into the model is properly wearing his or her mask.

Implementation status report

Work completed

- Research on prior implementations for use as guides; Rohith, Amrutha, Blessy
- Datasets to be used finalized
- Datasets for training and testing; Rohith, Amrutha, Blessy, Brijesh
- Code for basic face detection and cropping; (found and reformatted by) Blessy

- Code for feature extraction; Brijesh
- Training and testing a model for mask detection with like images/videos; Brijesh, Blessy, Rohith, Amrutha
- Code for report samples: Brijesh, Blessy

Further work that can be done

- Relaying the classification to the assessed individual.
- Merging all components.

(GitHub Link: https://github.com/brijesh-rao/maskrcg)

References

https://docplayer.net/29870149-Long-term-effectiveness-of-dynamic-speed-monitoring-displays-dsmd-for-speed-management-at-speed-limit-transitions.html

https://pubmed.ncbi.nlm.nih.gov/8397665/

https://www.ijert.org/review-on-literature-survey-of-human-recognition-with-face-mask

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7775036/

https://projects.uplevel.work/insights/confusion-matrix-accuracy-sensitivity-specificity-precision-f1-score-how-to-interpret