MaskieFace: Enhancing Masked Face Recognition During the COVID-19 Pandemic

Siyi Zheng
University of Michigan
School of Information
zhengsyc@umich.edu

Yan Long
University of Michigan
EECS Department
yanlong@umich.edu

Li Chen
University of Michigan
Robotics Institute
ilnehc@umich.edu

Yuguo Zhong
University of Michigan
Civil and Environmental Engineering
yuguoz@umich.edu

ABSTRACT

Due to the impact of COVID-19, wearing masks has become part of people's life. Existing face recognition algorithms, however, suffer from degraded performance when recognizing masked faces. In response of this, our team used various computer vision techniques to improve the existing models' accuracy of masked facial recognition. We developed algorithms capable of adding various types of masks onto no-mask images in existing face recognition data sets to achieve data augmentation. We applied the FaceNet model and machine learning algorithms to construct the recognition pipeline. In general, our model had a 16% increase in recognition accuracy compared to the baseline, with the SVM poly classification achieving the highest accuracy of 85%.

1 DATA PROCESSING

For this project, our team used the Labeled Faces in the Wild (LFW) Face Database[2], which contains 13233 images for 5749 people.

Training Set and Testing Set Preparation

In order to make the experiment rigorous, we decided to keep the number of images same for each person.

We revised the data set to test the performance of our model for different levels of occlusion and different styles of masks. There are two kinds of data sets which were used in this project. The first data set contains a training set of images with people not wearing masks and a testing set of images with people wearing masks, and is used as the baseline. The other data set's training set is a mixture of people wearing masks and without masks, and is used in our revised model.

GitHub repository: https://github.com/ilnehc/MaskieFace.

Applying Masks Automatically

Due to the lack of masked face data sets, we developed an algorithm to add masks to the images in the LFW Database based on the model developed by Aqeel Anwar[1].

The model uses dlib package to find landmarks from images of people's face. The landmarks include eyebrows, eyes, nose bridge, nose tip, chin and mouth, which are shown in Figure 2(a). Our team revised the code to generate masked faces with different occlusions as shown in Figure 1.

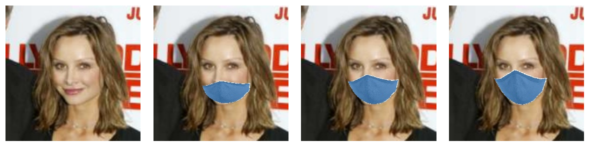

(a) Landmarks (b) Low (c) Normal (d) High

Figure 1: The original face and the masked faces with different levels of occlusions.

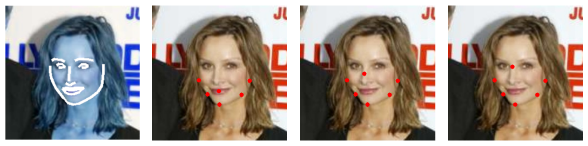

(a) Landmarks (b) Low (c) Normal (d) High

Figure 2: Face landmarks and mask key points with different occlusions.

After finding the landmarks of the human face, we need to find the key points as shown in Figure 2 to locate the masks.

The first pair of points are near the cheekbones, which are computed as the intersections of the chin and the horizontal line which crosses the middle of the nose. The second pair of points are near the bottom of the lip, which are computed as the intersections of the chin and the horizontal line which crosses the bottom of the lips. The fifth point is the intersection of the chin and the perpendicular line which crosses the nose.

The last point, which controls the occlusion of faces, is chosen from the lip-mid-point, the nose-mid-point, and the eyes-mid-point, which are computed as the mean of the landmark points accordingly. For the low occlusion situation, the mask only covers the face below the nose. Thus, the lip-mid-point is a great candidate feature point. For the general occlusion situation, the mask covers half of the face. The mid point of the nose is a good feature point splitting the human face to two equal parts. As for the high occlusion situation, the mask should cover as high as possible and shouldn't cover the eyes. Hence, the mid point of the eyes should be the highest point where the mask could reach.

