Facial Detection with Masks

Computer Vision

Scott Bing

BSSD 6000 System Software Design

Jonathan Daniel Lee

December 1, 2020

Contents

Table of Figures

Abstract

Trying to control a pandemic in a nation of 330 million people has been challenging. In place of treatments and vaccines, medical science experts claim that wearing a mask is the best mechanism currently available to contain the virus. How many individuals comply with the mask suggestion? This project attempts to find the number of individuals who wear a mask. It uses Python's Open-CV package to recognize faces in a crowd video. This project uses the Caffe pre-trained model. Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR)/The Berkeley Vision and Learning Center (BVLC) and community contributors.[1] Using simple mathematical ratios it returns a percentage value of people who are wearing masks.

[1] https://github.com/BVLC/caffe

What Is Face Detection?

Face detection is the process of recognizing human faces in an image or a video. It should not detect non-human faces. A face has key indicators that distinguish it from other objects in the image. These key indicators are eyes, cheeks, nose, mouth, hair, and chin. The application is trained by scanning hundreds of thousands of images with faces and hundreds of thousands of images without faces. The application must be able to distinguish a human face even where all of its indicators are not present. For example, if a face is wearing sunglasses or simple eyeglasses, the application must still be capable of positively identifying the face. Even side profiles of faces are fed into the training module. Faces should be detected at any orientation.

Python's OpenCV package comes with trained facial classifiers called cascades. A cascade is a machine learning detection algorithm that identifies faces using their unique features. Cascades have facial feature classifiers for the eyes, nose, mouth, smile, side profile, etc. These classifiers come in two stages, training and detection. Training is performed on two sets of files, negative and positive. Negative images are those devoid of any faces. Positive images contain nothing but faces. The final training model is an XML data set that is passed on to the detector.

Facial Detection vs. Facial Recognition

This is not a Facial Recognition project. This is a Facial Detection project. Facial Detection does just that. It simply detects faces from other objects in the image or video It is just another category of image object recognition. Facial Recognition performs two tasks. It first detects faces; then it matches the face to a particular individual. Extensive training is involved with facial recognition. This project is only interested in distinguishing faces with or without a mask. It is not the goal of this application to identify who is wearing or not wearing a mask. This application is not used to enforce a mask mandate law or to reprimand anyone for not wearing a mask.

How does Facial Detection Work?

Face detection is a process of several steps. First, the face detection process finds every human face in an image. Then it analyzes what is known as the *Region of Interest* or *ROI* in each face. The *ROI* is the unique characteristics that make up a face, eyes, nose, mouth, chin, cheeks, hair, facial hair, glasses, etc. The face detection process draws a rectangular box around each face it locates. Faces also possess a different color and texture from other parts of the body.

Face detection is a very specialized form of image processing known as object detection. Images are made up of pixels. Pixels are arranged in rows and columns spanning the entire size of the image. Each pixel carries a color setting. Color is not needed to successfully identify faces. The image is converted to black and white which is referred to as grayscale. Each pixel is examined and compared to its surrounding pixels. As pixels go from light to dark the application tags this as a gradient. The process does this for every pixel in the image. When it is finished the gradients start to reveal the edges of a face and the features within. Figure 1 illustrates this process. You can see the edges of the face revealed from the gradients. This process is known as the *<u>H</u>istogram of <u>O</u>riented <u>G</u>radients, HOG*.

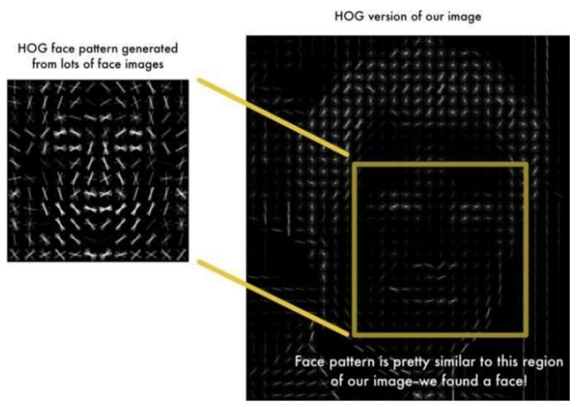
HOG version of our image
HOG face pattern generated from lots of face images
Face pattern is pretty similar to this region of our image–we found a face!

