Masked Face Dataset Generation and Masked Face Recognition

Xuying Ning, Rui Cai

March 26, 2023

Abstract

2D convolution is a notion from Digital Signal Processing and has been used in Convolution Neural Networks. In our project, we are trying to implement Convolutional Neural Networks to deal with challenging face recognition problems. In the post-pandemic era, wearing face masks has posed great challenge to the ordinary face recognition. We selected 50 identities with 1702 images from Labelled Faces in the Wild (LFW) Dataset, and simulated face masks through key point detection. We compared three different convolutional neural network's performance on our masked face recognition problem. Instead of directly using pre-trained models, we fine-tuned the model on our new dataset and use the last linear layer to do the classification directly, furthermore, we proposed using data augmentation strategy to further increase the test accuracy, and fine-tuned a new networks beyond the former study, one of the most state-of-art networks, Inception ResNet v1. The best test accuracy on 50 identity MFR has achieved 95%.

1 Introduction

For our final project, we decide to contribute to Masked Face Recognition (MFR) problem. Deep CNN based neural networks has shown great success on normal face recognition. Masked face recognition, on the other hand, aims to recognize a face with a mask basing on the eyes and the forehead regions. In this project, we curated a new dataset by simulating face masks on images of LFW dataset, cropped the upper half of the face image, did the data augmentation, and compared 4 models of solving masked face recognition, including Eigenface and 3 different deep neural networks, Inception ResNet(v1), ResNet50, and VGG16. To help implement the real world applications, we also proposed *Exponential Margin*, to determine whether a new image is one of the recognizing targets not.

1.1 Related Works

In the former study, there are three methods proposed to solve occlusion, matching approach, restoration approach, and occlusion removal approach. Since occlusion removal approach using together with CNN has shown advance in recent years, our project only focused on occlusion removal approach.

Hariri[1] proposed to a way of removing occlusions, by sampling the masked face image into 100 regions of the same size and use cropping filters to crop the bottom half 50 regions and leave the upper 50 region of interest. He used the pre-trained model directly to extract features of input crpped images, and use bag of features to do the classification. The datasets he has worked on are RMFRD and SMFRD. According to his reults, using VGG16 pre-trained on ImageNet as a feature extrator, achieved the best performance 91.3% on RMFRD dataset, while ResNet50 pretrained on ImageNet achived the best performance on SMFRD dataset,88.9%.

1.2 Motivation and Contribution of Our Work

In the real application, the amount of samples we can get of each target would be very limited, and the camera setting and the environment can varies a lot, most importantly, randomly taken images wouldn't be center around the face, the person's face most of the time only take up a very small proportion of the image. Those facts make the two mostly used MFR datasets, RMFRD and SMFRD, inappropriate. Both datasets are centered on faces, and only left little margin outside the face. Furthermore, for RMFRD, there are only 5000 images are real images with face masks of more than 500 identities, instead, there 90000 images without face masks. This makes the dataset appropriate for methods training with both masked and unmasked images, such as matching approach. For SMFRD, the shortage of this dataset is that the environment of the image setting is not varying as much as we need. This inspired us to simulate a new dataset which can be more adaptable to solve real world problem.

And for the masked face recognition methods, what was proposed in the previous study, directly using the pre-trained model on ImageNet to extract features is not satisfying, since the domain difference is too large. We think the model can do better if we do the fine-tuning of our datasets. Furthermore, Inception ResNet has shown better performance than ResNet on normal face recognition, thus, we want to compare this model's performance to the previously dominant models' on our problem.

In our project, our contribution to previous studies are listed as follows:

1 We constructed a data set with large environmental changes and small number of targets and samples, which can be used when computing resources are insufficient, or when evaluating small-sample learning models.

2 We used data augmentation strategy which hasn't been used in the previous study to achieve a higher test accuracy.

3 We fine-tuned three different model on our dataset to generate better results and do the comparison with traditional face-recognition methods like Eigenface.

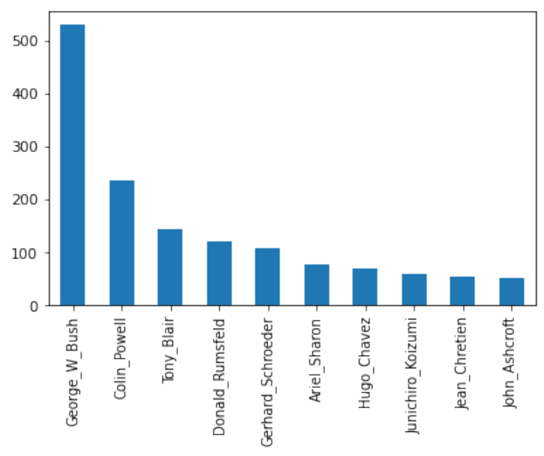
4 We proposed Exponential Margin to determined if the image corresponds to targets that we are trying to recognize.