2 FACE RECOGNITION

A modern face recognition pipeline consists of 4 common stages: detect, align, represent and verify. Before we perform face recognition, we need to detect faces, finding the location of the face on a image and cropping it to a certain size. We chose the state-of-art Multi-Task Cascaded Convolutional Neural Network (MTCNN) [8] for face detection. This model is implemented as a python library so faces can easily be extracted and resized as our final train and test data.

At present, several deep neural networks such as VGG-Face[3], Facebook Deepface[5] and Google FaceNet[4] have been developed. Based on Wang's[7] survey, FaceNet can achieve a relatively higher accuracy on the fundamental face recognition. Therefore, we chose it as our recognition model to embed the face as a 128 dimensional feature vector.

With all training and testing data embedding as feature vectors, multiple classification machine learning methods are implemented to verify the person, such as Soft Vector Machine (SVM), Multi-layer Perceptron Neural Networks (MLP), K-Nearest Neighbors (KNN) and Random Forest. They will predict the testing data based on the training dataset and the labels will be compared with the ground truth label to get the final accuracy results.

FaceNet

FaceNet is a state-of-art face representation or recognition system which achieves a record accuracy of 99.63% in the LFW dataset. FaceNet introduces a triplet loss in its training process which minimizes the squared distance between all faces of the same identity, whereas maximizes the squared distance between a pair of face images from different identities in the feature space $\mathbb{R}^d$, as shown in Figure 3.

We used the pre-trained Keras FaceNet model and weights provided by Hiroki Taniai[6] for the deep architecture. With each image as input, the model will do a forward pass to predict the image as a 128 element feature vector.

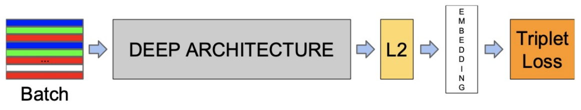


Figure 3: FaceNet model structure[4].

Classification

SVM. SVM is one of the most effective algorithm to do multi-class classification. A hyperplane is selected to best separate the points in the input variable space by their class and it is learned from training data using an optimization procedure that maximizes the margin, which is calculated as the perpendicular distance from the line to only the closest points. Meanwhile, different kernels can transform the input to another dimension to get better performance. We applied linear kernel (1) and polynomial kernel (2) for test.

$$K(x, x_i) = sum(x \cdot x_i) \tag{1}$$

$$K(x, x_i) = 1 + sum(x \cdot x_i)^d \tag{2}$$

MLP. MLP model optimizes the log-loss function using LBFGS or stochastic gradient descent (ADAM optimization algorithm in our code). We only used one hidden layer for the classification process so it is a vanilla neural network at this circumstance.

KNN. KNN is an easy and fast algorithm and doesn't require training prior to making predictions. Since each person has 4 no-mask images or 2 masked and 2 unmasked images, the nearest neighbor value K is set to 4 and euclidean distance is used for all tests.

Random Forest. Random forest uses bagging and feature randomness to create a large number of individual decision trees that operate as an ensemble, an uncorrelated forest. Based on tests, we adopted entropy criteria for splitting and a random state value 3 was preferred.

Results and Analysis

We explored five situations as stated in 1, faces covered by N95 masks, surgical masks, cloth with 3 different levels of occlusion.

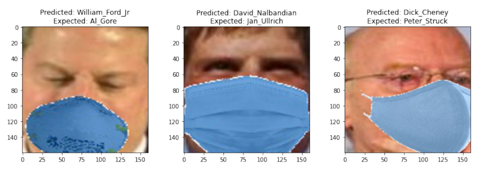
The face recognition results are shown in Table 1. In all of the five data sets, adding masked faces into the training set shows considerable improvement. The final accuracy for normal mask cases such as disposable surgical masks and normal height cloth masks can both achieve 85% or higher. Four aspects of comparisons including the effect of levels of occlusion, types of masks, classification methods and overall performance compared to baseline will be summarized as the following respectively.