Figure 1: Histogram of Oriented Gradients HOG

Once the pixels outlining the face have been identified, the process must now deal with faces that are not oriented straight on. These are faces that are looking down, up, and to the side. The facial detection software must still identify these poses as legitimate faces. The face detection software uses an algorithm called *Face Landmark Estimation, FLE*. The faces are warped so the *ROI* positions always remain the same. The algorithm creates 68 points that identify the edges and features of the face. The machine will be trained to pick out these points. Figure 2 illustrates the FLE in action.

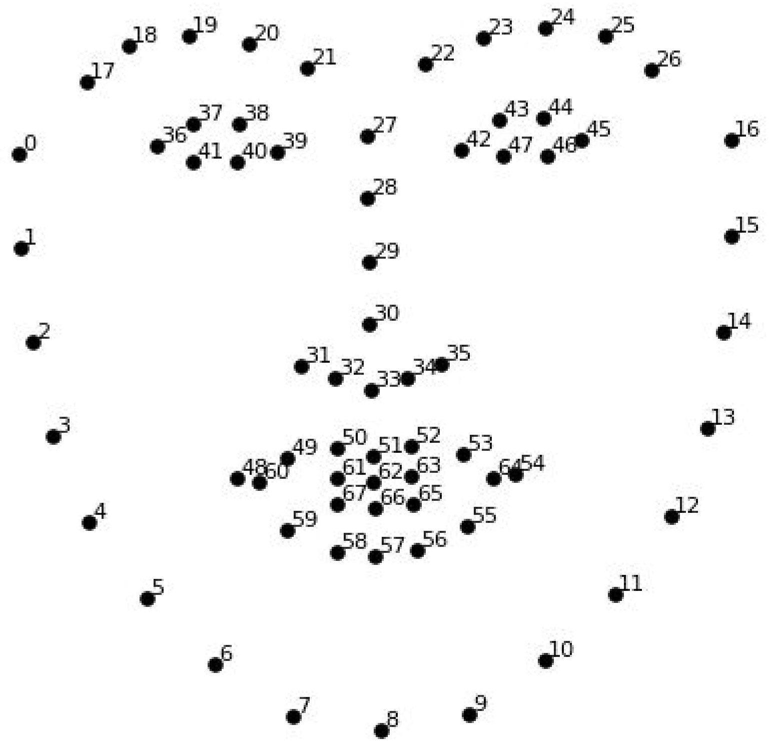

Figure 2: Feature Landmark Estimation FLE

To handle faces that are turned or facing down or up, the process, scales, rotates, warps, and centers the *ROI* so that everything lines up. This makes it easier for the software to find all of the facial characteristics.

Training Process

The training process happens in two phases. The model includes a face mask classifier. A classifier separates the faces in a simple binary fashion. The first phase prepares the face mask detector classifier for processing by the second training phase. Faces with masks are separated from faces without masks. Figure 3 graphically represents the concept of classification.