2 Technical Approach 1: Dataset Generation

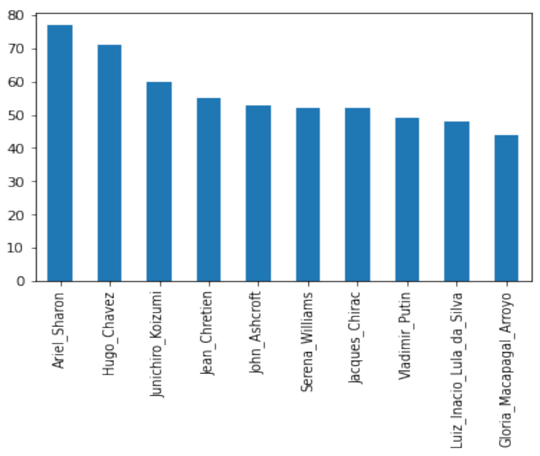
2.1 Filtered Labelled Faces in the Wild (LFW) Dataset

LFW dataset has been a dominant model testing dataset in face recognition. This database was created and maintained by researchers at the University of Massachusetts, Amherst (specific references are in Acknowledgments section). 13,233 images of 5,749 people were detected and centered by the Viola Jones face detector and collected from the web. We used deep-funneled version of the dataset. The images in the dataset has grealy varied background, lighting and fae orientations. It's more challenging than RMFRD and SMFRD.

In the EDA of the dataset, we find that that LFW dataset has large variance on the sample sizes of each identity. There are only 29% of the identities have 2 or more images, which means most of the identities in the dataset doesn't have enough images to be split into train, test and validation set. Also, the largest number of pictures one people can have, is more than 500 (George HW Bush), which is 500 times more than 71% of the people in the dataset. To debias the dataset, we randomly selected 50 identities who has the number of pictures in the range of (20,80). After selection, the average number of pictrues of one person is 33. In the picture below, we can see the filtered dataset has been largely debiased.



(a) The Original LFW Dataset



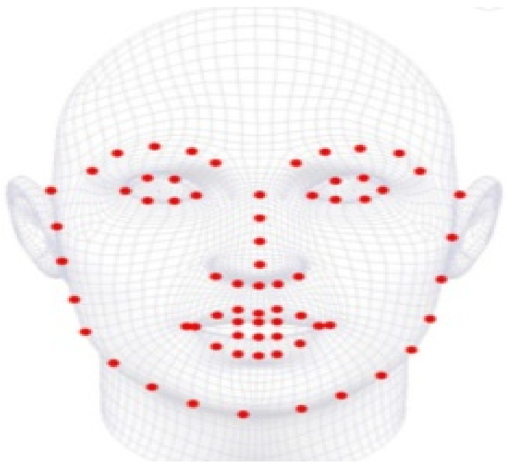
(b) LFW Dataset After Filtering

Figure 1: 10 Identities With the Most Pictures and the Number of Corresponding Pictures

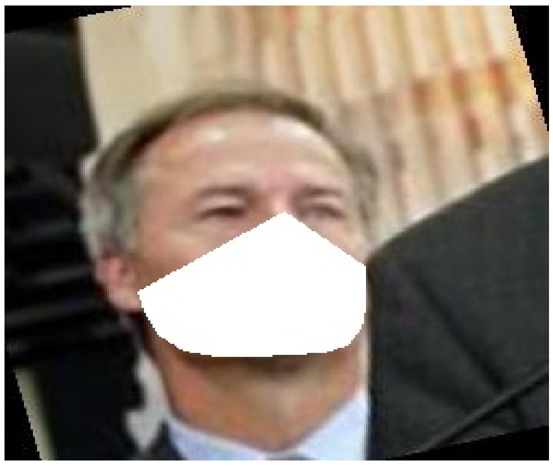
Since the number of recognizing target is small (50), and the number of samples of each target is also limited, this dataset is a good choice for researchers with limited computing resources (like us) or wants to investigate the small sample machine learning.

2.2 Generation of Masked Face Dataset using Dlib

Due to the lack of datasets with faces wearing masks, we designed a method to put masks onto faces in the selected database of LFW dataset. It is applying dlib package to locate the 68 keypoints on one's face as shown in the picture, and we manually selected 14 points(3–15, 29). Fortunately, OpenCV is able to form convex shapes , just as mask shapes, so we applied openCV to connect the chosen points and generate the mask shape polygon to replace the real mask. We as well have tried to use real N95 masks, but it had bad performance on tilted faces, such that the mask could not match the location of the nose and mouth. Consider the common colors of masks, we chose black(R:0,G:0,B:0), white(R:255,G:255,B:255) and sky blue(R:250,G:206,B:135) for generalization. This method also has done well on side faces. Below figure shows the performance using dlib for mask generation.