Apparently, in three cloth data sets, with more area masked and more face feature information lost, i.e., from low, normal to high, the recognition accuracy will drop accordingly. However, we can see that there is an average 10% increase from normal to low occlusion which indicates that nose is one of the most important features for recognition in this model. It is not surprising to see an average 4% drop in accuracy when people try to lift their mask from the middle of the nose to the bridge of the nose as there are few feature points being covered in that area.

It has been noticed that among the three mask types, N95 masks covers smallest area on human faces. However, the accuracy based on dataset where people wear N95 masks turned out to be the lowest. One possible reason is using the N95 mask, the bottom half of faces remain features of both the mask and the chin. The model might mis-recognize the chin and the mask. While under the case of cloth masks or surgical masks, only the feature of the mask is remaining, and no mis-recognition occurs.

Most importantly, the model trained by including the masked face pictures improved the accuracy of about 16% by average compared to the baseline with the highest of 85% in accuracy by SVM poly classification method considering all kinds of masks and all levels of occlusion.

We also notice that there are some failure cases as shown in Figure 4. They usually happen when the original image has 'bad' illumination or a view angle. Occlusion leads to the loss of a great part of the face features.



(a) N95 (b) surgical (c) cloth

Figure 4: Failure cases.

3 EXPLORATIONS: MASK REMOVAL

Besides the loss of information caused by the masks on human's faces, another issue the masks introduce is the increased level of noises. People wear masks in unpredictable ways, with the masks being in various positions, shapes, colors, etc. The non-uniform patterns of masks on people's faces are thus a potential source of noises that could lead to performance degradation in masked face recognition. In view of that, we intended to explore the feasibility of removing the masks in the images before they were input into the neural network, and investigated the influence it might have on recognition accuracy. Specifically, for each image of a person wearing a mask, we want to find the boundary of the mask and "remove" all the pixels within that boundary by assigning 0 to those pixel values. We want to find the exact boundary of the mask in order to minimize the information loss, as compared to a coarser mask removal approach where the lower half of the face are all removed. In this project, we tried two image segmentation methods introduced in class.

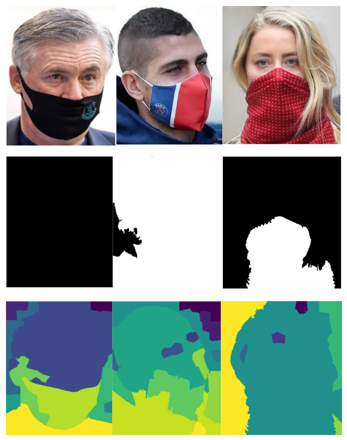


Figure 5: Examples of using graph-cut segmentation (2nd row) and MSF segmentation (3rd row) to identify the boundaries of masks and conduct mask removal. It's impossible to find a set of parameters that work for the various images.

Graph-cut Segmentation

The first algorithm we tested is the graph-cut based Foreground-background segmentation. In this particular problem, the only foreground object is the mask. The problem setup differs from that of the homework problem since now there's no user that will choose a foreground superpixel. As a result, we have to make the program to automatically find the first foreground superpixel. To that end, we combined the naive approach stated above, and used the superpixel positioning at 1/2 the width, and 2/3 the height of the face rectangle as the foreground pixel. The reason this coarse method works here compared to above is that the whole mask occupies a relatively large area compared to the top boundary alone of the mask, so there's larger room for tolerance.

