Real-time Face Mask Detection using MTCNN, PCA, and SVM

Aan Patel, Kayla Dutcher, Rohil Bhattarai, Shaswat Joshi

November 2020

1 Introduction

Quite some effort has been put into public-space computer vision applications to help combat the ongoing COVID-19 pandemic. One such application, **Real-time face mask detection**, involves first the detection of faces in a given frame of a video feed, and then the classification of each face to detect if the person is wearing a mask or not. If implemented nationwide, such Computer Vision solutions can have a profound impact on the spread of the virus that has created a pandemic.

Support Vector Machines (SVMs) have been proven to be great at a variety of classification and regression problems. We aim to create an application using a machine learning model that utilizes **Principal Component Analysis (PCA)** to reduce the dimensionality of image data, and then classify the image of a face as **with_mask**, **without_mask**, or as **mask_weared_incorrect** (n.b: this name was a grammatical error in the labels. We will use it as-is to stay in agreement with the dataset used) with the help of an SVM.

We aim to create a real-time classification application that detects faces from a webcam using a pre-trained Multi-Task Cascaded Convolutional Neural Network (**facenet-pytorch MTCNN**).

2 Application Structure

Real-time face mask detection consists of 4 steps:

1. Sampling frames from the live feed from a camera
2. Finding all the faces and saving their bounding box coordinates in a given frame using MTCNN
3. Passing each individual face to the PCA-SVM pipeline and classifying them as either **with_mask**,**without_mask**, or as **mask_weared_incorrect**

4. Displaying the result as the frame with a colored frame overlay which is green (**with_mask**), orange (**mask_weared_incorrect**), or red (**without_mask**)

3 Dataset Selection for the Classifier

For training the model required for classifying face images, we chose the **face-mask-detection** dataset from Kaggle.
The dataset consists of two folders:

- **annotations/**, which contains XML files with metadata including:
    1. The image filename
    2. Labels (**with_mask**,**without_mask**, or **mask_weared_incorrect**)
    3. Face bounding boxes for the corresponding training image.
    4. The name of the XML file is formatted as: [name_of_image].xml
- **images/**, which contains PNG image files

The data was processed using Python into:

1. A numpy array, **data**, where each element is a flattened float32 array generated from 224x224 scaled RGB images of extracted faces based on the annotation XML file data.
2. A numpy array, **labels**, that contains the string label for each corresponding sample in the **data** array.
3. A numpy array, **label_data**, that contains an integer label (0, 1, 2) for each corresponding sample in the **data** array.

The **data** and **labels** array construction process was borrowed from [4].

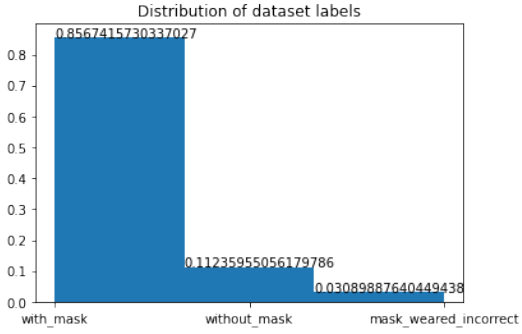

Figure 1:

	precision	recall	f1-score	support
mask_weared_incorrect	0.00	0.00	0.00	2
with_mask	0.93	0.96	0.94	79
without_mask	0.57	0.5	0.53	8
accuracy			0.90	89
macro avg	0.5	0.49	0.49	89
weighted avg	0.87	0.90	0.89	89

Table 1: Classification Report

4 Classifier Model - Training and Analysis

We used the PCA algorithm offered by the **scikit-learn** library to extract the top 70 principal components and thus reduce the number of features the SVM classifier will need to deal with.

The output of the PCA algorithm was pipelined into the SVM classifier provided by the **scikit-learn** library.

The SVM classifier uses a radial basis function (**rbf**) kernel, which is a proven method [3] for clustering high-dimensional data.

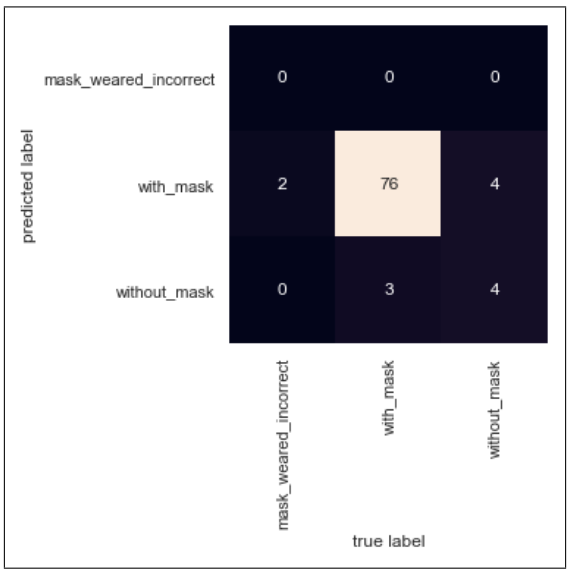

Figure 2: Confusion Matrix

We achieved an accuracy of of 89.88% with C=20 and γ=0.005 for the SVM. For finding the SVM hyperparameters C and γ, we used the **GridSearchCV**