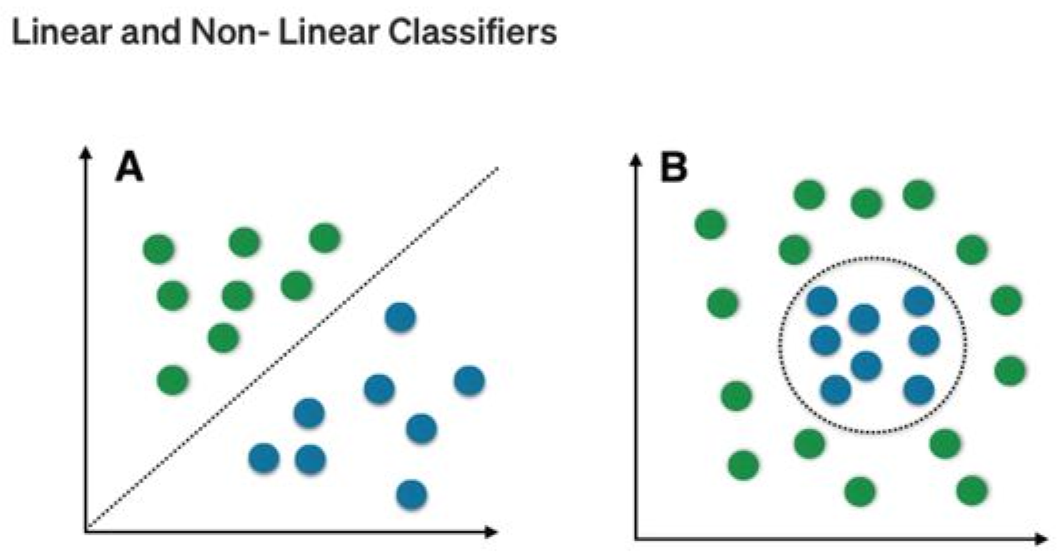

Figure 3: Linear and Non-Linear Classifiers

Let's say the green dots represent data points for the faces with masks and the blue dots represent those without. A graphic object, al line, or a closed polygon represented by a circle in the diagram separates the two groups. No probabilities are calculated as in linear regression. Classification simply identifies each group of objects by a chosen characteristic, in our case the presence or absence of a mask.

The classifier is trained to utilize Python Keras/Tensorflow functions. This process builds the final face mask detection classifier. The face mask detection classifier is marshaled into a data stream.

During the second phase, the face mask detector classifier is applied to each face, both positive and negative. The training module detects the *Region of Interest, ROI*. This is the region of the face where a mask would typically be worn. I also include areas where the facial characteristics occur, the eyes, nose, mouth, chin, hair, eyebrows, etc. At this point, the training module makes the final determination of whether the face includes a mask or not. Figure 4 illustrates this process.

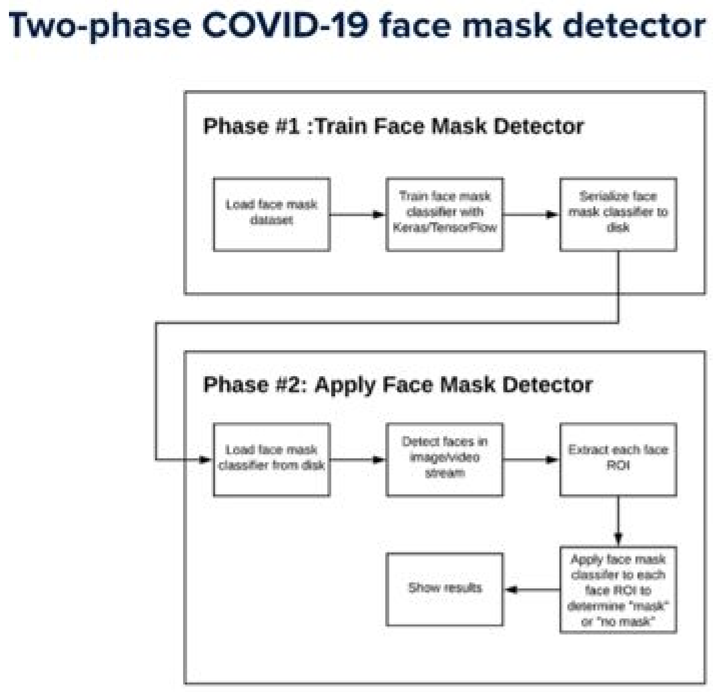
Two-phase COVID-19 face mask detector
Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifier to each face ROI to determine "mask" or "no mask"
Show results

Figure 4: Two-phase Covid19 face mask detector

Face Mask Detection Data

The training data is divided into 2 folders, positive and negative. The positive folder contains images of faces wearing face masks, the negative folder contains images of faces without masks. During the training process, the training module is fed the set of negative images. This is established as the baseline. Then the set of positive mages is shown to establish the identity of a face with a mask.