(a) 68 keypoints of one's face

(b) An example of masked front face


(c) An example of masked face with elevation angle

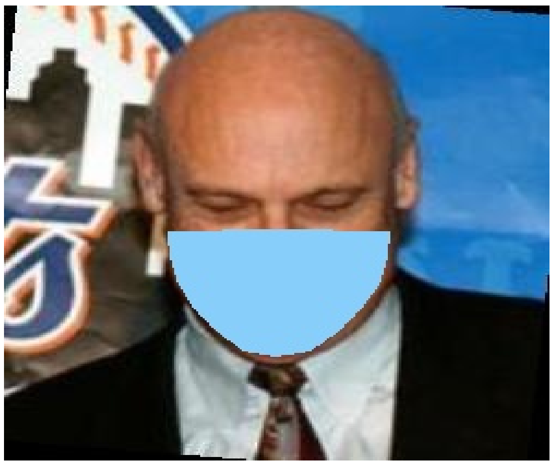
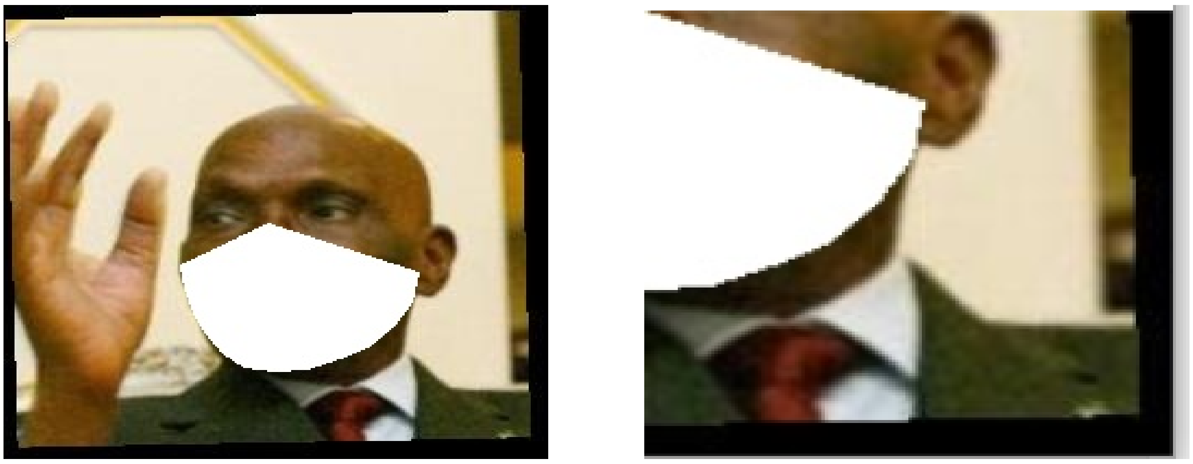

(d) An example of masked face with depression angle

Figure 2: Method and Examples of Generating Masked Dataset

2.3 Masked Face Detection

Multi-task Cascaded Convolutional Networks (MTCNN) and dlib package are two most popular face detectors people are using to crop faces from a large image. We tried face detection after simulating face masks on LFW Dataset, it turns out that neither one of the two methods can work well. Since the face masks has covered two face detection key points, the nose and the mouth, MTCNN are often unable to detect faces from the image, while dlib's face detector are able to work sometimes, but the cropping results are very bad. Dlib uses pretrained masked face detection net to locate the faces, and we tried to use the dlib detected and cropped face image as input of Inception ResNet, the best test accuracy was only 41% since the horrible return of face detection. Some bad return of face detection has been shown below.


(a) An Example of Simulated Masked Face Picture

(b) Bad Detection Results by Dlib

Figure 3: Bad Face Detection Results on Masked Simulated Dataset

Since there are no existing good face detector to detect masked faces, we have to give up some of the property of building the real world application. Instead, we used MTCNN in facenet-pytorch package to detect face in the filtered LFW dataset first and then simulate face masks on the well cropped face image. The reason why we chose MTCNN in facenet-pytorch is that, in comparison, this detection methods is the fastest among all with great performance on various environment.[2] In our practice, we left 15 pixel margin around the face, in order the jawlines can be detected and the face masks can be well simulated on the original image. The following picture shows the face cropping result by using MTCNN.

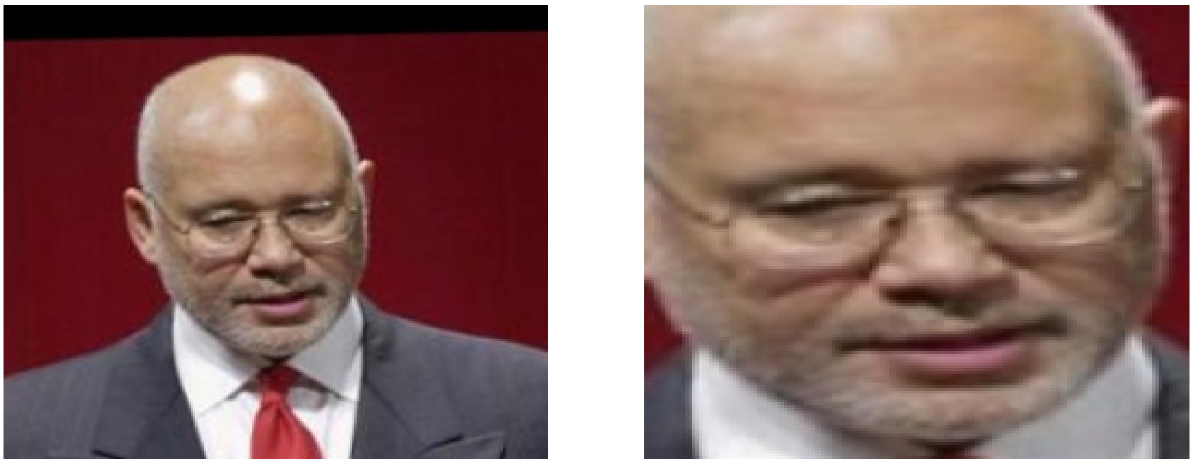

(a) Original Picture of an Example in LFW Dataste (b) MTCNN Cropped Version

Figure 4: MTCNN Face Detection and Cropping

2.4 Cropping faces in masked dataset

For the facial recognition model of people using a face mask, only part of the upper part of the face is extracted. This is due to the fact that the covered part of the face has little information for the recognition model, and the mask the person is using always varies and does effect the training process(as proved in the experiment). We finally chose upper 1/2 part of the face for the final training. So the overall cropping pipeline could be :