The graph-cut based algorithm produces very unstable segmentation depending on the superpixel chosen and the patterns of the masks. Masks of single colors are easier to segment and the algorithm can identify the boundaries of the masks well sometimes. But at other times it fails because of variations in which superpixel is chosen as the foreground pixel. In order to overcome that in the failure cases, we had to tune the distance metrics of the color histogram and the pixel positions, but unfortunately, different pictures require quite different configurations. Another problem is that for masks that have more than one color blocks, the algorithm usually fail to group the blocks of one mask together. A possible solution to this problem is running the algorithm multiple times, each time with a different superpixel as the

		SVM linear	SVM poly	MLP	KNN	Random Forest
N95	Baseline	**67.711**	62.410	58.795	58.795	46.265
	Ours	79.036	79.036	**80.964**	74.940	75.663
Surgical	Baseline	69.903	62.864	62.864	**70.631**	53.155
	Ours	84.223	**85.437**	83.495	79.126	76.699
Cloth Normal	Baseline	73.659	69.765	67.073	**76.341**	64.146
	Ours	85.366	**86.098**	83.659	81.463	75.122
Cloth Low	Baseline	**89.021**	87.828	84.487	88.544	84.010
	Ours	93.795	**94.033**	93.079	92.363	87.828
Cloth High	Baseline	**70.343**	63.725	58.578	68.873	60.294
	Ours	79.657	**81.863**	79.902	74.510	72.794

Table 1: Masked face recognition accuracy results. Baseline contains a training set of unmasked faces and a testing set of masked faces. Ours has a training and a testing set, but the training set is a mixture of masked and unmasked faces.

foreground pixel, and finally combine the results of different runs. But we notice that the computational time is quite long, making it impractical.

Minimal Spanning Forest Segmentation

Using MSF does not require the input of foreground pixel on the mask, but requires post-segmentation grouping. We inspected the results of MSF segmentation on the pictures we have, and found it usually unable to determine the boundary of the masks properly. Generally, the problems are similar to those of the graph-cut algorithm. Different images require different fine-tuned parameter setups. For example, for masks with multiple color blocks, it requires the size of the grouping filter to be set larger to cover the whole mask.

Summary

Using segmentation-based method to remove the masks in the images is infeasible. The first reason is that there are large variations in the masks' colors, shapes, etc., and the segmentation-based algorithms cannot capture the invariance among these variations. This is understandable, because the algorithms involve no concept of invariance with regard to detecting the masks on people's faces. The major consequence is that the algorithm is not robust and it requires fine-tuning of their parameters to make them work with a certain type of mask, which is impractical.

4 CONCLUSION

In this study, we improved the accuracy of masked face recognition in response to the COVID-19 pandemic. We developed an algorithm to add three types of masks to the LFW data set with three levels of occlusions to simulate different situations in the real life.

FaceNet was combined with various classification methods such as SVM, KNN and Random Forest. The baseline was trained with unmasked faces while the our model was trained with the mixture of masked faces and unmasked faces. In general, our model had a 16% increase in accuracy compared to the baseline while the SVM poly classification achieved 85% in accuracy which was the highest. We also explored the possibility of utilizing existing image segmentation algorithms to identify the boundaries of masks and remove them in order to reduce the noises introduced by various mask patterns in face recognition, but found the task non-trivial and requires purpose-built algorithms.

ACKNOWLEDGMENTS

Prof. Jason J. Corso from the University of Michigan EECS Department; GSIs [Oscar De Lima, Parker Koch] for EECS 504 at the University of Michigan.

REFERENCES

[1] Aqeel Anwar and Arijit Raychowdhury. 2020. Masked Face Recognition for Secure Authentication. arXiv:cs.CV/2008.11104

[2] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report 07-49. University of Massachusetts, Amherst.

[3] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Face Recognition. In *British Machine Vision Conference.*

[4] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2015). https://doi.org/10.1109/cvpr.2015.7298682

[5] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition.* 1701–1708. https://doi.org/10.1109/CVPR.2014.220

[6] Hiroki Taniai. 2018. keras-facenet. https://github.com/nyoki-mtl/keras-facenet.

[7] Mei Wang and Weihong Deng. 2020. Deep Face Recognition: A Survey. arXiv:cs.CV/1804.06655

[8] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23, 10 (Oct 2016), 1499–1503. https://doi.org/10.1109/lsp.2016.2603342