Here is a sampling of the training and detection data.

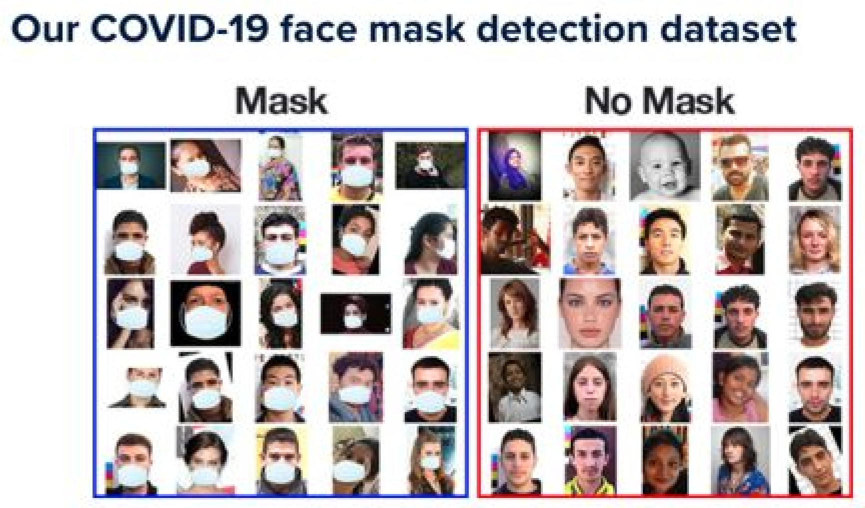

Figure 5: Covid-19 face mask detection dataset

The application also includes two pre-built models, *deploy_prototxt* and *res10_300x300_ssd_iter_1400000.caffemodel*. *deploy_prototxt* is the training settings module. It is a JSON file that sets up the parameters for the training session. The *res10* file is the actual training model.

The Facial Detection Application

The Facial Detection application is designed to execute against images as well as videos. There are two scripts *detect_mask_image.py* for images and detect_mask_video. The application uses runtime parameters to input images and videos, the pre-trained models, and a confidence rating. A confidence rating is a number between 0 and 1. The confidence setting sets the minimum probability of filtering weak detections. This allows more images to be identified that would normally be skipped due to poor image quality.

Pressing the letter *q* on the keyboard will quit the applications. When a video is examined the user has the choice of waiting until the video is complete or pressing *q* on the keyboard to close the display.

There are several Python face detection pre-trained models to choose from. This application uses the Caffe model. Python's OpenCV package provides a *Haars* model. The *Haars* models offer not only a generic face detection model but also several distinct selective models. These include a facial profile model and a facial covering model. The *Caffe* model was chosen for this project because it is easy to install and its generic model covers facial covering and facial profiles.

Facial Detection Challenges

On occasion, the computer misses the identification of a face in an image. Sometimes the faces are too small or too far away from the camera for the computer to detect facial features. For faces with masks, sometimes the color of the mask matches the skin color of the individual. Facial pose and movement can depreciate facial detection. This is more evident using face detection with video capture.

It is difficult to get an accurate mask utilization percentage in a video as every time a face moves the application generates a new generation of the facial bounding box. This skews the count of actual individuals wearing a mask. If the bounding box is used to tally the number of individuals wearing a mask, it will be prone to error as there could be multiple bounding boxes per individual. This is not an issue with a still image

The application did not catch all of the faces in the images. With any technology, there are limitations. If the faces are too far away from the camera, it might be difficult for OpenCV to recognize them. Further training of the model might eliminate these limitations. Of course, that involves executing the training module on a high-powered computer with access to big data. A simple laptop may not be able to handle such a task. This why it is recommended that novices use pre-trained models.