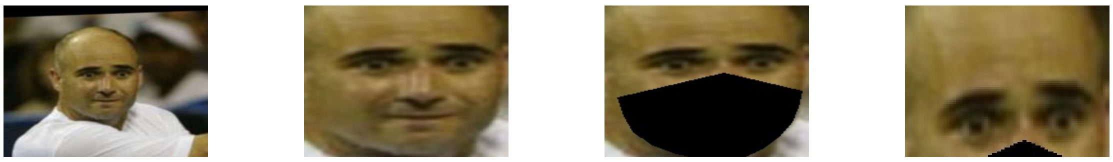

(a) First Step (b) Second Step (c) Third Step (d) Final Step

Figure 5: The Pipeline of Wearing Mask and Cropping Face

2.5 Train Test, and Validation Split

For each target, we randomly split pictures of that target into train, test and validation set, according to the ratio (0.8,0.1,0.1). So that we can have unbiased train, validation and test sets still roughly unbiased on each target. The numbers of images in train, test and validation set are 1340, 166 and 196 respectively.

2.6 Dataset Augmentation

Some of the cropped upper half face images (resized and normalized version) are shown below. As one of our dataset's property, the orientation of the face are not always facing the front. Many images of our dataset capture the person's side face.

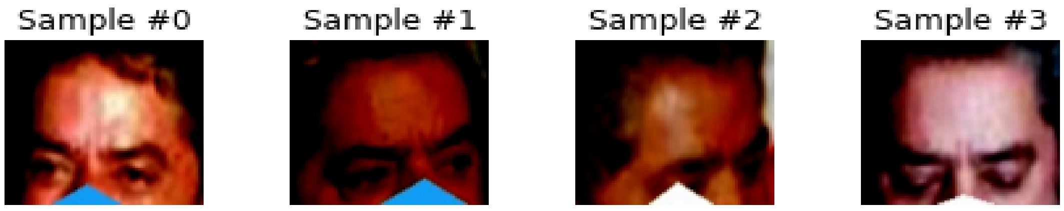


Figure 6: 4 Examples of Cropped Upper Half Face Image

Since the number of training samples for each target is very limited in our dataset, we need to decrease the possibility of the network learning unnecessary features, like face orientations.

We proposed using random rotation to augment our dataset, so that we can simulate a person having sample pictures facing the different angles. We used RandomRotate in pytorch with degree in the range of (-20,20) degree, to generate 3 more random rotated version of pictures for each training samples. The new augmented dataset contains not only the original samples, but also the random rotated versions of samples. The new dataset is now four times the size of the original dataset. The following figure shows some random rotated version of an original sample.

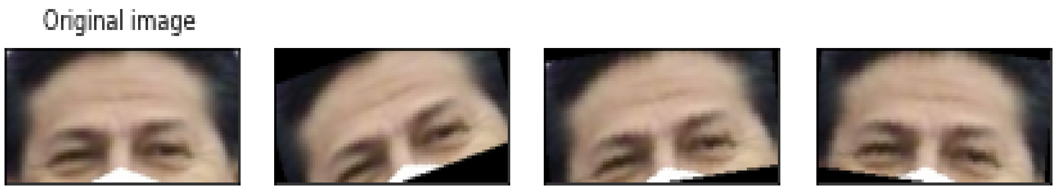


Figure 7: Data Augmentation Through Random Rotation

Besides the random rotation of pictures, we also used jitterColor to increase the brightness of faces since the misclassified samples are usually dim and ambiguous, but the result turned out to be relatively similar. We compared test accuracy of fine-tuned Inception ResNet(v1)s' results on both un-augmented dataset and augmented dataset. We used the same hyper parameters on two cases for better comparison (except the max training epochs since the one with data augmentation converged with less epochs). The test accuracy after data augmentation has been increased by 3%, which indicates our data augmentation method has made the model more generalizable. The hyper paramters and the test accuracy are listed in table 1 and 2 below.

Hyper Parameters	
Batch Size	8,8,8
Shuffle	True
Optimizer	Adam
Learning Rate	0.001
Max Epoch	50,35

Table 1: Hyper Parameters of Inception ResNet(v1)

Test Accuracy Comparison	
Without Data Augmentation	After Data Augmentation
87%	90%

Table 2: Test Accuracy Comparison

3 Technical Approach 2: Idea of 2D Convolution in Convolutional Neural Networks

We learned 1D convolution in the lecture, which is the formula below:

$$y[n] = x * k = \sum_{m=-\infty}^{+\infty} x[m]k[n-m]$$

where, we call $k[n]$ kernel in CNN. Since the convolution is commutative, we usually sum over kernels instead of the picture, since the kernel is usually smaller. Also, since the kernel's non zero entries are usually defined with positive indexes, therefore, in practice, we don't flip in the convolution. The resulted 2D convolution is as below:

$$y[i,j] = K * X = \sum_{m}\sum_{n} X(i+m, j+n)K(m,n)$$

where X is the 2D image, and K is the kernel, the range of m and n is defined on the non zero entries of the kernel. If the image is 3 channel input, the 2D convolution is simply done with each channel and add them up.

As researcher's have found that certain kernels can have certain effects on the pictures. For example, some kernels can sharpen the image, some can Gaussian blur the image and some can do edge detection. The figure 8 is from Wikipedia and it illustrate examples of effects of certain kernels.

The idea of Convolutional Neural Networks is using backward propogation to learn layers of different kernels to extract certain features that can used to classify images.

