Building An Accurate Detector for Masked Faces in the Pandemic Era

Zishen Li
li.zis@northeastern.edu

Junmei Luo
luo.jun@northeastern.edu

Tony Yukuang Zhang
zhang.yuk@northeastern.edu

Github:
https://github.com/Tonyz4516/detector-for-masked-faces

1 SUMMARY

Since March, Covid-19 has taken a heavy toll on American societies. As suggested by the health experts and the governments, having everyone in the community wear a mask can help mitigate the risk of asymptomatic people spreading the virus. To improve people's awareness and help businesses monitor the points of entrance and enforce the social distancing policies, society calls for better contactless monitoring and identity verification methods.

In such a situation, corresponding masked face detection algorithms can be designed to help monitor whether people wear masks while in public places, such as markets, groceries, and offices. This can help the society fight against the virus by increasing the awareness of wearing masks and reduce the cost of relying people to remind. Furthermore, non-masked face recognition and masked-face recognition could be explored separately. If a person is not wearing masks, we can use face recognition to find out the person's name and information, and give him or her some reminders. On the other hand, masked-face recognition can be used at the gate of offices to recognize employees without taking off masks. We can also extend non-masked face recognition to more complex environments, and multi-face with the background can be one possible way. Given the limited time and resources, the project may touch 2-3 sections mentioned above.

We utilized the Real-World Masked Face Dataset from Github, which includes real-world masked faces, faces with simulated masks, and non-masked faces. CNN is considered the most common deep learning method in facial recognition, and we adopted the idea of transfer learning and began with a pre-trained CNN model.

In the first phase, we have completed data cleaning, the mask detector, and non-masked face recognition. The improvement for non-masked face recognition and other approaches will be explored in the next phase.

2 METHODS

In this project, we built a face mask detection to first detect whether a person is wearing a mask, for those who don't we further built a face recognition to find out who they are. If the person is in our dataset, the system will provide his/her name, otherwise, an unknown tag will be given. [2]

2.1 Data processing

2.1.1 Dataset.

The dataset[8] we are using is from the Github: Real-World-Masked-Face-Dataset, which contains approximately 10,000 faces with a real mask for 1,000 people with a clear label. However, the distribution of masked and unmasked faces are highly imbalanced. To tackle this problem, we augmented the masked face with mirror flip, grayscale, and color(red, yellow, green) as shown in Figure 1 to increase the number of that 10 times more than before. After the augmentation, there are a total number of 150907 photos in the dataset, where 60% are face images and 40% are masked photos. We split it into training and test sets with 75% and 25%, respectively. The dataset is paired with masked and unmasked faces with a name label. For the face mask detector task, it is not necessarily for us to use the name label as we only care about masking or not. While for the bare face recognition task after the mask detection, we need to use those bare faces' name labels.

2.1.2 Image Resize.

While our model requires the same size for all images, we decided to resize them into 160 x 160, which is suitable for most cases. As shown in Figure 2, the first images is the original image. In the beginning, we just used a simple resize function and for those with different scales or shapes, we just stretch the images to

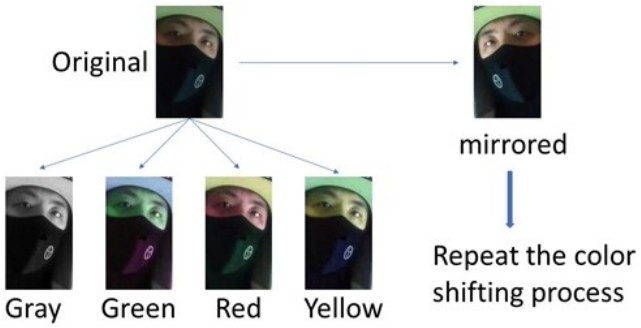

Figure 1: Data Augmentation

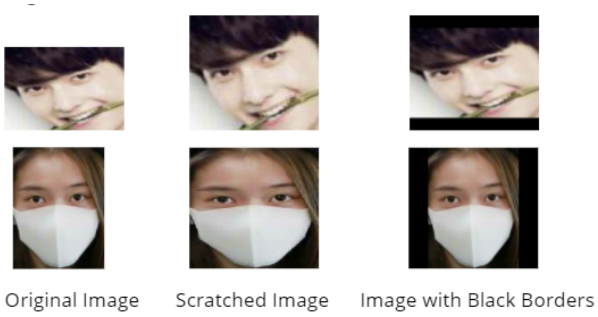

Figure 2: Resized Images Example

make it fit as shown in the second images in Figure 2. After the benchmark model, we found that the image distortion might result in a wrong label, so we tried another way. For those images with different scales, we first add black borders to make the aspect ratio to 1:1, and the black borders are evenly distributed on both sides. Then resize it into 160 * 160, examples are the third images in Figure 2.