cross validation helper from the **scikitlearn** library. From the confusion matrix in Figure 1, and the classification report in Table 1, we can conclude that our model can easily identify the **with_mask** class, and can make fair sense of whether or not a person has worn a mask, but it fails to detect improperly worn masks. This is primarily because of the lack of more samples for the **mask_weared_incorrect** and **without_mask**.

5 Face Detection using MTCNN

MTCNN provides a robust, pre-trained neural network model ideal for detecting faces on a live camera feed. We used it on our application to provide a cropped and scaled image of only the detected face to the PCA-SVM pipeline on the fly. The model output is then used to color the bounding box of the detected face.

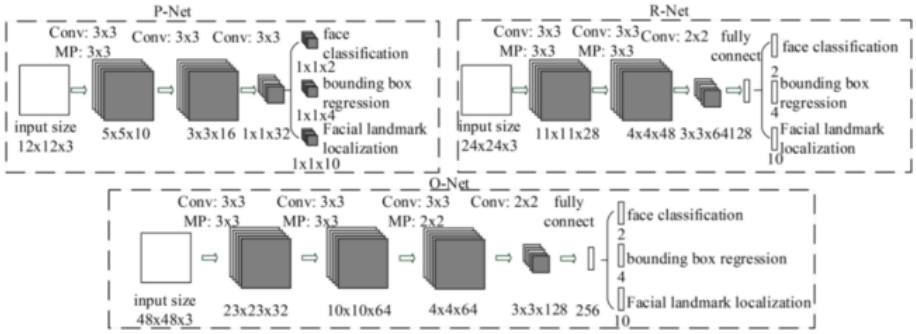

Figure 3: A quick overview of the basic structure of the MTCNN Model.[2]

MTCNN features 3-stages, each having a deep neural network with several layers:

1. The P-net - determines the probability of a face in a given kernel
2. The R-net - refines the coordinates of the boxes bounded by the P-Net
3. The O-net - takes R-net bounds and then determines facial landmarks

Figure 2 reproduces the MTCNN neural network structure.

6 Conclusion

Though our PCA-SVM classifier fails to usually detect the **mask_weared_incorrect** class, as can be observed in Figure 4, it can clearly make out the difference between the **with_mask** and **without_mask** classes. So, given the accuracy of the classifier at doing this important classification, we are satisfied with our application.

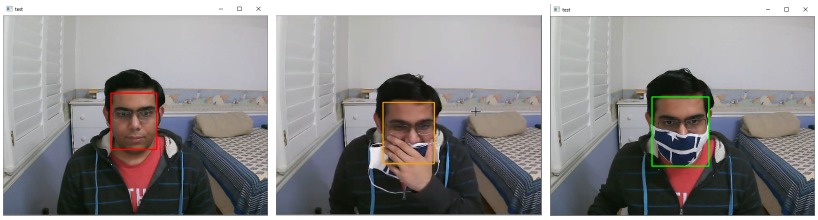

Figure 4: Application output: red box - without mask, orange box - mask worn incorrectly (rarely detected), green box - with mask. Application code available at: https://github.com/aannirajpatel/COMP562-Face-Mask-Detect

References

[1] Chi-Feng Wang. "What Does A Face Detection Neural Network Look Like?" Towards Data Science. July 24, 2018.
https://towardsdatascience.com/face-detection-neural-network-structure-257b8f6f85d1

[2] Kaipeng Zhang, Zhanpeng Zhang and Zhifeng Li. "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks."
https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf

[3] Jake VanderPlas. "In-Depth: Support Vector Machines." Python Data Science Handbook. November 2016.
https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

[4] Kaggle user; notadityabhat. "Face Mask Detector" July 2020. kaggle.com.
https://www.kaggle.com/notadithyabhat/face-mask-detector

[5] Kaggle user; Larxel. Face Mask Detection Dataset. Kaggle. May 2020. kaggle.com
https://www.kaggle.com/andrewmvd/face-mask-detection

[6] Kaggle user; timesler. Guide to MTCNN in facenet-pytorch. May 2020. kaggle.com
https://www.kaggle.com/timesler/guide-to-mtcnn-in-facenet-pytorch