3.1 Illustration of VGG16 and Residual Connection

VGG16 was proposed in 2014 by researchers in Oxford[4], and it's a very elegant and deep CNN and has very good performance on ImageNet benchmark. There are

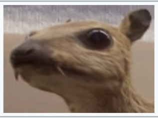
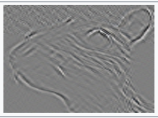
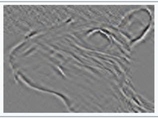
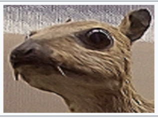
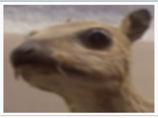
Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Figure 8: Kernel's Effects

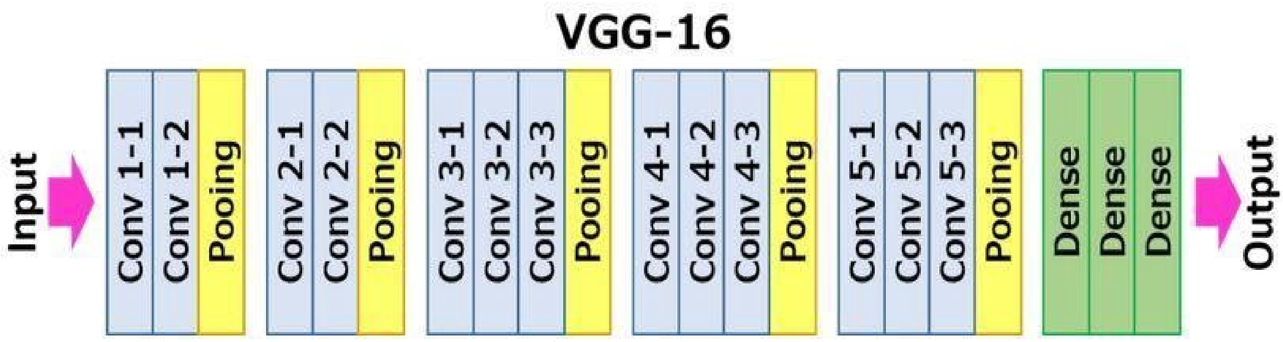


Figure 9: VGG16's Architecture

16 layers of weights to learn. The architecture is shown below. VGG16 is not using largely varied hyperparameters. Instead, it is focusing on kernal size of (3*3) and Max Pooling size of (2*2).

The inputted image size of VGG16 is (224, 224, 3). The Conv1-1 and Conv 1-2 layers are convolutional layers with kernel size (3*3) and 64 kernels in each layer. Pooling Layer afterward is Max Pooling with stride (2*2). The second set of convolutional layers 2-1 and 2-2 has 128 kernels in each. The following third set has 256 kernels in each. The fourth and fifth sets both have 512 kernels in each. The dense layers are just flatten the (7*7*512) feature map and input it into the fully connected layers.

VGG16 is a representative of deep-net. However, with the depth of the neural networks increases, small gradients are more likely to die out at the end of the networks. To tackle the problem, ResNet[5] was proposed. The invention of residual connection provides huge boost of solving gradient vanishing. A residual connection connects the output of one earlier convolutional layer to the input of another future convolutional layer several layers later. The input of the mainstream model and the input of the convolutional prior are combined using a simple sum. The figure below shows the mechanism of residual connection.

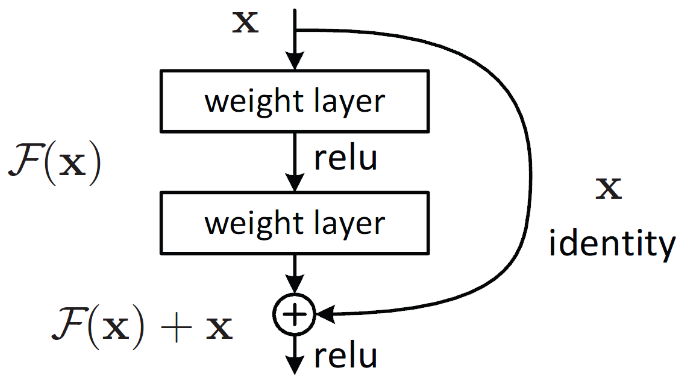

x
weight layer
relu
weight layer
relu
$\mathcal{F}(\mathbf{x})$
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
x
identity

Figure 10: Residual Connection

In our project, we intend to compare the performance of different CNN baese models on our problem. We find that residual connection does improved the performance a lot. Next, we will talk about how to did the transfer learning using VGG16, ResNet50 and Inception ResNet v1.

4 Technical Approach 3: Idea of Principal Component Analysis in Eigenfaces

The traditional face recognition techniques has a main difference with the modern deep learning techniques. It is that we need to manually extract the features in human faces(embed the face), while in deep learning the feature extraction is learnable. In a signal processing perspective, in traditional techniques we manually design methods to do image processing, while in modern techniques the machine is learning to do image processing by itself.

Eigenfaces is a traditional face recognition technique. It mainly applies PCA on the image processing of faces. Typically, the final goal of PCA is dimensionality reduction.

In face recognition, we regard each image as a vector, and first flatten the N*M image into a vector of length N*M. Then we conbine the face vectors into a huge matrix X with each face vector as its rows. We calculate the average of all these face vectors in the train samples μ.

$$\mu = \frac{1}{m}\Sigma_{i=1}^{m} x_i$$

Then we apply PCA on the covariance matrix of X to get the principal component matrix W, with each column as the sequential principal component of the faces. Then we choose the first k component to see the reduction of dimensionality and project the face vector onto the subspace of PCA using formula:

$$y = W_k^T(x - \mu)$$

where W_k represents the first k columns of W. Actually we could reshape the vector into N*M matrix to show the picture with pixels, which are demonstrated in the latter part. We then use the reformulation of PCA to project the test samples into the PCA subspace:

$$x = Wy + \mu$$

In the test procedure, we first flatten the faces image and subtract with the average vector of training samples. Then we project the test face vector into the PCA subspace, then we calculate the defined distance between the test embedding(the projected vector) and other labels to determine whose face it is using a set threshold to decide.