2.1.3 Data Cleaning.

The dataset used for mask detection contains some small size images(width<100 or height <100) that are either of low quality or partial faces. The low data quality could be one possible reason for the low accuracy of the model. The image size distribution is shown in Figure 3. The two red lines show the size of images passed to the FaceNet system. We removed all images with width or height smaller than 100 for both training and test set.

We also manually went through the miss classified images from our benchmark model to fix the wrong labeled problem and remove low-quality images as well as images with faces covered by something else other than masks.

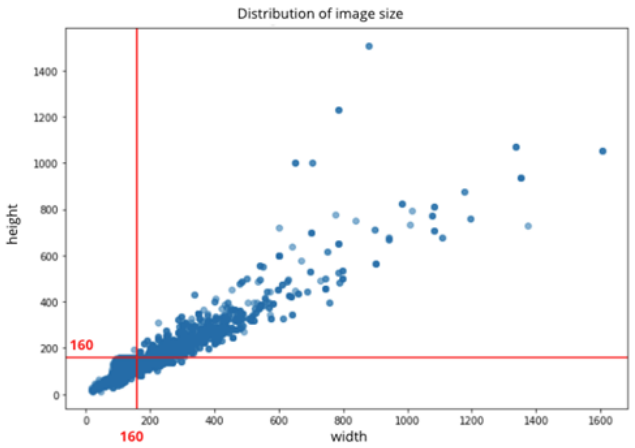

Figure 3: Distribution of Image Size

After cleaning the dataset, we have about 105K images in the training set where 67K images are bare faces and 38K are masked faces; 34.8K images in the test set where 22K images are bare faces and 12.6K are masked faces.

3 MODELING

3.1 Mask Detector

The flow chart of our whole process towards the mask detector model is shown in Figure 4. After the data preprocessing part as we stated in the previous part, we had well-organized and formatted images files. Since for the mask detector task, the images we had is already after face alignment, so we skipped it. But for the later task, the face recognition model, the face alignment is required for that dataset, in which we implemented MTCNN.

Our face mask detector model can be considered as two parts, the first part is a pre-trained FaceNet model which takes the images as input and can give us embeddings from output. Then we normalized the embeddings to get better performance on the classifier models. The second part is a classifier which will take the embeddings as input and predicted probabilities or labels as output.

3.1.1 Face Alignment with MTCNN.

Multi-task Cascaded Convolutional Neural Networks (MTCNN) is a framework developed as a solution for both face detection and face alignment with very good

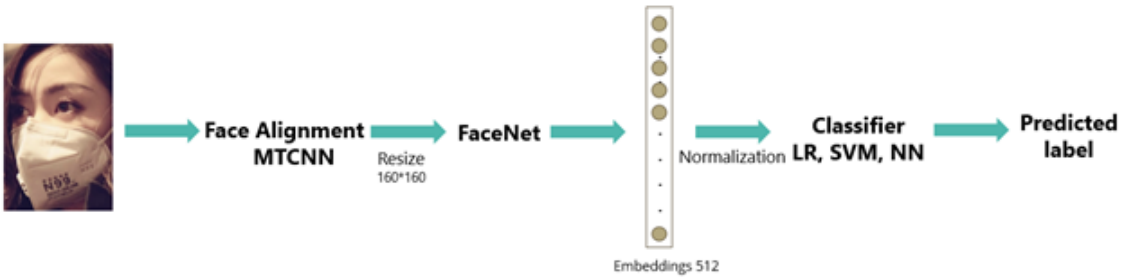

Figure 4: Flow Chart for Mask Detector Model

results. It uses cascaded CNN with three stages including the proposal network (P-Net), the refine network (R-Net), and the output network (O-Net).

We used the Python/Tensorflow implementation of MTCNN provided by David Sandberg from GitHub[6].

3.1.2 FaceNet.

For the main part, we decided to use the FaceNet model to accomplish our tasks, since this model is widely used in face recognition and performs very well.

The FaceNet[7] model is a system originally presented from research by Schroff, Kalenichenko, and Philbin. It can directly learn a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. In the FaceNet model, they also introduce triplet loss that is compatible with each other and allow for direct comparison between each other. For the performance, they achieve a very high accuracy of 99.63% on the widely used Labeled Faces in the Wild (LFW) dataset.

We used the pre-trained FaceNet model called 20180402-114759, which is trained on VGGFace2 dataset with Inception ResNet v1 architecture by Google.[4]

3.1.3 Classifier.

We first use the logistic regression to build our benchmark model and see how well our model performs. After that, we tried several kinds of classification models to train the embeddings, such as Logistic Regression with different regression methods (L1 and L2), Support Vector Machines with different kernels (linear, rbf, sigmoid, and polynomial), and Neural Networks.