Another factor that limits the success of facial detection is the image or video quality. If the image quality is poor, the application will have issues identifying a face. Faces might not be recognized when the image is smudged, distorted, or blurred images, however with targeted training, an application can recognize faces with these conditions. Some facial recognition applications can still detect faces with glasses, masks, and other facial accessories.

Face detection quality in a video is affected by head movement. A human head does not remain stationary. It moves every second. An application cannot recognize a face when the head is continually moving. This is why the identification rectangles on a video flash continually. If the video identifies a face in a single frame, facial detection in this case is a success.

One of the biggest issues with successful face detection is image resolution. Any degradation of resolution will affect the ability of the application to detect a face. This is a simple fact of image analysis. Any reduction in the quality of an image will affect the outcome of its analysis.

Results

A simple statistic was recorded for each bounding box. A green bounding rectangle is placed around a face wearing a mask. A red rectangle surrounded a face without a mask. A tally is recorded each time a box was created. If the box is green, the face wearing mask count is incremented. If the box is red, the face not wearing mask count is incremented. These tallies are calculated by the total number of boxes and multiplied by 100 to get a final percentage. This statistic appears in the title of the screen. Figure 6 shows the bounding boxes and the statistics in the image title bar.

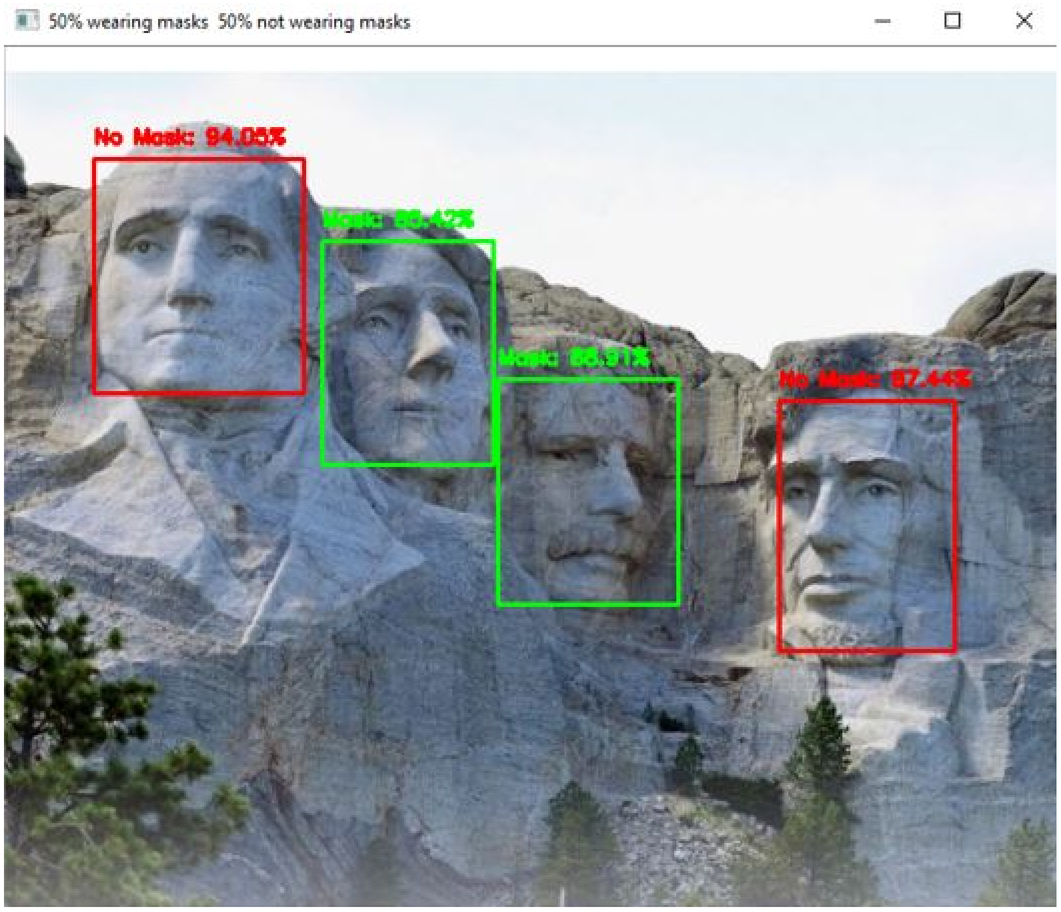

Figure 6: Bounding Boxes and Statistics

This simple statistic is accurate for a static image. It is problematic to calculate this statistic for a moving image. The boxes are drawn for each frame of the video. It could be done, but it would be challenging. A frame would need to be connected to an individual face. The count would be incremented once based upon an individual face rather than a bounding box. The issue here is that there are multiple bounding boxes created for each face. The code could still tally by bounding box, but it would have a second check to see if it were dealing with the same face. If it is the same face do not increment the tally. Only increment the count when a new face is encountered. The software would need to train on every face in the video to correctly tally the number of faces with masks and without.

The calculated statistic is only accurate for the number of faces that the application recognizes. In a crowd image, the application may only capture a small number of faces out of the crowd. Camera distance and degraded image quality prevent the application from capturing every face in the crowd. This is simply a relative measurement based upon the number of faces detected.