5 Experiment: Fine-tune Pre-trained Models on MFR Problem

5.1 Model Selection

According to the previous study, the pre-trained ResNet50 and VGG16 on ImageNet have shown great performance on the feature extraction of masked face images (the bottom half of the faces has been cropped). We chopped off the ouput layer of the two model, freezed the other layers, and only trained on the last prediction layer, the result is not promising, only around 60% accuracy, which is due to the large domain difference between ImageNet and our MFR dataset. Therefore, we propose using fine-tuning to improve the performance of the pre-trained model on our problem by fine-tuning. Instead of using models to spit out embeddings and train another MLP to do the classication, we directly add the classification layer with 50 units at the end of the network to do the prediction.

Beside the two models mentioned in the previous study, we also fine-tuned a more state-of-art CNN based model, Inception ResNet(v1) on our dataset. Inception ResNet v1 is a hybrid model by adding residual connection into the inception v3 modules, which has generate state-of-art performance on ImageNet benchmark, 3.08 percent top 5 error. Although Inception ResNet v2 is more advanced than the v1, but it requires more computational resources. Therefore, we chose to add Inception ResNet v1 into our model comparison. The Inception ResNet v1's architectures is shown in the figure below.[3]

Three models were pre-trained on different datasets, we are using VGG16 and ResNet50 pre-trained on the ImageNet, while the Inception ResNet v1 was pre-trained on VGGFace2 dataset proposed in 2017. VGGFace2 is a large and well

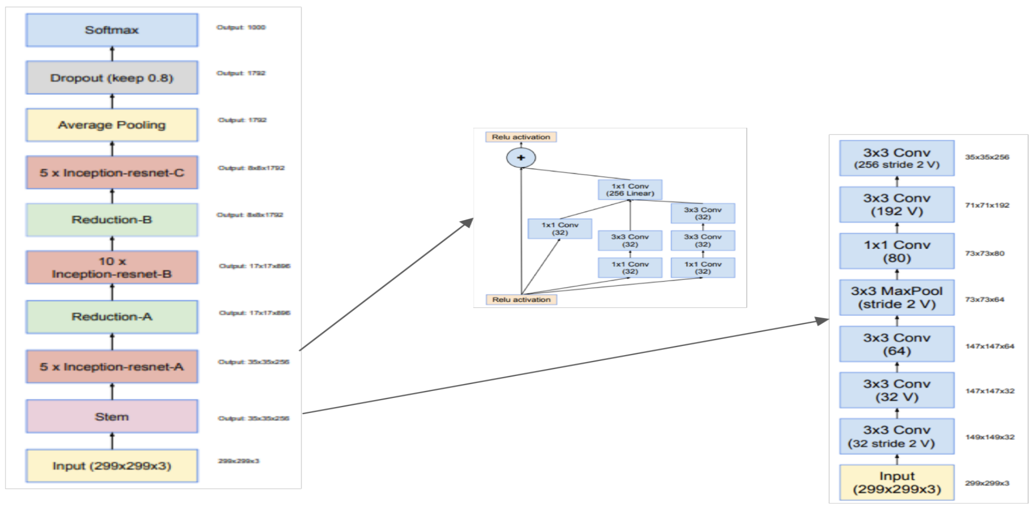


Figure 11: Inception ResNet v1

curated dataset of human's facial images, and now is been widely used in the face recognition area. Since the domain difference between VGGFace2 and our MFR dataset is much smaller than that of ImageNet and our dataset, we expect the converging process of fine-tuning the pre-trained Inception ResNet v1 faster than VGG16 and ResNet50. The tabel below shows details of the pre-trained models.

	Inception ResNet v1	ResNet50	VGG16
Pre-trained Dataset	VGGFace2	ImageNet	ImageNet
Parameters	23508274	23610482	15.184M
Layers(include pooling)	27	50	16
Input Size	(3,160,160)	(3,224,224)	(3,224,224)

Table 3: Model Information

5.2 Model Setup

The programming language used here is Python. For optimal operation, a high processing equipment (GPU) is needed. We chose to work with AutoDl's virtual machine with RTX 3090 with 24GB video memory.

5.3 Training Preparation

Since we compared the results of using data augmentation or not on Inception ResNet v1, and it is obvious that data augmentation leads to better results. Therefore, we are going to do the following training on our augmented dataset.

We preprocessed our dataset through the following three transformation:

1 Convert the data type to float 32

2 Convert numpy array to tensor

3 Resize the image to the corresponding input size of each model

4 Normalize the image by channels.

We rewrite the Dataset in torch, and use dataloader to split the train, test and validation set to batches with batch size equals to 8 and allows shuffling.

5.4 Fine-tuning Procedures

Basically, all three models shared the similar fine-tuning procedures as follows.

1 We download the pre-trained models and their corresponding weights through torchvision package or facenet-pytorch packages.

2 We chop off the last layer of the original model, and change it to the linear layer with last layer's output size as input size and 50 (number of classes in our problem) as output size, and let the weights in the last linear layer be randomly initialized. For the Loss function, we are using the Cross Entropy Loss.

