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

Our real-time face mask detection application consists of 4 steps:

1. Sampling frames from the live feed from a camera
2. Finding all the faces through their bounding box coordinates given a frame, through 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 directories:

- **annotations/**, which contains 853 XML files with metadata including:
  1. The image filename
  2. Labels for each face in each image (**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 853 PNG image files

The data was processed using Python into:

1. A numpy array, **data**, where each element is a flattened float32 array of length (224x224x3) 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].

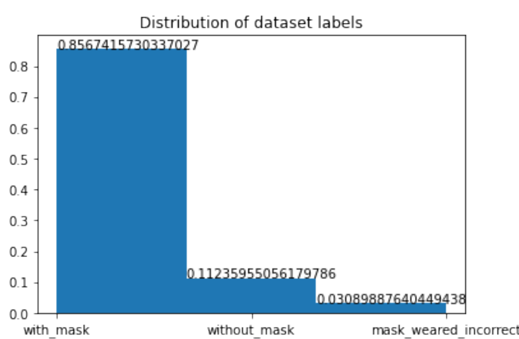

Figure 1: Dataset Label Histogram

4 Classifier Model - Training and Analysis

We used the PCA algorithm offered by the **scikit-learn** library to extract the top 140 principal components and thus reduce the number of features the SVM classifier will need to deal with.

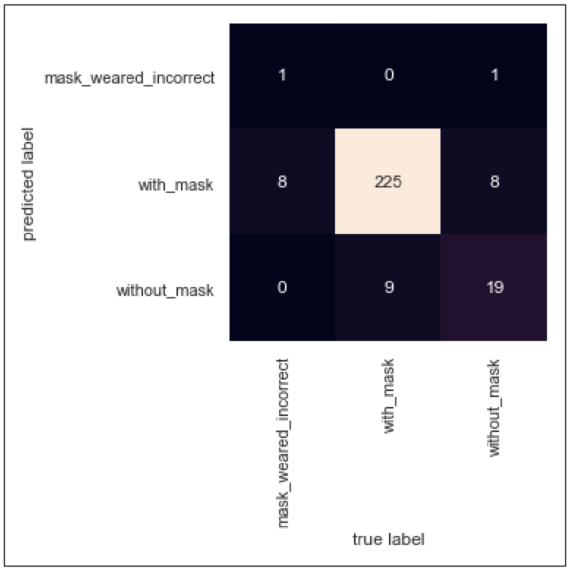

Figure 2: Confusion Matrix for the PCA-SVM classifier

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.

	precision	recall	f1-score	support
mask_weared_incorrect	0.50	0.11	0.18	9
with_mask	0.93	0.96	0.95	234
without_mask	0.68	0.68	0.68	28
accuracy			0.90	271
macro avg	0.70	0.58	0.60	271
weighted avg	0.89	0.90	0.89	271

Table 1: Classification Report for the PCA-SVM classifier

We achieved an accuracy of of 90.40% with hyperparameters C=1 and γ=0.005 for the SVM, found using the **GridSearchCV** cross validation helper from the **scikitlearn** library. From the confusion matrix (Figure 2), and the recall values in the classification report (Table 1), it is clear that the model is able to differentiate between the **with_mask** and **without_mask** classes. Due to lack of enough training samples however, it fails to detect improperly worn masks.

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.

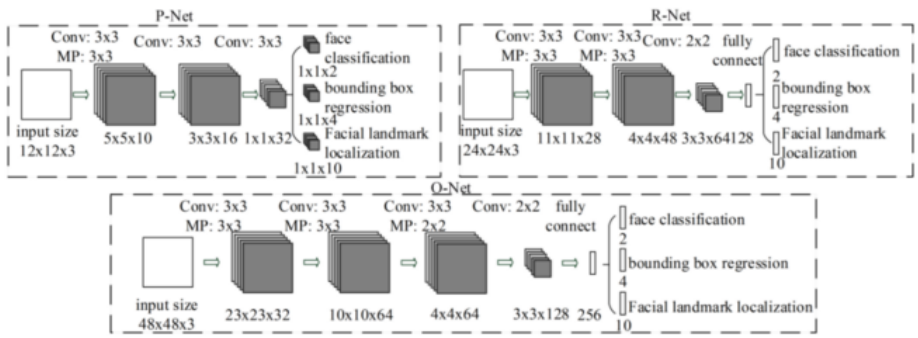

Figure 3: A quick overview of the basic structure of the MTCNN Model.[2]

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

6 Conclusion

Our model fails to usually detect the **mask_weared_incorrect** class. However, Figures 4 and 2 show that our model is able to perform the essential classification between the **with_mask** and **without_mask** classes.

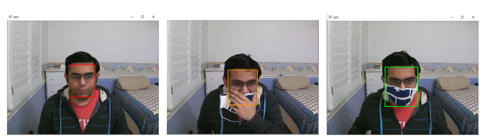

Figure 4: App output: red - no mask, orange - improper, green - with mask

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