Conclusion

The application did detect faces with and without masks. With still photos, the application did an adequate job. There were some shortcomings. Some faces were either too far away or too blurry to detect. Adjusting the confidence level did capture a few more faces. Pictures with crowds did not capture all of the faces. This made it difficult to determine the true count of those with masks and those without.

Faces with masks in videos presented additional challenges. The movement caused the application to detect the same face with and without a mask. It also marked the same individual face numerous times. This made it difficult to get a good estimate of how many persons were wearing masks. The get a more accurate count, the application would need to additionally identify each face.

This application could be useful to get a good sense of how many people were observing disease prevention guidelines. It would not be too intrusive as it does not identify who o is or is not wearing a mask. This is not a mask mandate enforcement application. It simply gives officials a sense of how well a particular segment of society is observing the wearing of masks

Bibliography

“A Gentle Introduction to Deep Learning for Face Recognition.” *Machine Learning Mastery*, 30 May 2019, machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/.

Chojecki, Przemek. “Crazy GPT-3 Use Cases.” *Medium*, 30 July 2020, medium.com/towards-artificial-intelligence/crazy-gpt-3-use-cases-232c22142044. Accessed 16 Nov. 2020.

“COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning.” *PyImageSearch*, 4 May 2020, www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/. Accessed 16 Nov. 2020.

Geitgey, Adam. “Machine Learning Is Fun! Part 4: Modern Face Recognition with Deep Learning.” *Medium*, Medium, 24 July 2016, medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

Hjelmås, Erik, and Boon Kee Low. “Face Detection: A Survey.” *Computer Vision and Image Understanding*, vol. 83, no. 3, Sept. 2001, pp. 236–274, www.sciencedirect.com/science/article/pii/S107731420190921X, 10.1006/cviu.2001.0921.

Kulkarni, Shraddha, and Gururaj S. P. “Deep Learning Based Object Detection Using You Only Look Once.” *International Journal of Research in Advent Technology*, vol. 7, no. 4, 10 Apr. 2019, pp. 9–12, 10.32622/ijrat.74201902.

Liu, Wei, et al. *SSD: Single Shot MultiBox Detector*. 2016.

“PoseNet.” *Www.Edgee.co.uk*, www.edgee.co.uk/demo/_posenet/index.html. Accessed 22 Nov. 2020.

"Priya-Dwivedi/Deep-Learning." *GitHub*, github.com/priya-dwivedi/Deep-Learning/tree/master/Question_Answering_coronavirus. Accessed 16 Nov. 2020.

Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, 1 June 2017, pp. 1137–1149, 10.1109/tpami.2016.2577031.

Singh, Shilpi, and S.V.A.V. Prasad. "Techniques and Challenges of Face Recognition: A Critical Review." *Procedia Computer Science*, vol. 143, 2018, pp. 536–543, 10.1016/j.procs.2018.10.427.