3.2 Face Recognition

For the face recognition task, we again used FaceNet as it aims to deal with face recognition purposes, especially for a small training set. In the project, we adopted the MTCNN which is part of FaceNet to align the faces and then pass the cropped face to the NN model to have the embeddings. Finally, used the triplets loss to train the model. We explored 3 different numbers of training images for each class(person's name): 6, 35, 95, while the images to be tested are always 5. The training and test are operated on Nvidia GPU [3][1] so that the training time is only a couple of minutes. The face recognition is only applied to bare faces, assuming they are correctly classified by our mask detector in a real-world scenario.

4 RESULTS

4.1 Mask Detector

In our benchmark mask detector model, we explored 3 types of classifiers: logistic regression, SVM(Support Vector Machine), and NN(Neural Network). For logistic regression, we used L1 and L2 norm. The ROC of logistic regression with the L2 norm can be seen in Figure 5. The AUC of the model is 0.99, the accuracy is 0.95, and the best threshold is 0.38.

All the other models' performances are listed in Table 1. The NN model provides the best accuracy with an acceptable training speed(10mins). The logistic regression with the L1 norm also performs well with a relatively fast training speed. These two models are used in the next stage to find the best model with the balance of accuracy and training speed.

After cleaning the dataset, refining the resize methods, and normalizing the embeddings, we finally increased the accuracy to 99.6% on the test set. Table 2

Table 1: Accuracy for Different Classifiers

	Test Accuracy	Train Accuracy	Training Time
Logistic Regression(L2)	0.950	0.952	8.75s
Logistic Regression(L1)	0.951	0.953	98.7s
SVM linear	0.951	0.954	2hrs
SVM rbf	0.943	0.945	4hrs
SVM sigmoid	0.940	0.940	2hrs
SVM poly	0.601	0.606	2hrs
Neural Network	0.957	0.969	10mins

Table 2: Improvement in Accuracy and Speed

	Logistic Regression(L1)	Neural Network
Training Speed	10s	5mins
Data processing methods	Accuracy	
Data Cleaning	0.972	0.972
Normalization	0.965	0.989
Data Cleaning + Normalization	0.981	0.996

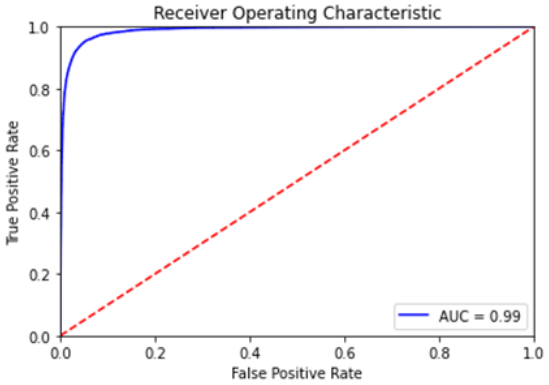

Figure 5: Receiver Operating Characteristic

shows the accuracy and training speed for each combination of the model and data process methods. As the training speed for the NN model is reasonable for any laptop with a commercial CPU, our final mask detector model is achieved by the NN classifier which will be part of the further face recognition task.

4.2 Face Recognition

In our face recognition model, we achieved 79.4% accuracy with 95 images in each class(person's name). In

Table 3: Accuracy of Face Recognition for Uncovered Face

	Face Recognition		
#images each class	6	35	95
accuracy	0.54	0.75	0.79

Table 3 we show the accuracy of each number of training images. As the volume of training images increased, the accuracy improved. However, compared with model performance on Labeled Faces in the Wild(LFW), 0.993, we still have plenty of room to improve. One promising improving direction is to increase dataset quality and diversity.

5 DISCUSSION

We analyzed the bias and variance trade-off in as well as the misclassification problem in our face mask detector. Base on the analysis we improved our detector accuracy from 95% to 99.6%, which is close enough to the Bayes error in the task. We discussed the possible

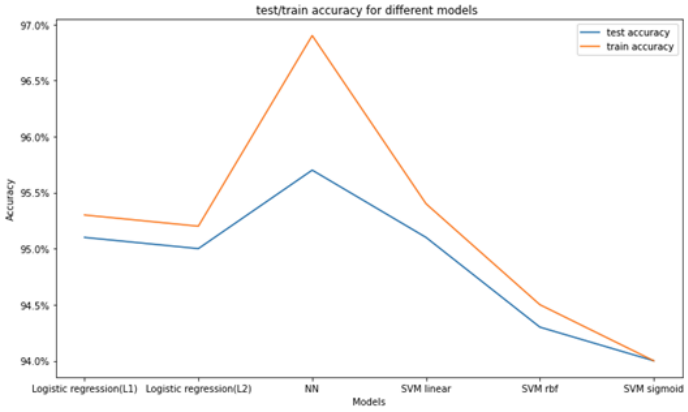

Figure 6: Accuracy for Different Models

component we could add to improve the face recognition model and implement the whole system in the real world.