3 Freeze the weights except the last fc layer and train on the last layer first. Set the maximum trianing epoch to be 2, let the weights of the last layer converges to a decent level.

4 Unfreeze all the layers and train for maximum 50 epochs. After each training epoch, turn off the drop off and batch norm layers, evaluate the model on the validation set. Always save the model corresponds to the smallest validation loss.

5 Reload the saved bets models and test on the test set. The prediction of the test image corresponds to the unit in the last output layer which has the largest value.

6 Calculate the test accuracy, save the confusion matrix, record the "score" of each prediction.

It is worth mentioning that, for the VGG16, with the Drop Out layer with pobability set as 0.5, the model is not converging in our practice, we changed drop out layers in VGG16 to Batch Norm layers with default parameters, and change the second linear layer's output unit as 1024 for network surgery. And for Inception ResNet, we didn't freeze the last layer first but directly trained on all parameters which also generated a good results.

5.5 Results Discussion

All models successfully converged in the end, we recorded the train and validation loss. As an example, the recorded of loss from tensorboard of our Inception ResNet v1 is shown below. The pink line is the training loss and the blue line is the validation loss.

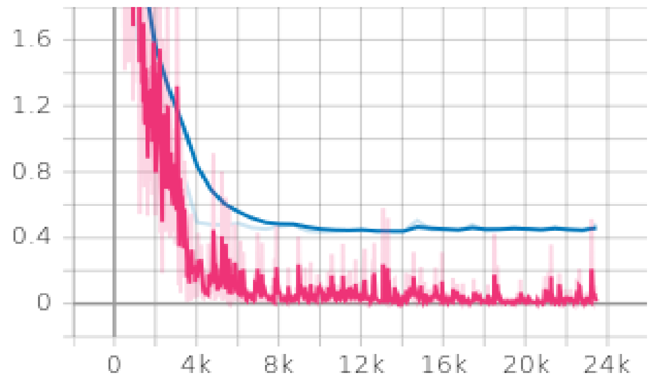


Figure 12: Training and Validation Loss of Inception ResNet v1

We tried different sets of hyper parameters, and the following table 4 shows the corresponding hyper parameters of each model's fine-tuning.

	Inception ResNet v1	ResNet50	VGG16
Batch Size	8,8,8	8,8,8	8,8,8
Optimizer	Adam	Adam	Adam
Learning Rate	0.001	0.001	0.001
Best Performing Epoch	35	50	10

Table 4: Hyper Parameters

In the training process, we used validation set to saved the model with minimum validation loss, but when we compare the test accuracy of the saved model and the last epoch's model(didn't minimized validation loss), the test accuracy of the last epoch is higher than the saved model. This could be resulted from the number of the validation set samples is too small(166), or because of our model is still during potential under-fitting. Thus, we listed the test accuracy of both cases in the table below.

Test Accuracy Comparison				
	Eigenface(Traditional)	Inception ResNet v1	ResNet50	VGG16
Best Accuracy	25%	90%	95%	70%
Early Stopped Accuracy	Not available	89%	92%	66%

Table 5: Test Accuracy Comparison

As what is shown in the table 5, both Inception ResNet and ResNet50 has shown great performance on our MFR problem. But what unexpected is that ResNet50 shows higher accuracy compared with a more state of art Inception ResNet v1. This might be resulted from the depth difference of two model, ResNet50 is deeper and the architecture is more symmetric. what is obvious is that the residual connection has greatly improved the performance, which could be resulted from solving the gradient vanishing problem when try to distinguish many very similar images of different identities.

5.6 Comparing with Traditional Method

We also put the test accuracy of a traditional face recognition method–Eigenface. We used histogram equalization and put the grayscale upper half face image into training, and the result turns out to be relatively bad as we are not expected, which is 25% accuracy as shown in the above table. We tried to analysis the result, since if using the whole face for training(without masks), the accuracy would be reach up to 75%. This may due to the fact that the upper half face contains not enough face information for eigenface model to classify the face related embeddings in the eigenspace, which further indicates the "magic" of deep learning. Below shows the mean and eigenfaces of the persons of the selected dataset.

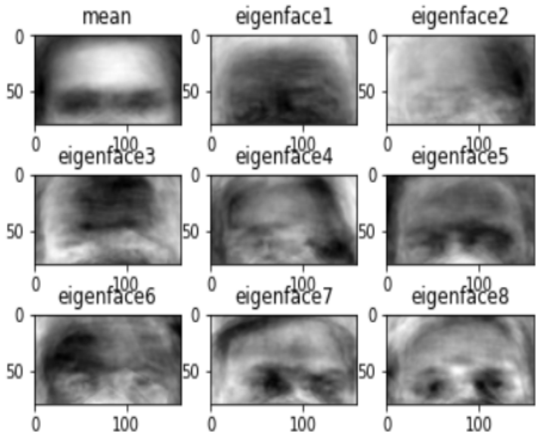


Figure 13: PCA Processed Eigenfaces of the dataset

5.7 Wrong Prediction Check

We checked the wrong prediction on each of the model. The following figure shows some of our wrongly predicted input images of Inception ResNet v1.

We can find that most of the wrongly predicted samples are sided faces, or the person in the image is doing some frowning which twisted the facial features. It is not a surprise, since the side face left less features for the recognition and both cases will change the normal placing of people's facial features, especially eyebrows. The reason why we humans can distinguish people around us even if they sometimes have hideous expressions is because we are not observing static pictures, but can acquire more features in the movement of subjects. This also enlightens us whether

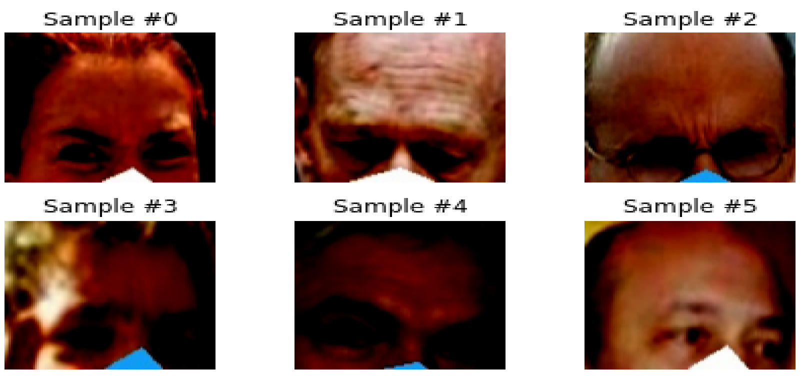


Figure 14: Bad Prediction Samples

deep learning may be able to produce better recognition results on a small video stream.

6 Determine A New Image Not in Targets

In most of the real application, the system not only needs to recognize the target subjects, but also needs to tell whether a person is not any one of the recognizing targets.

Intuitively, if a input new image is not from any of our recognizing target, the confidence of the prediction won't be as high as inputting a target's image. Therefore, we defined a "confidence score" to examine the confidence of the prediction.

$$CS = \frac{max\left(exp\left\{F\left(x\right)\right\}\right)}{\sum\left(exp\left\{F\left(x\right)\right\}\right)/n} \tag{1}$$

where F represent our neural network, and n is the number of output units,50. In plain words, we are calculate the maximum number of the exponential ouput over the average of exponential ouputs to determine the confidence of the prediction.

We randonly selected 195 images of the non-target pictures in LFW dataset and use the same preprocessing methods to deal with them. After inputting our network(ResNet50), We calculate the percentile of CS for both testset and the non-target set(195 images), and the result is shown in the table below.

Target(196)	Non-Target(195)
15 Percentile	85 Percentile
49.61	46.18

Table 6: Confidence Score Comparison

We removed the 15% extreme values, and compared the 85 percentile of targets' confidence score and 15 percentile of the confidence scores of non-targets. As we can see, there is a 3.42 margin between the confidence scores of target and non-target. We call the margin exponential margin. In the real application, we will calculate the confidence score of the inputted image, if the the confidence score is lower than 49.61, we will regard the inputed image as from a stranger(not any one of our recognition subjects).

7 Conclusion and Future Works

As we first filtered the faces to optimize the LFW dataset into a cohort of 50 people and then completed the generation and cropping of masked dataset within the whole pre-processing procedure, we successfully trained a face-recognition model using fine-tuned ResNet models and data argumentation techniques. The face of someone found in the database can be 95% correctly classified to provide the name tag and probability of success. It is notable that in the model comparison part, the deep learning neuron networks are obviously better than tradition models on the masked face-recognition topic. Compared to random guess of one category with possibility 1/50, Eigenface increases this number to 1/4 while DNN achieves 19/20. Likewise, the model performance is positively correlated with the complexity of the structure of network. While applying VGG16, the model could only reach 70% accuracy. As we tried the models with ResNet series which becomes deeper, the performance greatly improved.

Our data argumentation method did improve the performance of recognition, and we created a standard for recognizing strangers, which are not included in our dataset.

In the first precessing part, we first cropped the face and then created the masked dataset, which is not generalized for masked face detection since the right sequence of the system is recognition after detection. Therefore, as a future work, we need to train a detection model to detect the masked and unmasked faces in one picture for the purpose of building a face recognition system with surroundings.

It is also possible to notice that we only chose the 50 categories of faces in the database. When we increase the number of categories, the performance actually became worse. So our model could be regraded as a system of recognize a person's acquaintance, but not suitable for huge database recognition. As a recent research tendency, for big dataset, customizing a new loss function is necessary rather than just using CrossEntropyLoss.

Our code is uploaded at: `https://github.com/luisrui/Seeing-AI-system.git`. The extra packages we've used in python include torch, facenet-pytorch, pandas, numpy, torchvision, cv2, shutil, and dlib.

Our cropped and masked-face dataset, augmented training set, and their corresponding label csv files have been uploaded to Google Drive, the link to the Google Drive: `https://drive.google.com/drive/folders/1GvQS-xWG699w6QhPKdgc0-7fSdQWxBCo?usp=share_link`.

The original LFW can be downloaded from Kaggle.

Our saved models can be find on Google Cloud: `https://drive.google.com/drive/folders/1pwvN32jLg7875TgLzMftVG3_WfG_y95o?usp=share_link`. For each of the model's details can be found in github's README file.

8 References

1 Hariri, W. Efficient masked face recognition method during the COVID-19 pandemic. SIViP 16, 605–612 (2022). https://doi.org/10.1007/s11760-021-02050-w

2 Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. arXiv. https://doi.org/10.1109/LSP.2016.2603342

3 Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv. https://doi.org/10.48550/arXiv.1602.07261

4 K. Karen Simonyan and A. Zisserman, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION," in ICLR, Apr. 2015.

5 He, K., Zhang, X., Ren, S., Sun, J. (2016, June). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)