5.1 Bias and Variance Analysis

To increase the performance of the mask detector model, we analyzed the accuracy of the benchmark model for both training and test set. For models with different classifiers, the accuracies(shown in Figure 6) on training and test set are quite similar, except the NN model which has a gap of 1.5%. However, if we compare the gap in the NN model with the Bayes error rate which should be the human level error rate(0%) in this task, there still approximately 3% for the model to increase. Since the gap between the training set error rate and the Bayes error rate is larger than that between the training and test set, the direction for the model to improve should be reducing the bias instead of variance.

5.2 Misclassification analysis

We manually went through the miss classified images and found 7 misclassification types: partial face, wrong labeled, face covered by other objects, low quality, artificial masks, cartoon effects, small size(<100). The examples of these images can be seen in Figure 7.

For the partial face, face covered by other objects, low quality, artificial masks, cartoon effects errors, we removed those images from the test set manually. We also relabeled the wrong labeled images. For small images, as they are either wrong cropped or low-quality images

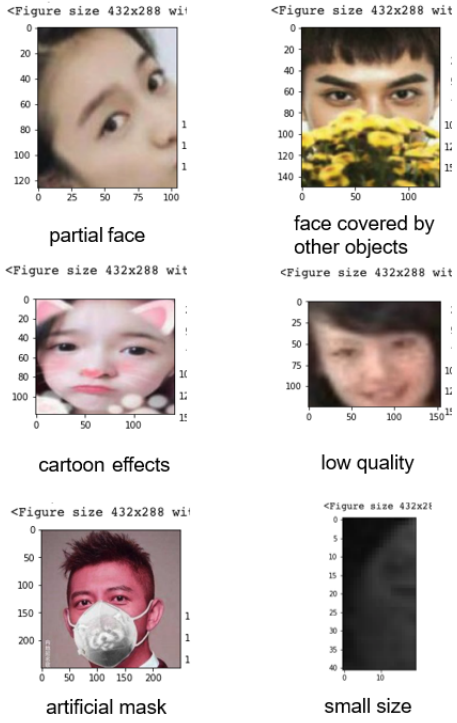

Figure 7: Misclassification Types

and composed 30% of the false-negative misclassified images, the too-small size problem may be a possible reason for miss classification. Thus, we removed them from both the training and test set. After the misclassification analysis, we improved the quality of our dataset through various methods which assisted further model training.

5.3 Normalization

The normalization of the embeddings improved the training time as well as the accuracy as shown in Table 1 and Table 2. Normalization scaled the input value so as to avoid the vanishing problem in activation functions(tanh, Relu, etc), leading to the decrease of training time. Normalization also has a positive impact on model accuracy, which may relate to the gradient descent. Because normalization ensures (a) that there are both positive and negative values used as inputs for the next layer which makes learning more flexible and (b) that the network's learning regards all input features to a similar extent. Our project again verifies the importance of normalization in machine learning.

5.4 Face Alignment in the Future

To improve the project in the future and implemented it in the real world, we need to add a face alignment component in the system[5] which is also one goal in the next stage of the project. The dataset we used is aligned faces which will not be the case in the real world. Additionally, most commonly used face alignment algorithms are aiming to detect a bare face, as we will recognize faces with masks, we need our own approach to align face with and without mask at the same time. With a better alignment component, we can crawl the web and create a high quality and large diversity dataset, which could potentially improve the face recognition model and serve all the purposes of the project.

6 STATEMENT OF CONTRIBUTIONS

Zishen Li: data cleaning, classification, misclassification analysis

Junmei Luo: data preprocessing, mask detector model

Tony Yukuang Zhang: data preprocessing, face recognition model

REFERENCES

[1] [n. d.]. PeopleNet Model. https://ngc.nvidia.com/catalog/models/nvidia:tlt_peoplenet

[2] N. Damer, J. H. Grebe, C. Chen, F. Boutros, F. Kirchbuchner, and A. Kuijper. 2020. The Effect of Wearing a Mask on Face Recognition Performance: an Exploratory Study. In *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*. 1–6.

[3] Amey Kulkarni. [n. d.]. Implementing a Real-time, AI-Based, Face Mask Detector Application for COVID-19. https://bit.ly/3k8P1BC

[4] Athul P. [n. d.]. Building a face recognition system with FaceNet. https://medium.com/@athul929/building-a-facial-recognition-system-with-facenet-b9c249c2388a

[5] Adrian Rosebrock. [n. d.]. OpenCV Face Recognition. https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

[6] David Sandberg. [n. d.]. Facenet Implementation. https://github.com/davidsandberg/facenet

[7] F. Schroff, D. Kalenichenko, and J. Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 815–823. https://doi.org/10.1109/CVPR.2015.7298682

[8] X-zhangyang. [n. d.]. Real World Masked Face Dataset. https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset