Wear A Mask : Rule Breaker Detection

Xiao Yuchao, Geet Jethwani , Yan Weiquan

Institute of Systems Science, National University of Singapore,
Singapore 119615

ABSTRACT

In this report, a Wear-A-Mask rule breaker detector is proposed. With the technique of face detection and mask detection, it can recognize who doesn't wear a mask at a high accuracy. With the technique of face reidentification, it can compare the face feature of rule breaker with the stored face features in the database and identify who the rule breaker is. With the technique of image enhancement like denoising, illumination improvement and super resolution. This system can be applied to more extreme scenario such as dark environment, equipped with low end camera, etc.

Key words: Mask detection, Face reidentification, image enhancement, deep learning, intelligent sensing technique

1. INTRODUCTION

1.1. Mask Detection

Nowadays, wearing a mask is the most important way to keep yourself and other people safe because of the COVID-19. It is very important to find the people who remove their mask at common facilities, especially in a high population density but small space place like 7-11 convenience store, so that the manager of facilities can remind or ask those people to wear their mask. As we all known that the number of people in a common facility is dynamic, so there is not a changeless face database about all people who are in this facility.

To find the people who does not wear a mask, there are two main mission: 1. Capture a face image and the Identification code of the person at the entry of facility; 2. Identify the person that without a mask via the face image of the database. There are two technique difficulty: 1. How to detect face accurately and quickly and capture image clearly; 2. Identify face accurately via one image per person. Besides dealing with above questions, we using multithreading to achieve capturing image while inputting information.

The World Health Organization (WHO) reports suggest that one of the primary course of transmission of the COVID-19 infection is through respiratory droplets.

Respiratory droplets are generated when an infected person coughs or sneezes and any person in close range with the person is at a risk of being infected and that is the primary reason we have been advised to wear a mask by the World Health Organization.

Thus, it is always important to wear a mask, not just for our safety, but for others. To ensure this the government has proposed several rules and regulations where in every person always must mandatorily wear a mask. This however, needs to be checked by an individual or authorities at all times which can be cumbersome. In order to simplify the process we plan to build a wear a mask : rule breaker detection system, where in every person is being monitored at all times at public places,like a shopping complex, and in case the person is seen without a mask , the person can be identified and traced down. In order to do that we have the following sub components like face detection , face recognition , mask detection and image enhancement.

1.2. Image Enhancement

Computer vision has been widely applied in our daily life, e.g. face detection, advanced driver assistance system (ADAS), defect detection. However, to build an efficient cv-based system, we not only need a appropriate model, but also high quality image dataset. It could be very difficult to promise all the images collected are in good quality and in correct format. That's why we need image enhancement. Image enhancement has played and will continue to play an important role for computer vision. Our project, wear a mask (WAM) rule breaker detection, is computer vision-based system. However, what we propose is a software to detect WAM rule breaker in a public area such as shopping mall or library. Due to the different surrounding environments, different image capture device and other conditions, the quality of input image may be high or low so that we introduce the image enhancement into our project to make sure the system works well even under bad conditions.

To detect if a person is wearing a mask, there are two main difficulties: 1. In low light environment, it is difficult for the face detection model to capture face in the raw image; 2. If the person is far from the camera, it is difficult for the detection model to capture the details of the full body. Since for each full body bounding box, it only contains little pixels. Aim to solve these two problems, illumination improvement and super resolution techniques are applied in image enhancement part. Besides these two techniques, which is based on deep learning approach, we also implement some traditional image enhancement techniques like point-based processing, area-based processing and histogram-based processing.

2. RELATED WORK

2.1 Face Detection

Traditional face detection uses neural networks (such as multi-layer perceptrons) and classifiers (such as SVM) to realize the binary classification of whether a certain area of the image is a human face. After the success of image classification by convolutional neural networks. It was quickly used in the field of face detection and face recognition. In 2015, Haoxiang Li et al. proposed the Cascade CNN model. The model is composed of multiple convolutional networks that are used as classifiers. In the same year, Lichao Huang et al. proposed the DenseBox model. This model uses a fully convolutional network to directly predict the target rectangle and target category confidence in the same network. By locating key points at the same time of detection, the detection accuracy is further improved. In 2016, Kaipeng Zhan and others proposed the MTCNN model, which uses a network to complete the tasks of face region detection and face key point detection. The model is improved on the basis of Cascade CNN, but the overall idea is cleverer and more reasonable. This model is a model widely used now.

Our work is carried out on the basis of MTCNN, because MTCNN uses a network to complete the tasks of face detection and face key point detection, which can perform face alignment after face recognition. And the overall complexity of the model is well controlled, and real-time face monitoring can be achieved.

2.2 Face Recognition

The core of face recognition is the calculation of image matching. Traditional algorithms use geometric features or global features combined with PCA and other data dimensionality reduction algorithms to realize face recognition. After deep learning began to be applied in the field of face recognition, the recognition speed and accuracy have been improved by leaps and bounds. In 2014, FaceBook proposed the DeepFace model on CVPR, which consists of 6 convolutional layers and 2 fully connected layers. The model can use softmax to output the classification of 4030 characters in the last fully connected layer. The model can also use Relu as the activation function to extract features, and then use methods such as Siamese network to verify whether the two faces belong to the same person. In the same year, Xiaoou Tang et al. proposed the DeepID model and published it in CVPR2014. Its application field is face verification, which is to judge whether two pictures are the same person. The network architecture of this model consists of 4 convolutional layers and 1 fully connected layer. The purpose of this model is to extract features with high quality. After extracting the facial features of the two pictures, first connect the two feature vectors, and then use the joint Bayesian classifier to determine whether they belong to the same person.In 2015, the Google team developed a new face recognition system: FaceNet, which can directly map face images to Euclidean space. The distance of the space represents the similarity of face images. The team also used a new loss function: triplet loss. The choice of the loss function is very important for the convergence of the model.

Our face recognition work is carried out on the basis of FaceNet. The FaceNet model has strong face extraction capabilities, and has good adaptability to face mirroring, face rotation, and lateral faces. This is very helpful for our project.

2.3 Face mask Detection

In the mask detection the approach is to complete the face detection first and then apply a masknet model in which we extract the features and then apply a classifier to detect the mask .One approach on perfoming this task[1] is to apply deep transfer learning (ResNet50) as feature extractor and then apply a classifier such as decision tree or SVM . Resnet-50 is a robust feature extractor compared to traditional Machine learning approaches [2] .

A residual neural network (ResNet) is a kind of deep transfer learning based on residual learning. The advantage of using versions of Resnet are to get rid of the vanishing gradients that have their specific residual block.[3] It basically has a convolution layer followed by a fully connected layer.

An approach used with Resnet-50 layers is to replace the last layers of the model with traditional machine learning classifiers.
In classification, the last layer in resnet-50 is removed and replaced with clasiifier like SVM. Oher possible classifiers include decision tree and ensemble techniques.

2.4 Image Enhancement

2.4.1 Denoising

The basic mathematical definition for the image with noise is:

$$Y = X + E$$

Where X is the what the image should be (without noise). E is the noise and Y is the representation of the image. However, we always can only get the image Y and try to separate the X from Y without the knowledge of E. For this goal, we need to utilize some properties of image to do the denoising. According to the properties being used or the related techniques, the denoising methods can be divided into:

1. Filtering
2. Sparse coding
3. External prior knowledge
4. Deep Learning

Filtering is adopted in our project, the idea of filtering is tried to design a filter, convolve the image with that filter and remove the noise.

The idea of sparse coding is people found the nature image (without noise) can usually be represented as a sparse matrix by some specific models, while the noise cannot. Therefore, we regularize the image with this sparseness property to remove the noise.

The idea of external prior knowledge is to find the pattern from the images without noise. According to the found pattern from the no-noise images, we can regularize the other images to remove the noise.

For the deep learning, that can be in different ideas for different networks, such as filtering or find the external prior knowledge or in other ideas. The difference is we utilize the power of deep network to find the optimal filter or other regularizer automatically to denoise the image.

2.4.2 Illumination improvement

Improving the lighting condition is always important in image processing field. Since many image processing methods are proposed to process the image in daytime, they always perform bad when the input image is in dark environment. Many researchers tried to recover the image to a good lighting condition with different methods. In traditional processing method, we have gamma transformation, histogram equalization, etc. In deep learning filed, we can divide the methods into supervised and semi-supervised methods. For example, HDRNet trained a convolution network to generate bilateral grid of coefficients and apply coefficient layer to the preprocessed low light image to generate a high-quality image (supervised).

2.4.3 Super resolution

For super resolution, we also have traditional and modern methods. For the traditional method, like bilinear interpolation, we calculate the value for added pixel according to the pixel's neighborhood. However, this result of this method is always blurred, without the detail of image. Nowadays, we always use deep learning technique to do the super resolution. SRCNN proposed a simple convolution network, upsample and reconstruct the image into a higher resolution. Generative adversarial network is widely applied in super resolution. SRGAN is based on this idea. It trains generator and discriminator simultaneously to generate the high resolution image.

3. PROPOSED APPROACH

3.1 Face Detection – MTCNN

Face detection can be regarded as a specific case of object-class detection which focus on the detection of frontal human faces. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. In our project, face detection plays a role in capturing face images and mask detection. For the capturing face, the pain point is that the program should capture image clearly and accurately so that the captured image can be used as database for face identification. In order to deal with the pain point, we choose the MTCNN to do the face detection job after we compared the face recognition interface of OpenCV, the face recognition model of align and the MTCNN model.

3.1.1 MTCNN

3.1.1.1 MTCNN Overview

MTCNN is a python (pip) library written by Github user ipacz, which implements the paper Zhang, Kaipeng et al. "Joint Face Detection and Alignment Using Multi - task Cascaded Convolutional Networks." IEEE Signal Processing Letters 23.10 (2016): 1499–1503. Crossref. Web.

MTCNN model is composed of three CNN networks that named Proposal Network(P-Net), Refine Network(R-Net), Output Network(O-Net). The image is processed by P-Net, R-Net, O-Net in turn, and then outputs the bounding box and key feature points of the face. Therefore, MTCNN can simultaneously realize face detection and face alignment.

3.1.1.2 Workflow and Structure of MTCNN

The implementation process of MTCNN is divided into three parts: 1. Enumerate candidate regions; 2. Exclude non-human regions; 3. Further finely align regions and feature points. Its workflow is as follows:

1. Use bilinear interpolation to resize the image to different sizes at a certain ratio to form an image pyramid. This step is for the network to detect faces of different scales.
2. The network structure and schematic diagram of Proposal Network (P-Net) are as Figure 3.1.

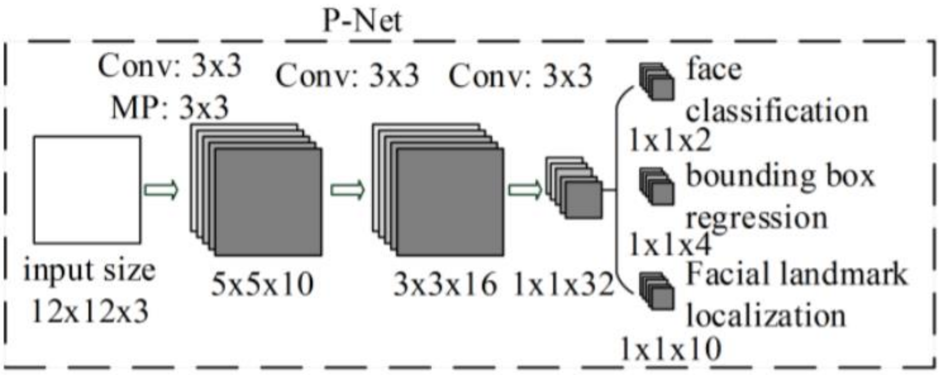


Figure 3.1 Schematic diagram of P-Net structure Data source: Zhang, Kaipeng et al. "Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks."

Generate a number of 12*12 windows by sliding the window on the input picture, and input the generated window into P-Net. P-Net roughly judges whether the input window contains a face, and then uses non-maximum suppression (NMS) to merge highly overlapping candidates. Since the detection is performed on a set of pictures of different sizes, candidate regions of different sizes can be generated when mapped to the original picture. The output result of P-Net is the image of all detected faces, the coordinates of the bounding box and the score of each result. In our project, there is no need to use the score item, so the output is omitted.

3. Resize all bounding boxes to 24*24 using bilinear interpolation, and then enter them into Refine Network (R-Net). The network structure and schematic diagram of R-Net are as follows:

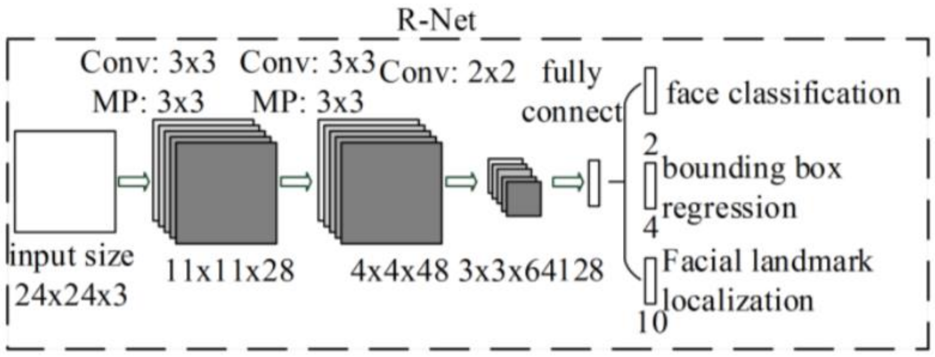


Figure 3.2 Schematic diagram of R-Net structure

Data source: Zhang, Kaipeng et al. "Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks."

R-Net will judge whether all bounding boxes contain faces and use NMS to merge bounding boxes with high overlap. This step will eliminate most of the candidate bounding boxes and return the bounding boxes most likely to contain human faces. The same reason as the previous network, so the output of the scoring item is ignored.

4. Use bilinear interpolation to resize the bounding box output by R-Net and then enter the Output Network (O-Net). The network structure and schematic diagram of O-Net are as follows:

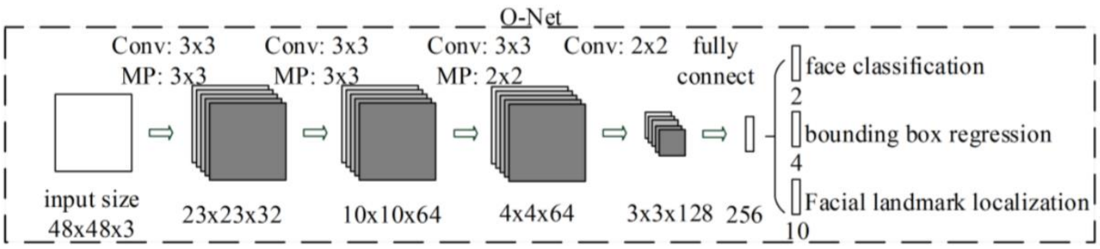


Figure 3.3 Schematic diagram of O-Net structure

Data source: Zhang, Kaipeng et al. "Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks.

O-Net is similar to R-Net, but this stage will further analyze and process the characteristics of these bounding boxes. While eliminating some of the bounding boxes, it will return the bound box containing the face, the coordinates of the five Landmarks (eyes, Nose tip, mouth corner) and the score items.

3.1.1.2 MTCNN in Our Project

Since MTCNN needs to use a lot of data for training, in our project, we directly use the pre-training model of MTCNN to realize the face detection function. We used a data set containing more than 16,000 images to test the model many times, and the accuracy of face detection was over 97%. The face detection results are as Fig. 3.4.

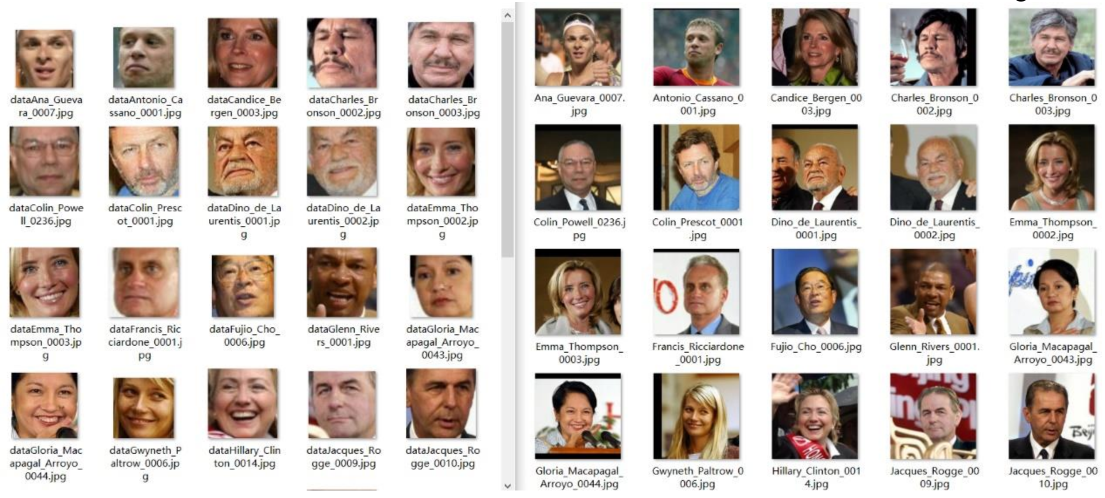

dataAna_Guevara_0007.jpg
dataAntonio_Cassano_0001.jpg
dataCandice_Bergen_0003.jpg
dataCharles_Bronson_0002.jpg
dataCharles_Bronson_0003.jpg
dataColin_Powell_0236.jpg
dataColin_Prescot_0001.jpg
dataDino_de_Laurentis_0001.jpg
dataDino_de_Laurentis_0002.jpg
dataEmma_Thompson_0002.jpg
dataEmma_Thompson_0003.jpg
dataFrancis_Ricciardone_0001.jpg
dataFujio_Cho_0006.jpg
dataGlenn_Rivers_0001.jpg
dataGloria_Macapagal_Arroyo_0043.jpg
dataGloria_Macapagal_Arroyo_0044.jpg
dataGwyneth_Paltrow_0006.jpg
dataHillary_Clinton_0014.jpg
dataJacques_Rogge_0009.jpg
dataJacques_Rogge_0010.jpg
Ana_Guevara_0007.jpg
Antonio_Cassano_0001.jpg
Candice_Bergen_0003.jpg
Charles_Bronson_0002.jpg
Charles_Bronson_0003.jpg
Colin_Powell_0236.jpg
Colin_Prescot_0001.jpg
Dino_de_Laurentis_0001.jpg
Dino_de_Laurentis_0002.jpg
Emma_Thompson_0002.jpg
Emma_Thompson_0003.jpg
Francis_Ricciardone_0001.jpg
Fujio_Cho_0006.jpg
Glenn_Rivers_0001.jpg
Gloria_Macapagal_Arroyo_0043.jpg
Gloria_Macapagal_Arroyo_0044.jpg
Gwyneth_Paltrow_0006.jpg
Hillary_Clinton_0014.jpg
Jacques_Rogge_0009.jpg
Jacques_Rogge_0010.jpg

Fig. 3.4 Result of Face Detection

Data source: Dataset LFW

3.1.2 Face Detection

Using MTCNN for face detection can output multiple faces in the picture at the same time. In our project, we plan to install information entry and face detection modules at the entrance of public facilities, so we only expect to return one face detection result each time. We set the highest score item of the face detection return result as the detected face. This face will first be expressed as a 128-dimensional feature using a feature extraction network (explained in detail below) and stored in a data file (.CSV), and then will be stored in the image database.

3.2 Face Recognition – FaceNet and One-Shot-Learning

Face recognition is a very broad concept, which includes face identification and face verification.

Face recognition is a very broad concept, which includes face identification and face verification.

The former realizes the matching of the human face with the pictures in the database, and the latter realizes the judgment whether the human face belongs to the database and the verification of the correctness of the matching result. The core of face recognition is the recognition of image features. In our project, it is shown as a match between the detected face and the face in the database.

Our project plans to deploy face recognition systems in public facilities with small spaces. These facilities are characterized by a large flow of people, and there are almost no people who are stable in the facilities. Therefore, the face database of such a facility is dynamic and the data scale is small. Therefore, we aim to use a dynamic small-scale database to realize face recognition work, which can also be considered as face re-recognition work.

Face recognition can be seen as a multi-classification task, but in many practical applications it is difficult to use ordinary classifiers to process. One of the reasons is that there are few training samples for each class, and it is difficult for us to obtain a large amount of data for the same classification. Even if the classification is small, it is difficult to obtain a good classification effect. In addition, there is a more difficult problem to deal with is retraining. Since our database is dynamic, if a classifier is used, the classifier must be retrained for each additional person. This is obviously not practical and reasonable. This forces us to use a classifier that does not require training or not to use a classifier. Classifiers that do not require training can be implemented with KNN. For the classification of a photo, 1-NN can be used to calculate the distance between the input image and each image in the database, and the smallest distance is regarded as the classification result. However, directly calculating the image distance has a low processing power for face images or profile images with a certain angle.

Therefore, we use deep learning-based methods to achieve end-to-end learning of image feature extraction and feature comparison at the same time by only use one image of each face and that named One-Shot-Learning.

The input of this model is the two pictures waiting to be compared, and the output is the distance between the two pictures. The pictures to be recognized and the database pictures are sequentially input into the network, and the distance between the pictures to be recognized and all the database pictures is calculated to realize face identification.

Our deep learning network is divided into two parts, feature extraction network and distance calculation network. The feature extraction network is two symmetrical convolutional neural networks. The parameters and weights of the two networks are exactly the same. After feature extraction, two face images can output two similar feature vectors. The distance calculation network is a dual-input fully connected network, the input of the network is two feature vectors, and the calculation result is the distance between two pictures. Next, the implementation of the feature extraction network and the One-Shot-Learning will be introduced in detail.

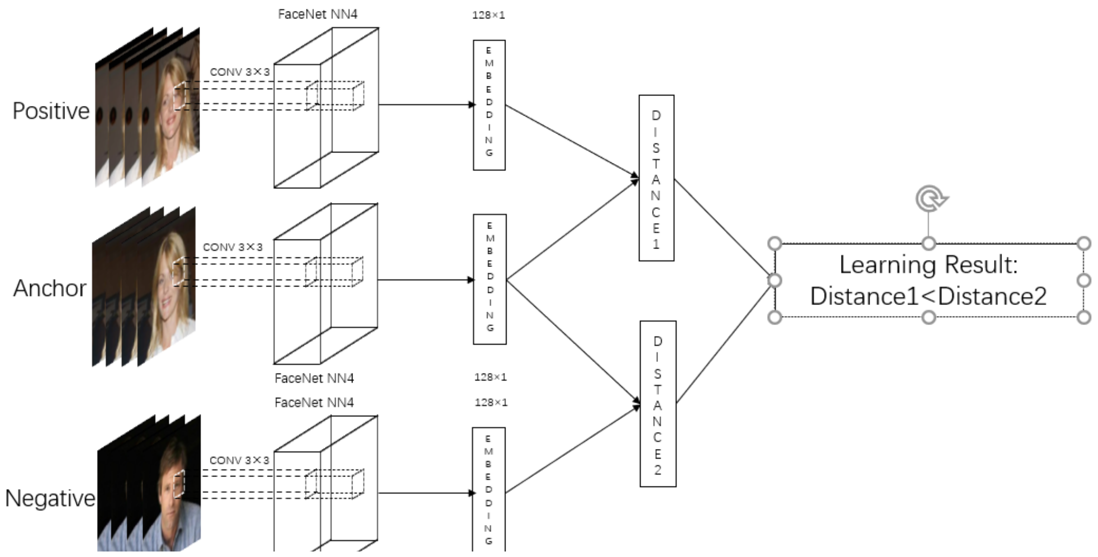


Fig. 3.5 Triplet Loss Learning Data source: Dataset LFW

3.2.1 FaceNet model

In 2015, Google published an article entitled: A Unified Embedding for Face Recognition and Clustering. In this article, they proposed a method system called FaceNet, which directly learns the mapping of images to points on Euclidean space. FaceNet uses multiple complex CNNs to map human faces to Euclidean feature space.

We analyze the structural parameters of four series CNN: NN1, NN2, NN3, NN4 through the paper. NN1 has a staggering 140M parameters, which will cause a great burden on the calculation of the machine. Even the tailored NNS1 has 26M parameters, which means that a machine with excellent performance is needed.

Obviously the NN1 series is not practical. NN2 only needs 7.5M parameters. Compared with NN1, it has greatly reduced the demand for machine performance. NN3 and NN4, compared to NN2, have the same network structure, but with smaller input size, which further reduces the burden on the machine.

After analysis and calculation, NNS4 has only 3M parameters, which is the least among all FaceNet series networks, which is very helpful for us to deploy the application on a machine with ordinary performance. The structure of NN4-Small is shown in the following table:

3.2.2 Triplet Loss

FaceNet also proposed a new loss function: Triplet Loss, which is used for training the feature extraction network. Triplet Loss is currently a widely used loss function. Its purpose is to increase the distance of pictures of different classes and reduce the distance of pictures of the same class. The training input is a set of three pictures: Anchor, Positive and Negative. The training process is: input the three pictures into the network in turn, extract the features, and then calculate the distance between Anchor and Positive, and Anchor and Negative respectively. Expressed as a loss function:

$$L = \sum_{i}^{N} \left[\left\| f(x_i^a - x_i^p) \right\|^2 - \| f(x_i^a - x_i^n) \|^2 + \alpha \right]$$

After the loss function output is rectified by ReLU, the function behaves as follows: if the distance of the same type of pictures plus α is less than the distance of different types of pictures, there is no loss=0; if it is greater than the distance of different types of pictures, then loss is the function output value. In this way, the model can make the same class pictures in the feature space gather and keep different class pictures away.

3.2.3 One-Shot-Learning

For dynamic small-scale databases, we use the One-Shot-Learning method to realize face identification. One-Shot-Learning uses only one or a few pictures as a database to learn information about object categories. Siamese Network is a network structure commonly used to perform One-Shot-Learning. This network consists of two completely symmetrical networks and a combined network. The parameters and weights of the two symmetric networks are exactly the same, which ensures the symmetry of the output results. When using, input the measured picture and database picture at the same time, and the network outputs the similarity value of the two pictures. This similarity value can be trained with Triplet Loss as the loss function to output the distance between the two images, and force the network to bring the images of the same class closer.

For our project, we hope that the image will be separated from the final similarity calculation network in the feature extraction network, so that the face in the database can be stored as a feature vector in advance, which can effectively improve the comparison calculation speed. Because FaceNet has a very good performance in the extraction of facial features, we use FaceNet NN4-Small as the feature extraction network, and the output feature vector is then calculated by a fully connected network to calculate the image distance. The system structure is shown in the figure below:

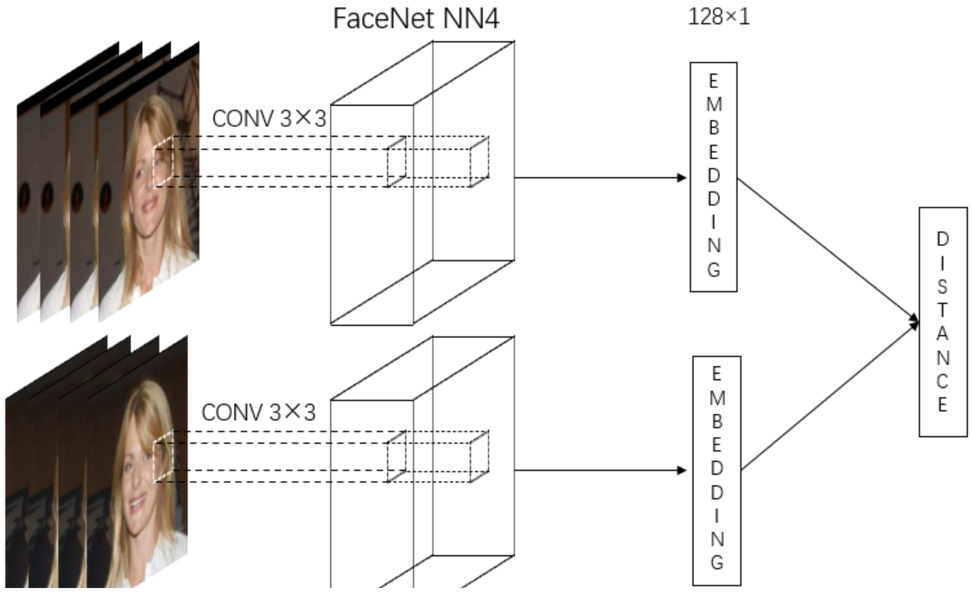


Fig. 3.6 One-Shot-Learning in our project

Data source: Dataset LFW

In actual use, we first save the face database data into a data table file (.CSV). When performing recognition work, only the detected face needs to be input to FaceNet to extract features, and then input to the distance calculation network at the same time as the face feature vector in the data file, and finally the distance between the two face vectors can be obtained.

3.3 System Structure

3.3.1 Face Detection

The face detection module includes face detection, information entry and feature vector storage. We use three threads to perform three tasks. The main thread opens the camera and creates two video windows. Window 1 displays the image captured by the camera, and window 2 detects the largest face image (when no face image is detected, window 2 simultaneously displays the image captured by the camera). The main thread initialization phase initializes the information entry identifier and starts the sub-thread 1, and the information entry task starts. We use the wait attribute of the input function itself to stall the child thread 1 to wait for information input. When a user enters information, the identifier changes, sub-thread 1 is closed, and the main thread starts sub-thread 2, which stores the captured face

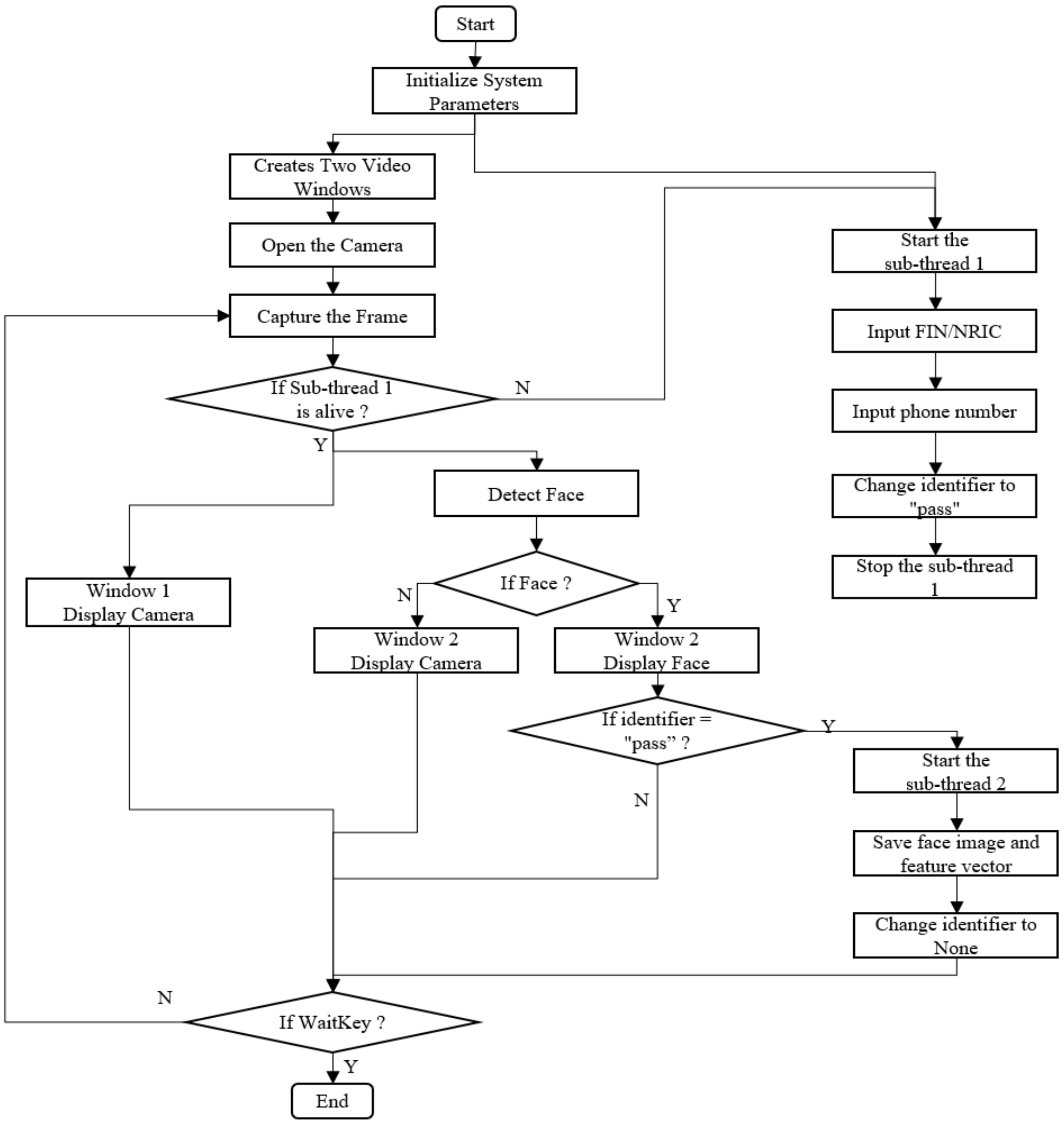


Figure Fig. 3.7 Face Detection module flow chart image and appends the facial feature vector to the data file (.CSV). The system execution flow chart is as Fig. 3.7:

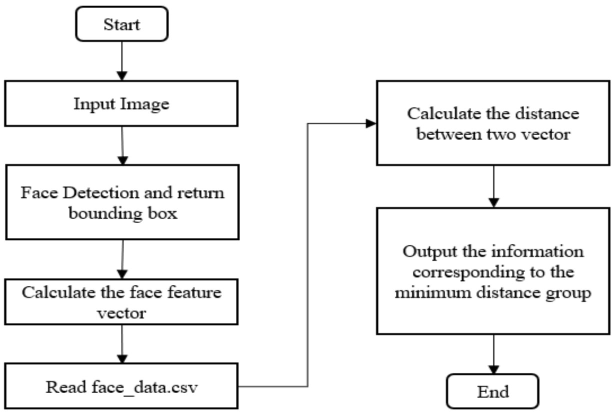

Start
Input Image
Face Detection and return bounding box
Calculate the face feature vector
Read face_data.csv
Calculate the distance between two vector
Output the information corresponding to the minimum distance group
End

Fig. 3.8 Face Recognition module flow char

3.3.2 Face Recognition

Face recognition starts after mask detection. When the detected face is not wearing a mask, the previous program sends the image captured by the camera to the face recognition module. First, use the human detection module to extract the face image, and then input the face image into FaceNet to extract the features. Finally, the extracted feature vector and the feature vector of the data file are sequentially input into the network and the Euclidean distance between the two features is calculated. Because the face data in the database is collected from the entrance of public facilities, the person recognized by the face must be included in the database, so the step of face verification can be omitted, that is, it is omitted to verify whether the person is in the database. We default that the group of faces with the smallest feature vector distance is the face recognition result we need. The system execution flow chart is as Fig. 3.8:

3.4 Mask Detection

Amidst the COVID-19, we all are in a phase wear wearing a mask is a must. To keep everyone safe it is a mandate to wear a mask in public places and shops.

The dataset contains of data that is with mask and without mask. The dataset contains images of people with mask and without mask that has been taken from internet sources (opensource image libraries , Kaggle and google images) there are about 700 images of each type .The model to perform mask detection was trained on this data.

Fig. 3.9. No Mask

Fig. 3.10. Mask

Approaches to mask detection :
1.Train Deep learning model (MobileNetV2)
2.Apply mask detector over images / live video stream
load the MobileNetV2 network

1. Train Deep Learning Model

The first step in our model training is to initialize the number of epochs(20) to train for,
and batch size (32) . Once this is done , we can move ahead and grab the list of images in our dataset directory and then initialize the list of data (in our case it is images) and class images.
Then we need to perform one-hot encoding on the labels and then construct the training image generator for data augmentation.

Once this is done , we load the MobileNetV2 network, ensuring the head FC layer sets are
left off and construct the head of the model that will be placed on top of the the base model. Loop over the base model and freeze them so they will be updated during the first training process.
Then we place the head FC model on top of the base model this is the model we will train on.
After we have set the model that we want to train ,

we compile our model and train the head of the network and make predictions on the testing set.
At last we calculate the accuracy and Save the model if the accuracy obtained is sufficiently good enough.

All the steps taken place while training our model is as follows :

- initialize the following :
    - learning rate
    - number of epochs
    - batch size
- Initialize the list of data and class images.
- load the MobileNetV2 network
- place the head FC model on top of the base model
- loop over the base model and freeze them
- compile our model and train the model
- make predictions on the test set
- Save the model

The reason why as to we selected MobileNetV2 is because as per ImageNet dataset MobileNetV2 outperforms MobileNetV1 and ShuffleNet .

It performs well even on small dataset.

Table 1 : MobileNetV2 Pros

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	3.4M	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	1183ms
MobileNetV2	72.0	3.4M	300M	75ms
MobileNetV2 (1.4)	74.7	6.9M	585M	143ms

The flow of the model training is shown below:

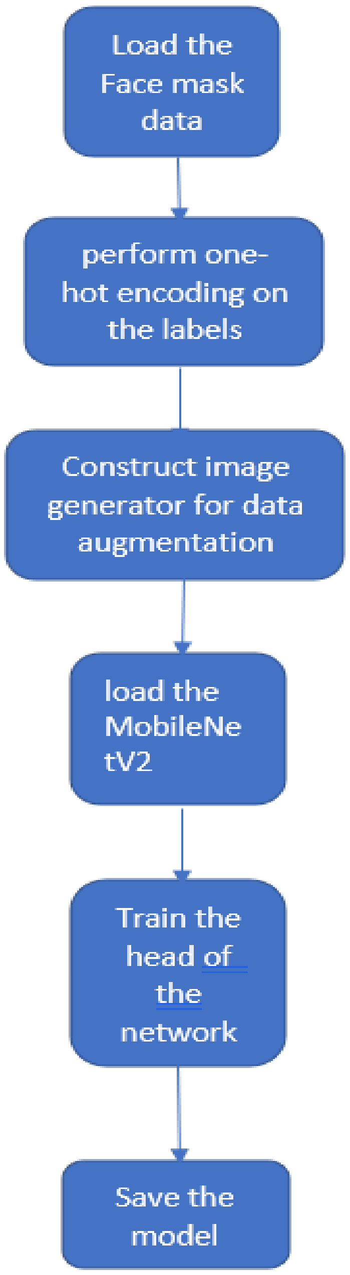

Load the Face mask data
perform one-hot encoding on the labels
Construct image generator for data augmentation
load the MobileNetV2
Train the head of the network
Save the model

Figure. 3.11. Model training flow

2. Apply mask detector over images

For this step grab the dimensions of the frame and then construct a blob from it.

OpenCV dnn – deep nueral network module contains a function that can be used for preprocessing images called blob .

So, in the first step we just grab the frame and construct a blob from it and then pass it through the network and obtain the face detections. We then pass the blob through the network and obtain the face detections. Once we're done with that, the next step is to initialize the list of faces, their corresponding locations and the list of predictions from our face mask network.

Then we extract the confidence or estimate the probabilities associated with the detection. Once the confidence levels are extracted, set the minimum confidence (0.5) and if the values are greater than those confidence levels the bounding box for the object is computed. Basically, the (x,y) coordinates are computed for the bounding box .Also , it has to be ensured that the bounding box falls within the dimensions of the frame .

Once these steps are performed , we extract the face ROI and convert it from BGR to RGB channel and then resize it .

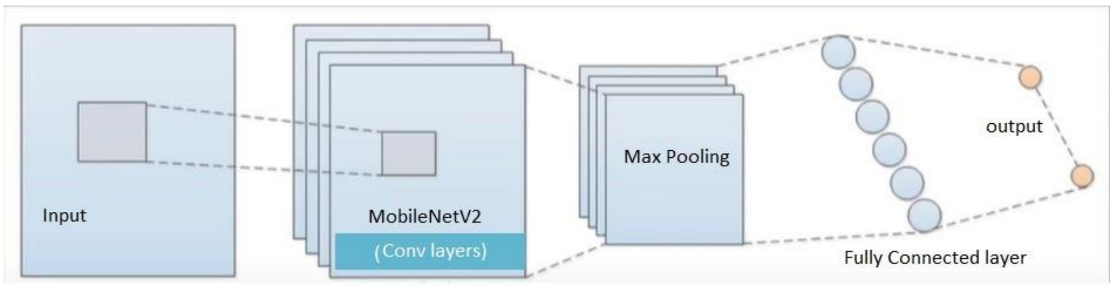


Figure. 1.12. Model training flow

Finally, we add the faces and bounding box to the respective lists and make predictions if at least one face was detected in the frame.

While making predictions the aim is to make those predictions and inferences faster .

Thus , for faster inference we made batch predictions on all the faces at the same time rather than performing predictions one by one .

Since we had trained our model earlier, we load the mask detector model from the disk and initialize the video stream. Once the video is turned on, we loop over the frames through a for loop. We grab the frame from the threaded video stream and resize it (800 pixels to cover a larger part of the screen) and then detect faces from then determine if they are wearing a mask or not.

If the person is wearing a mask we display a green bounding box around the face to indicate that the person is wearing a mask and that he can be allowed to either go inside the store or move around the shopping complex without restriction.

In case the person is not wearing a mask , he will not be allowed inside the store and in case he removes the mask within the store an alert is sent to the authorities in the form of a screen pop up and an email notification.

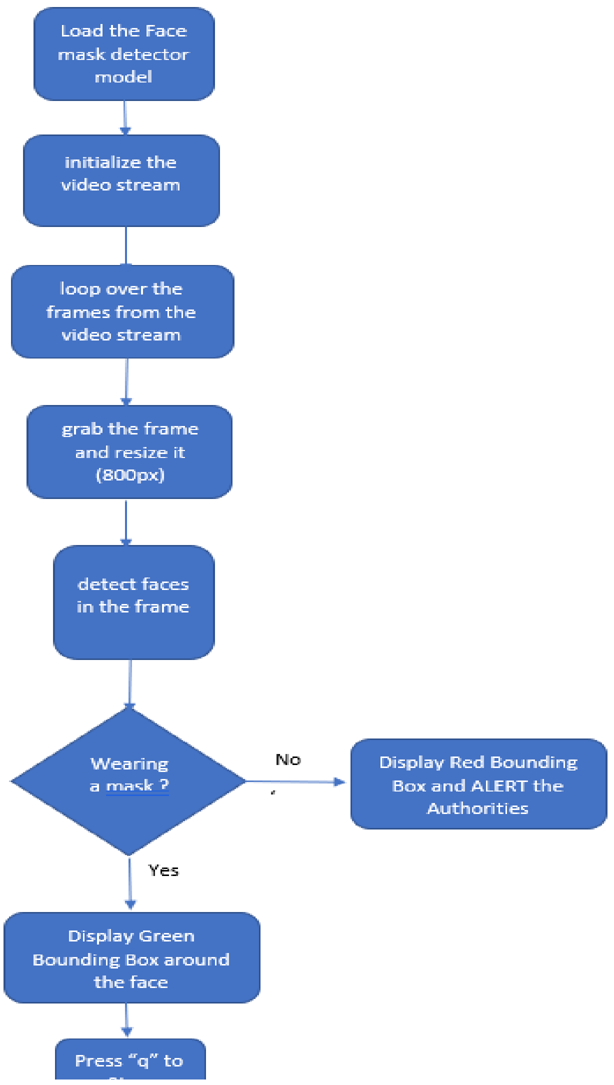

Load the Face mask detector model
initialize the video stream
loop over the frames from the video stream
grab the frame and resize it (800px)
detect faces in the frame
Wearing a mask ?
No
Display Red Bounding Box and ALERT the Authorities
Yes
Display Green Bounding Box around the face
Press "q" to

Figure 3.13 : Apply Mask Dataset

.

3.5. Image enhancement

3.5.1. area-based processing – Sharpening filter

Image sharpening is used to strengthen the difference in the image, e.g. the value difference between two neighborhood pixels. Sharpening filter is used to sharpen the image. Generally, a filter will be a 3x3 matrix, which is shown in Fig. 3.14.

x_{11}	x_{12}	x_{13}
x_{21}	x_{22}	x_{23}
x_{31}	x_{32}	x_{33}

Fig. 3.14. 3x3 Filter

Where $x_{rc}$is the value in r row and c column of the filter.

One of the classic sharpening filter is Laplacian sharpening filter. The idea of Laplacian sharpening is to calculate the difference or gradient in two directions: horizontal direction and vertical direction. The gradient is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

For image, the gradient calculation will be a linear operation, which is:

$$\frac{\partial^2 f}{\partial x^2} = [f(x,y) - f(x+1,y)] + [f(x,y) - f(x-1,y)]$$

$$\frac{\partial^2 f}{\partial y^2} = [f(x,y) - f(x,y+1)] + [f(x,y) - f(x,y-1)]$$

$$\nabla^2 f = 4 * f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)$$

So the sharpening filter will be:

0	-1	0
-1	4	-1
0	-1	0

Fig. 3.15. Laplacian sharpening filter

However, if we do the convolution operation on the image with the Laplacian sharpening filter, we only keep the edge information and lose most of the information in original image. So we need to add a identity filter which is to keep the original information of the image. The procedure can be explained as:

$$Info_{with_{grad}} = Info_{ori} + Info_{only_{grad}}$$

The procedure of the sharpening is shown in Fig.3.16.

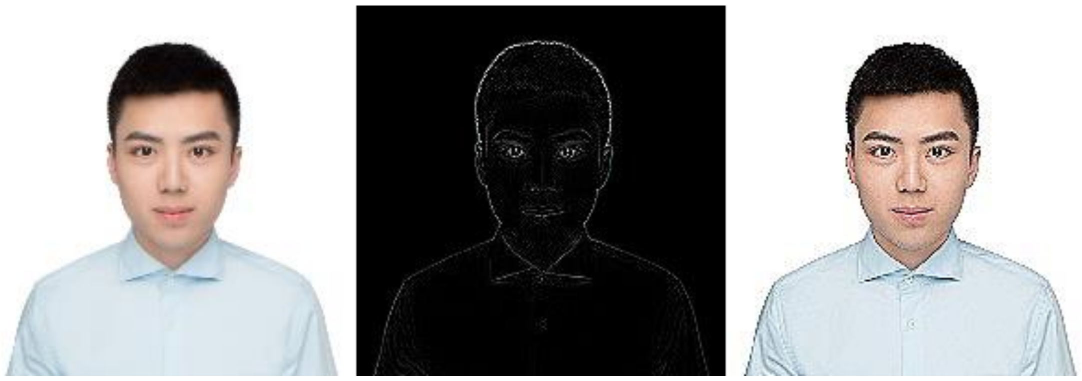


Fig.3.16. Original image (left), Edge image (middle), image with gradient information (right)

3.5.2. area-based processing – Denoising filter

There are two classic denoising filter: 1. Smoothing filter (Average filter) 2. Median filter. These two filters are applied to eliminate different kinds of noise. A 3x3 average filter is defined as:

1	1	1
1	1	1
1	1	1

Fig. 3.17. Average filter

Average filter is used to remove the noise where the average of noise is close to zero and the value of noise is much smaller than the real value of pixel. With average filter, the value of processed pixel will be:

$$f^{new}(x,y) = 1/9 \sum_{\substack{i\in[x-1,x+1],\\ j\in[y-1,y+1],\\ i,j\neq(x,y)}} f(i,j)$$

The procedure of the convolution with average filter is shown in Fig.3.18.

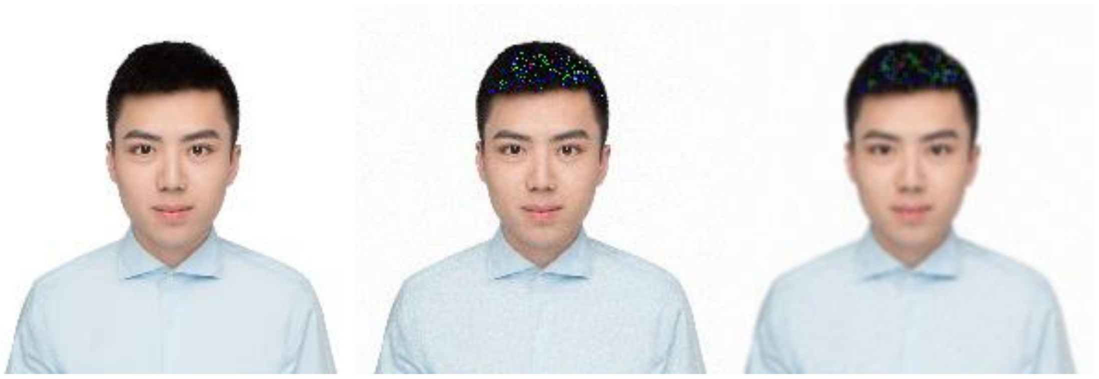


Fig.3.18. Original image (left), image with gaussian noise (middle), image processed with average filter(right)

Similar to the average filter, median filter can also remove the noise. However, median filter is better at handling salt-and-pepper noise instead of gaussian noise. The definition of 3x3 median filter is:

$$f^{new}(x,y) = median\big(f(i,j)\big), i \in [x-1, x+1], j \in [y-1, y+1]$$

The value salt-and-pepper noise is extremely high (close to 255) or extremely low (close to 0). For median filter, it only considers the median of the surrounding pixels so it can avoid the influence of salt-and-pepper noise. The procedure of convolution with median filter is:

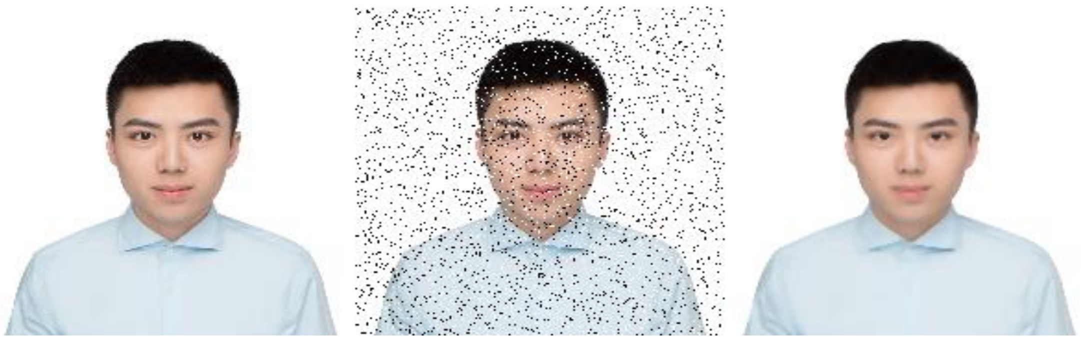


Fig. 3.19. Original image (left), image with salt-and-pepper noise (middle), image processed with median filter(right)

Obviously, applying these smoothing filters to eliminate the noise will introduce a new problem – blur image. However, in image processing field, noise always has a worse impact for subsequent operation, e.g. object detection, so we still use these filters to avoid the noise.

3.5.3. Histogram-based processing– Histogram equalization

Histogram equalization is a simple and effective image enhancement technology, which changes the gray level of each pixel in the image by changing the histogram of the image, and is mainly used to enhance the contrast of the image with a small dynamic range. In real world, our raw image may be concentrated in a narrow interval due to the lighting condition, resulting in the image not being clear enough. For example, the gray level of an overexposed image is concentrated in the high brightness range, while underexposure will make the gray level of the image concentrated in the low brightness range. Using histogram equalization, the histogram of the original image can be transformed into a uniformly distributed (balanced) form, which increases the dynamic range of the gray value difference between pixels, thereby achieving the effect of enhancing the overall image contrast. In other words, the basic principle of histogram equalization is: broaden the gray value with a large number of pixels in the image (that is, the gray value that plays a major role in the picture), and broaden the gray value with a small number of pixels (that is, The gray values that are not the main function of the picture) are merged, thereby increasing the contrast, making the image clear, and achieving the purpose of enhancement.

The steps of histogram equalization algorithm are as follow:

1. Calculate the histogram of the distribution of pixel value. (Suppose we have gray scale image, pixel value from 0 to 255)
2. Calculate the probabilities of each value where
$$P(value = x) = \frac{Count(value = x)}{Total\ number\ of\ pixels}$$
3. Calculate the distribution function where
$$cdf(value = x) = \sum_{i=0}^{x} P(value = i)$$
4. Histogram equalization.
$$f^{new}(x, y) = round\left(\frac{cdf\big(f(x, y)\big) - cdf_{min}}{R * C - cdf_{min}} * (L - 1)\right)$$

The transformation is shown in Fig. 3.20.

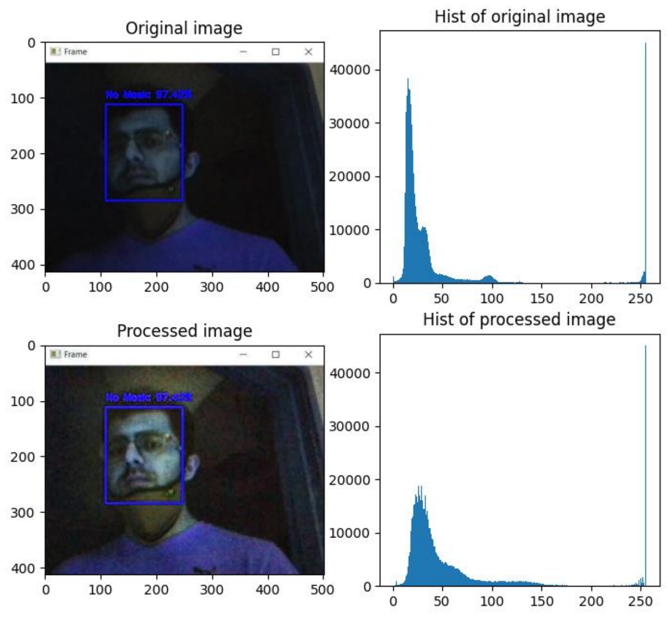


Fig.3.20. The change of distribution of pixel value before and after histogram equalization

In indoor place such as supermarket, generally we will not encounter very dark lighting condition. The problem is more likely to be unbalanced distribution of pixels (The values of pixels concentrate into a small interval). As a result, histogram equalization is really useful here to improve the quality of an image.

3.5.4. Illumination improvement—mbllen network

Histogram equalization can help to improve the lighting condition to some extent. However, if the lighting condition is really bad, we may not hope the histogram equalization can solve this problem perfectly. We should use illumination improvement technique instead. MBLLEN network is one of the modern techniques to improve the lighting condition for low light image. With this technique, we can see the things in the dark which means we can also apply our system in outdoor area where the lighting condition is not as stable as in the indoor area. In this part, I will show you how the MBLLEN network works and how to train the MBLLEN network.

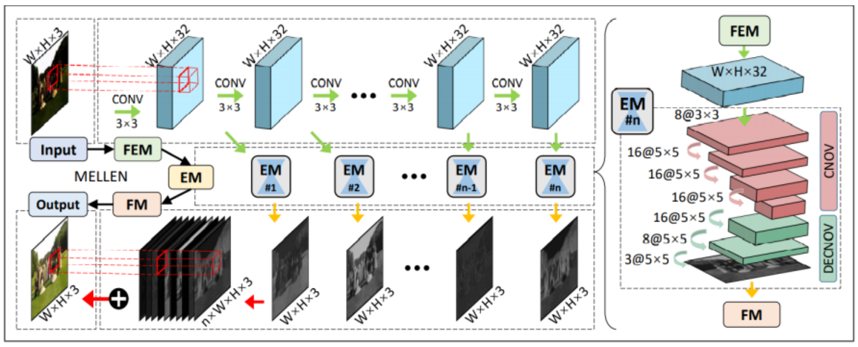


Fig. 3.21. The architecture of the MBLLEN network

3.5.4.1. Architecture

As shown in Fig. 3.21., we can see the network pipeline at the top is called feature extraction module (FEM). This pipeline is composed by 10 convolution layers, with kernel size of 3x3, activation function of ReLU, and no pooling layer. The output of each layer is distinct feature matrix with receptive field from small to large.

The middle pipeline is called enhancement module (EM), which is used to transform the original RGB features to lighting-enhanced features. For each EM, it contains one convolution layer with kernel size of 3x3, and three convolution layers with kernel size of 5x5, and three deconvolution layers with kernel size of 5x5 to recover the image to original size.

The bottom pipeline is called fusion module (FM), which is used to fuse all the outputs of EM. At each stage, FM will concatenate the output of EM and the current feature map for FM. To decide the how much information should be kept or abandoned, a kernel size of 1x1 is applied. This 1x1 kernel is same to define a learnable parameter for weighted sum, so it can learn what information should be kept and what information should be abandoned.

3.5.4.2. Dataset

To capture image in daytime and nighttime with same setting is really difficult, so the paper propose a synthetic dataset to train the network. The reference dataset is PASCAL VOC images dataset. For each image in PASCAL VOC images dataset, apply a random gamma adjustment to each channel to produce a low-light images. The process can be described as:

$$I_{out} = A * I_{in}^{\gamma}$$

where A is a constant determined by the maximum pixel intensity in the image and γ obeys a uniform distribution U(2,3.5). The size of training set is 16925 images, the size of validation set is 56 images, and the size of test set is 144 images.

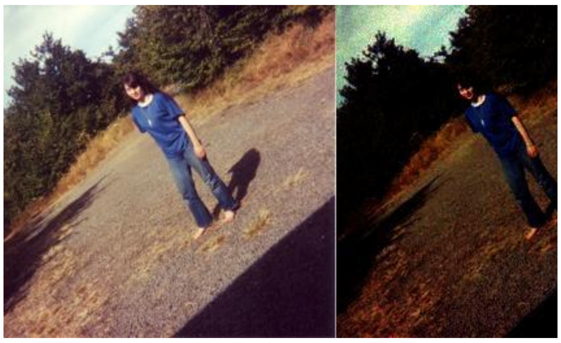


Fig. 3.22. One pair of original image (left) and synthetic low light image (right) in MBLLEN dataset

3.5.4.3. Loss function

Three different losses are applied for the consideration of structure information, context information and regional difference. For the structure information, two modern algorithm, SSIM (Structure Similarity) and MS-SSIM (Multiple-scale Structure Similarity) are utilized to calculate the structure loss. The definition of loss by SSIM and by MS-SSIM are:

$$Loss_{SSIM} = -\frac{1}{N}\sum_{n \in image}\frac{2u_x * u_y + C_1}{u_x^2 + u_y^2 + C_1} * \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

$$Loss_{MSSSIM} = \prod_{j=1}^{M}\left(Loss_{SSIM_j}\right)^{\beta_j}$$

$$Loss_{Structure} = Loss_{SSIM} + Loss_{MSSSIM}$$

Where N is N windows in an image, u_x is average of pixels in window n of image x, u_y is average of pixels in window n of image x. σ_x^2 is variance of pixels in windows n of image x. σ_y^2 is variance of pixels in windows n of image y. C_1 and C_2 are two constants. M is number of scales in MS-SSIM. β is parameter measured from the experiment.

For context loss, which is same to the perceptual loss. We use a pre-trained model to extract the features from two images and calculate the difference between two features instead of the output images. We apply this technique because the traditional loss like MSE only focus on pixel-level loss but we are more concerned about the difference of feature. Suppose we use VGG19 to do the extraction and the feature map is output from the 10th layer. The context loss will be defined as:

$$Loss_{VGG19-layer10} = \frac{1}{W * H * C}\sum_{W}\sum_{H}\sum_{C}||\emptyset(E)_{pixel} - \emptyset(G)_{pixel}||$$

Where W, H, C are width, height, channel separately. E, G are enhanced image, ground truth image. Φ is the VGG19 operation.

For the region loss, we add larger weights for those low light region while smaller weights for those efficient light region. The region loss is similar to a weighted average of MSE given the condition of different lighting. We first transform the color image to gray scale image. Then define the top 40% darkest pixels as dark pixels and the others are bright pixels. Then calculate the region loss. The definition of region loss is:

$$Loss_{region} = w_{low} * \frac{1}{W_{low} * H_{low}}\sum_{W_{low}}\sum_{H_{low}}\left\|E_{dark_{pixel}} - G_{dark_{pixel}}\right\| +$$

$$w_{high} * \frac{1}{W_{high} * H_{high}} \sum_{W_{high}} \sum_{H_{high}} ||E_{bright_{pixel}} - G_{bright_{pixel}}||$$

As the result, the total loss is defined as:

$$Loss = Loss_{structure} + Loss_{context} + Loss_{region}$$

3.x.4.4. Fine tune

Since the network is trained in the synthetic dataset which is existed in real world, it may learn some false knowledge from the synthetic dataset. A real world dataset, called “Low light paired dataset (LOL)” is also used to fine tune the model. Here is the comparison between before fine tune and after fine tune.

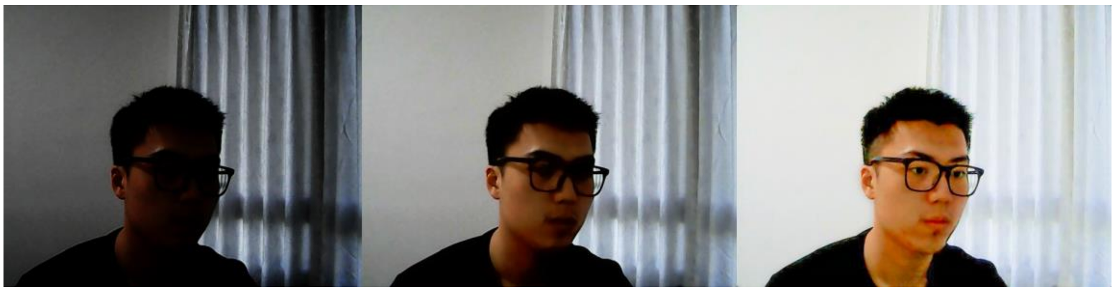


Fig. 3.23. Image in low light condition (left), generated image by without-finetune model (middle), generated image by finetuned model (right)

The subsequent SRGAN performance evaluations are all based on this finetuned model.

3.5.5. Super-resolution – super resolution GAN

In addition to the illumination condition, clarity is also an important indicator to represent image quality. If an image is in low resolution, you may need to resize your raw image to a specific size to feed it into a network. However, when we apply our WAM rule breaker detector into real world, due to different resolution of cameras and the distance between pedestrian and camera. The bounding box size of the pedestrian or face could be in very low resolution. It is difficult for the system to make decision and detect from a blur image. That’s why we introduce super-resolution into our project. GAN (Generative Adversarial Network) is powerful and here we applied an super resolution technique called SRGAN (super resolution GAN) to improve the image quality.

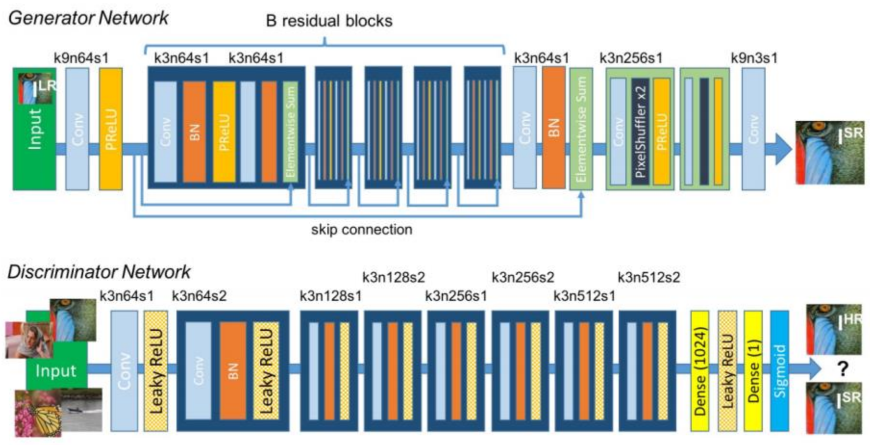


Fig. 3.24. The architecture of SRGAN

3.5.5.1. Architecture

As shown in Fig. 10., The top pipeline and network is the generator network, which is used to generate high resolution image from a low resolution image. The beginning of the network is a convolution layer with kernel size of 9x9 and activation function of PReLU (Parametric Rectified Linear Unit). This layer is mainly used to increase the number feature channels and improve the network fitting ability. The following module is a loop of residual blocks. For each residual block, it is composed by one convolution layer with batch normalization and activation, another convolution layer with batch normalization, and sum operation. After B residual blocks, we use one convolution layer to decrease the feature channels to 3 which is same to RGB image. Then two up-sample layers are applied to increase the resolution of image. Each up-sample layer will double the size of image so the size of output image is 4x the size of input image. The up-sample layer is called pixelShuffle layer. It first uses one convolution layer to increase the channels to four times as in the input data. Then periodic select the value in different channels to increase the size of feature map. The detail is shown in Fig. 3.25.

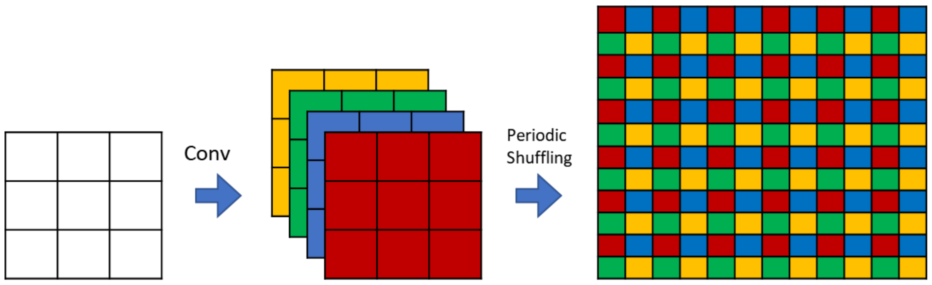


Fig. 3.25. What Up-sample layer inside. (Suppose the scale is 2, From 3x3 input to 12x12 output)

Since we get the generated high resolution image from the generator, how can we know quality of the output image? We build an adversarial network to force the output image to be very close to the ground truth image. The input of adversarial network is either a generated image or a ground truth image. If the input is generated image, then the output should be 0 or False (Not real image). If the input is ground truth image, then the output should be 1 or True (It's a real image). The adversarial network use a series of basic-convolution layers combined with stride-convolution layers to downsize the feature map and multiple linear perceptron to be the classifier. The output will be a probability of whether the input is real image.

3.5.5.2. Dataset

For super resolution, we use a dataset called DIV2K which contains both high resolution version and low resolution version of an image. The sizes of the images in DIV2K are different. To ensure the training process is stable and consistent, we randomly select an area of a training image with specific size of 96x96 at each training step.

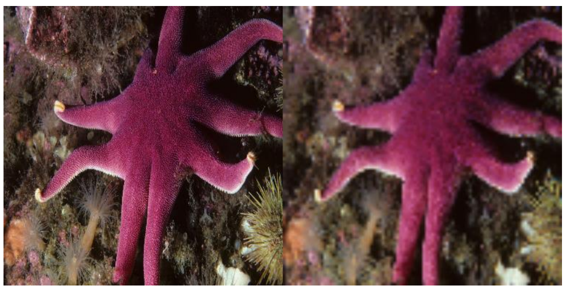


Fig. 3.26. One pair of HR image (left) and LR image (right) in DIV2K dataset

3.5.5.3. Loss function

For GAN, it is a two-stage training. The loss of generator is different from the loss of discriminator. That's because the targets of generator and discriminator are different. For generator, it needs to generate high resolution image as realistic as possible while for discriminator, it needs to correctly discriminate which image is real and which one is generated, so we design two different losses for two network.

Loss for generator: The total loss of generator is weighted sum of content loss and generator loss. To generate image as realistic as possible, we need content loss (Same to the content loss mentioned in MBLLEN network). To cheat the discriminator as much as possible, we need generator loss. The content is defined as:

$$Loss_{VGG19-layer10} = \frac{1}{W*H*C}\sum_{W}\sum_{H}\sum_{C}||\emptyset(E)_{pixel} - \emptyset(G)_{pixel}||$$

The generator loss is defined as:

$$Loss_{gen} = \sum_{n=1}^{N} -\log D\,(G(I^{LR}))$$

Where the D is the forward propagation of discriminator and G is the forward propagation of generator. I^{LR} is the low resolution image.

The total loss of generator is:

$$Loss = Loss_{content} + Loss_{gen}$$

Loss for discriminator: There is only one goal for discriminator: Correctly recognize which image is real and which image is generated, so we only have one loss for discriminator, which is defined as:

$$Loss = \frac{1}{N} * \left(\sum_{I \in Generated} logD(I) + \sum_{I \in Groundtruth} (1 - logD(I)) \right)$$

Where N is the total number of given images.

3.5.5.4. Fine tune

In our WAM rule-breaker detection project, we want to increase the resolution for those images where most of the pixels belong to human instead of background because it is totally a waste of computation resource. Therefore, we focus more on how to increase the resolution for a human image. However, the DIV2K dataset has lots of irrelevant image like animals or plants or other scenes. To solve this problem, I found a new dataset which contains only human face to fine tune the model. The subsequent SRGAN performance evaluations are all based on this finetuned model.

4. EXPERIMENTAL RESULTS

4.1 Face Detection module experimental results

4.1.1 Face bounding box and key point detection

For the face detection module, the bounding box and feature point detection are tested first, because correct face detection is the basis of face recognition. For the face detection module, first perform the bounding box and key points detection test, because correct face detection is the basis of face recognition. We use the LFW data set for face detection tests, and the test program outputs pictures marked with bounding boxes and key points. Some test results are as Fig. 4.1.

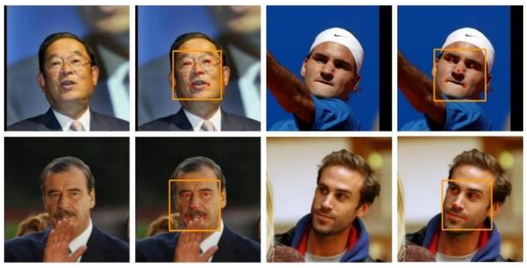
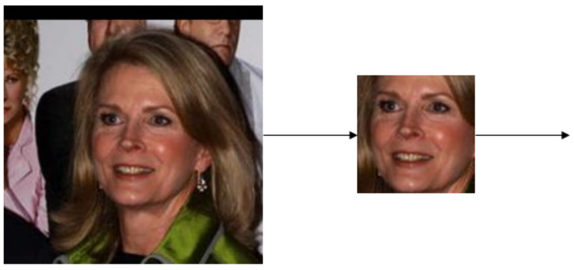


Fig. 4.1 Face detection test result

Data source: Dataset LFW

4.1.2 Face feature vector extraction and saving

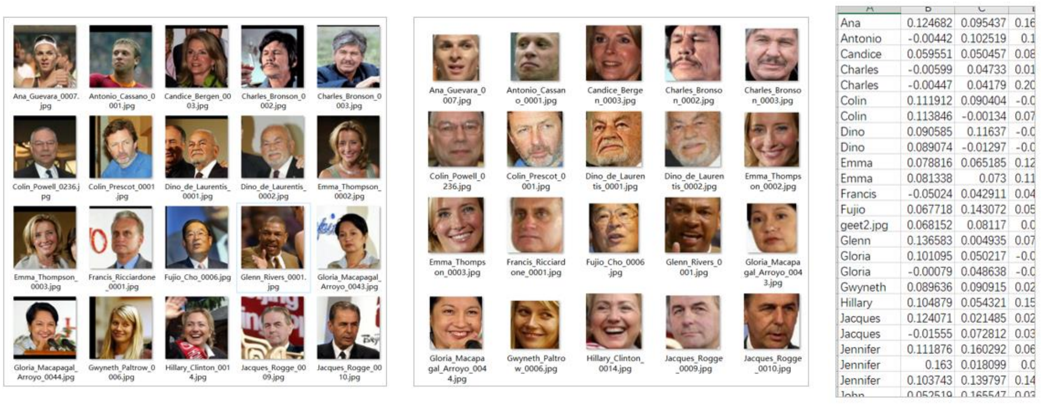
Face detection can extract the face image from the original image. On this basis, we also use a feature extraction network to extract 128-dimensional information of the image. The program execution results are as follows Fig. 4.2 and Fig. 4.3.



```
Candice 0.059551 0.050457 0.081996 0.135042 0.097386 0.140492 0.073097 -
0.11852 -0.01725 -0.094 -0.01247 0.037711 0.017887 -0.03393 0.102787 -0.07529
0.052821 0.014729 -0.03148 0.117817 0.008875 -0.01988 -0.06355 0.216357
0.081527 -0.15005 -0.22138 -0.06618 -0.01034 0.076073 0.090829 0.070339 -
0.06922 0.070037 0.059917 0.055633 0.067555 -0.06629 -0.00757 -0.10132
0.14471 -0.01304 0.066485 -0.17307 -0.05022 -0.00312 0.079928 0.030374 -
0.04322 0.017844 -0.05158 -0.11258 0.167293 -0.02511 0.09789 0.031724 -
0.11523 0.082503 -0.02268 -0.05488 -0.09374 0.188901 -0.01456 -0.23167
0.006683 0.205993 0.083211 -0.1048 -0.22067 -0.11627 0.013681 0.050688 -
0.03326 0.034517 0.104123 0.083546 0.020945 -0.07212 0.116674 0.079777
0.011344 0.086855 -0.01677 -0.05327 0.094299 0.071021 -0.04014 -0.01812 -
0.1023 0.098454 0.135451 -0.04781 0.105603 0.063354 -0.11728 -0.0341 -0.02479
-0.01058 0.025922 -0.03307 0.058221 -0.02354 -0.00713 0.006686 -0.13485
0.155654 0.055756 -0.02157 -0.05742 0.109772 0.053622 0.017521 0.011202 -
0.06451 -0.06171 0.003945 -0.1658 0.056478 -0.10737 0.073873 0.006826 -
0.03083 -0.0561 0.031611 -0.05688 0.159642 -0.01349 -0.10904
```

Fig. 4.2 One-person face feature extraction result

Data source: Dataset LFW



A	B	C	D
Ana	0.124682	0.095437	0.16
Antonio	-0.00442	0.102519	0.1
Candice	0.059551	0.050457	0.08
Charles	-0.00599	0.04733	0.01
Charles	-0.00447	0.04179	0.20
Colin	0.111912	0.090404	-0.0
Colin	0.113846	-0.00134	0.07
Dino	0.090585	0.11637	-0.0
Dino	0.089074	-0.01297	-0.0
Emma	0.078816	0.065185	0.12
Emma	0.081338	0.073	0.11
Francis	-0.05024	0.042911	0.04
Fujio	0.067718	0.143072	0.05
geet2.jpg	0.068152	0.08117	0.0
Glenn	0.136583	0.004935	0.07
Gloria	0.101095	0.050217	-0.0
Gloria	-0.00079	0.048638	-0.0
Gwyneth	0.089636	0.090915	0.02
Hillary	0.104879	0.054321	0.15
Jacques	0.124071	0.021485	0.02
Jacques	-0.01555	0.072812	0.03
Jennifer	0.111876	0.160292	0.06
Jennifer	0.163	0.018099	0.0
Jennifer	0.103743	0.139797	0.14
John	0.052519	0.165547	0.03

Fig. 4.3 Multiple person face feature extraction result

Data source: Dataset LFW

4.2 Face Recognition module experimental results

For the face recognition test, I input two photos of Person A at the same time as the positive sample group, and then multiple groups of Person A and other people's photos as the negative sample group. After calculating the distances of all sample groups, the partial results are as Fig 4.4.

We can see that the distance between the two images of the same person is the smallest, and the distance between the images of different people is larger.

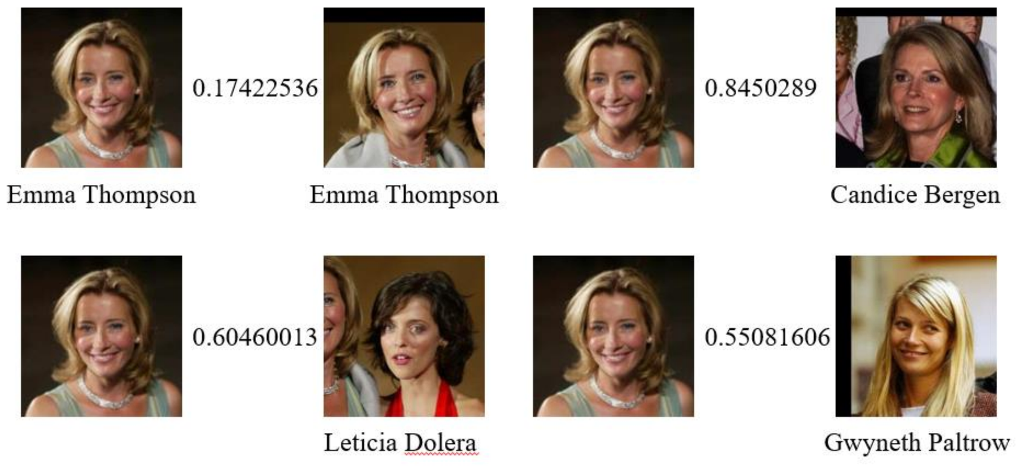


Fig. 4.4. Face recognition test result

Data source: Dataset LFW

4.3 Mask Detection

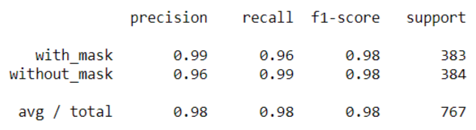
The mask detection was performed in various conditions such as low light , dim light , high illumination , marketplace , etc. The mask detection performed well in almost all scenarios. For the scenarios it did not perform well we implemented the image enhancement module . However, for mask detection we were able to successfully predict the Mask and No mask scenarios as well send an alert when no mask label was identified.

Even conditions such as printed masks were taken care of . So, in case a mask that is not in our training set like a really different printed mask . The model was able to identify the label with over 97% accuracy.

To evaluate the performance of the different classifiers, performance matrices are needed to be investigated through this work. The most common performance measures calculated are Precision, Recall, and F1 Score and they are presented as follows :.

The precision and recall is as follows :

Table 2 Performance Metrics



```
              precision    recall  f1-score   support

   with_mask       0.99      0.96      0.98       383
without_mask       0.96      0.99      0.98       384

 avg / total       0.98      0.98      0.98       767
```

Training loss and accuracy loss :

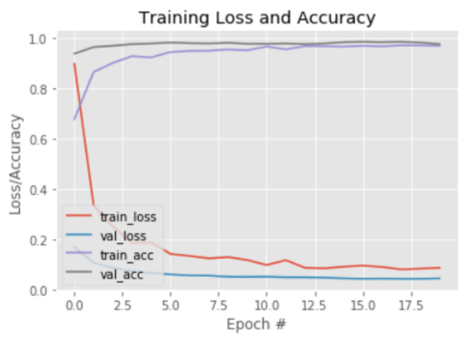


Fig. 4.5. Loss and accuracy curve

For a positive prediction ,i.e. , wearing a mask , we set the bounding box as Green (RGB) values as (0,255,0) .

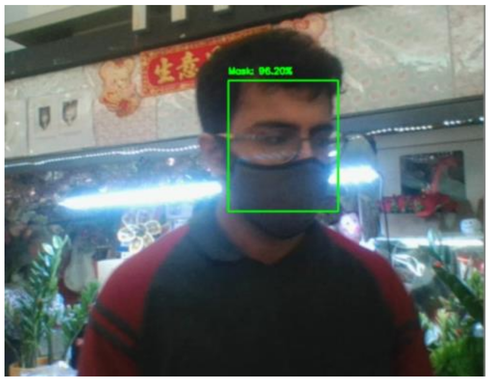


Fig. 4.6. Positive prediction

We also tested this scenario in case of face masks that are not the traditionally used face masks such as printed mask , like a face printed onto the mask which resembles close to an actual face . we managed to detect those masks as well .

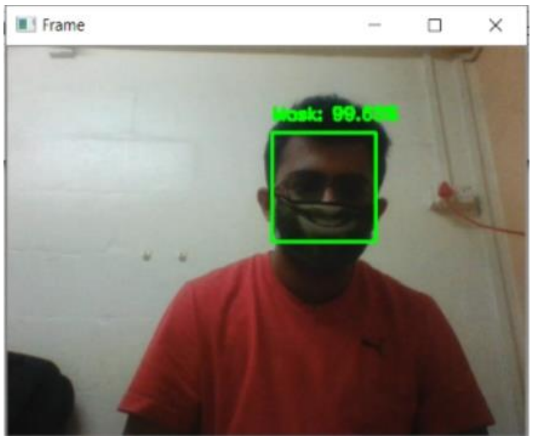


Fig. 4.7. Positive prediction

In case the person is detected as a rule breaker (not wearing a mask) , then we assign the red bounding box (0,0,255) .

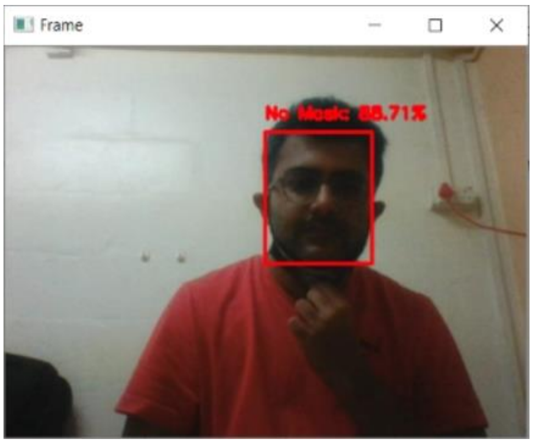


Fig. 4.8. Negative prediction

ALERT Notification

The next step is Alert Notification . In case a rule breaker is detected. So, if the label identified is “No Mask” , then we show a warning that a rule breaker is detected and some action needs to be taken .

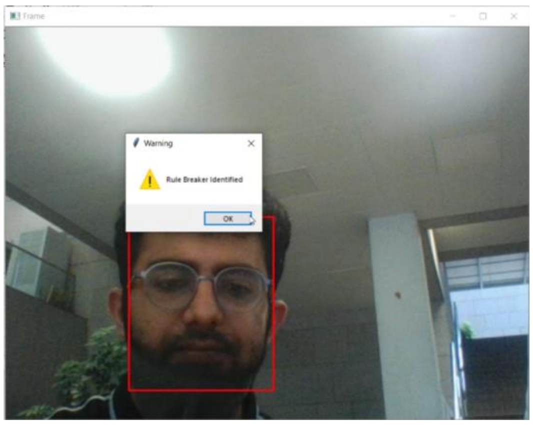
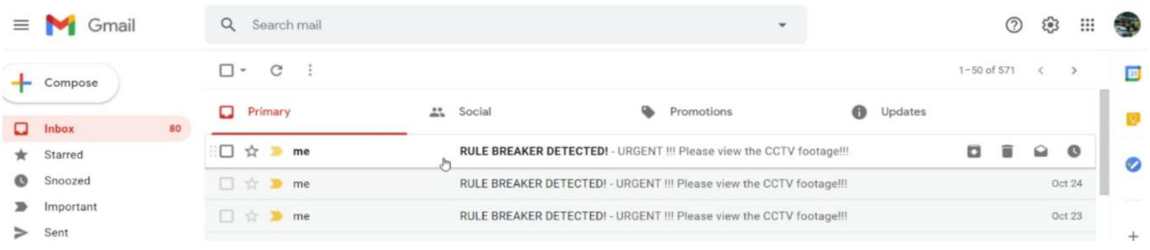


Fig. 4.9. Alert notification

Moreover , an email notification is sent stating that a rule breaker is detected and the CCTV footage must be checked .



me RULE BREAKER DETECTED! - URGENT !!! Please view the CCTV footage!!!
me RULE BREAKER DETECTED! - URGENT !!! Please view the CCTV footage!!! Oct 24
me RULE BREAKER DETECTED! - URGENT !!! Please view the CCTV footage!!! Oct 23

Fig. 4.10. Alert notification in Gmail

4.x. Image enhancement

4.x.1. Sharpening filter

Sharpening filter is used to enhance the edge of an object in the image. However, it is very difficult to justify the performance of sharpening filter. We can see the difference between with and without sharpening filter, which is shown in Fig. 4.11.

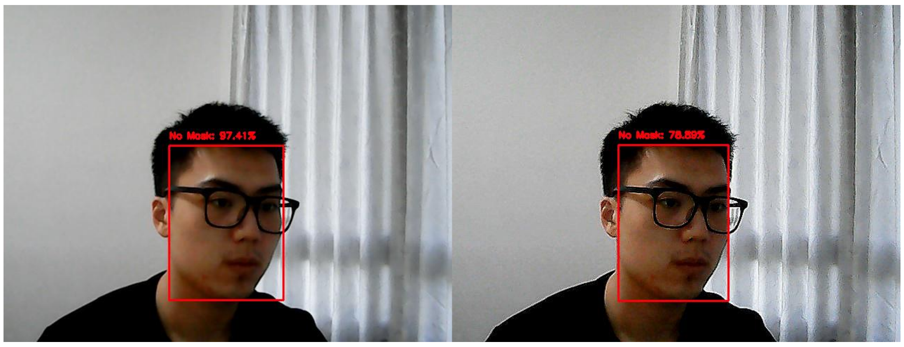


Fig. 4.11. Face detection on raw image (left), face detection on image with sharpening filter (right).

4.x.2. Denoising filter

Noise will have a great impact on image processing such as face detection. Two types of noise are added to our image to test the performance of our denoising filter. First is salt-and-pepper noise. Face detection is applied on both image with salt-and-pepper noise and image denoised by median filtering.

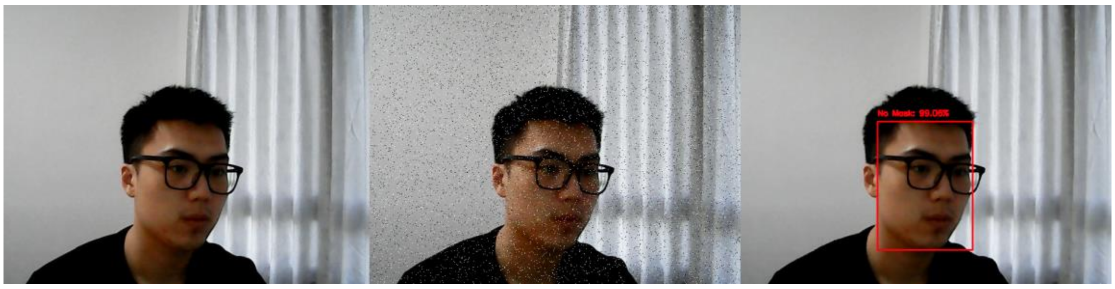


Fig. 4.12. Raw image (left), face detection on image with salt-and-pepper noise (middle), face detection on image denoised by median filtering.

From the Fig. 14. we can see that face detection becomes invalid when salt-and-pepper noise is added into the image. After the median filtering, even if the image becomes a little bit more blur, the face detector successfully detects the face and recognize the person does not wear a mask.

Second type of noise is gaussian noise. We manually add gaussian noise into our raw image and apply face detection on it. Then denoise the gaussian noise with average filtering and similarly apply face detection on it. The result is shown in Fig. 4.13.

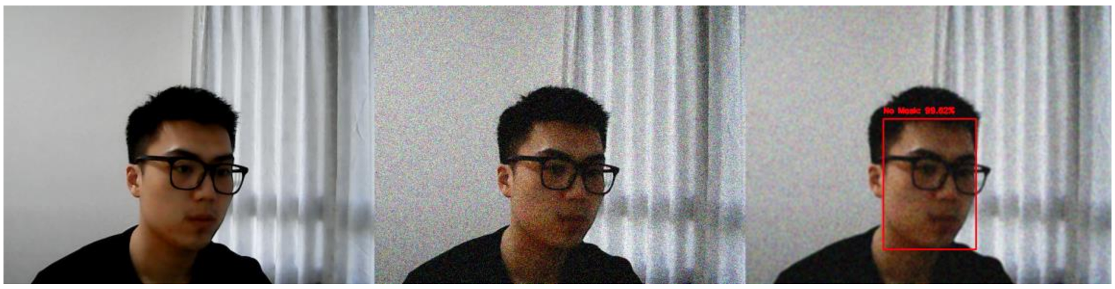


Fig. 4.13. Raw image (left), face detection on image with gaussian noise (middle), face detection on image denoised by average filtering

From the Fig. 4.13. we can see that face detection becomes invalid when gaussian noise is added into the image. After the average filtering, even if the image becomes a little bit more blur, the face detector successfully detects the face and recognize the person does not wear a mask.

4.x.3. Histogram equalization

To test the performance of histogram equalization, we apply gamma transformation on the raw image which is similar to the synthetic dataset mentioned in MBLLEN network. Then we utilize the histogram equalization to recover the image. We simulate in two different conditions (over-exposed and under-exposed). The result are shown in Fig. 4.14. and Fig. 4.15.

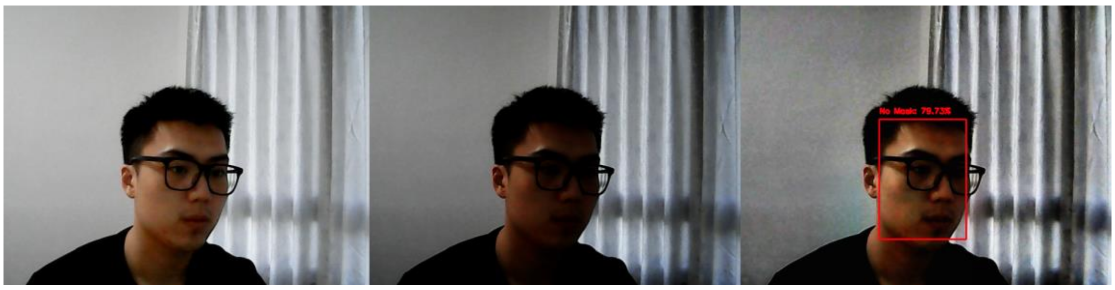


Fig. 4.14. Raw image (left), face detection on image in underexposed condition (middle), face detection on image with histogram equalization.

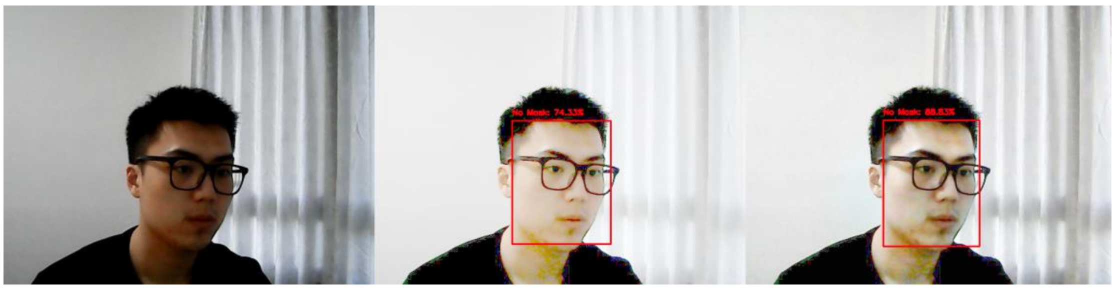


Fig. 4.15. Raw image (left), face detection on image in overexposed condition (middle), face detection on image with histogram equalization.

From the Fig. 4.14. and Fig. 4.15., we can see that the impact of underexpose is more severe than the overexpose. Generally, in indoor area like supermarket, it seldom happens to have low light condition (No matter daytime or nighttime). Therefore, we will not introduce advanced technique for those places to reduce the waste of computation resource.

4.x.4. MBLLEN

There are many different performance metrics to evaluate the performance of MBLLEN, including PSNR, SSIM, Average Brightness(AB), Visual Information Fidelity(VIF), Lightness order error(LOE), and TMQI. However, here we more focus on how it will influence the performance for face detection, i.e. Do it improve the performance of the face detection or other model being applied in this project? Here we use an extremer case to do the face detection. The gamma value is set to 3 and the performance of histogram equalization is shown in Fig. 4.16. We can see in extremely dark environment, histogram equalization is not useful anymore but with MBLLEN, we can successfully recover the image and detect the face, which is shown in Fig. 4.17.

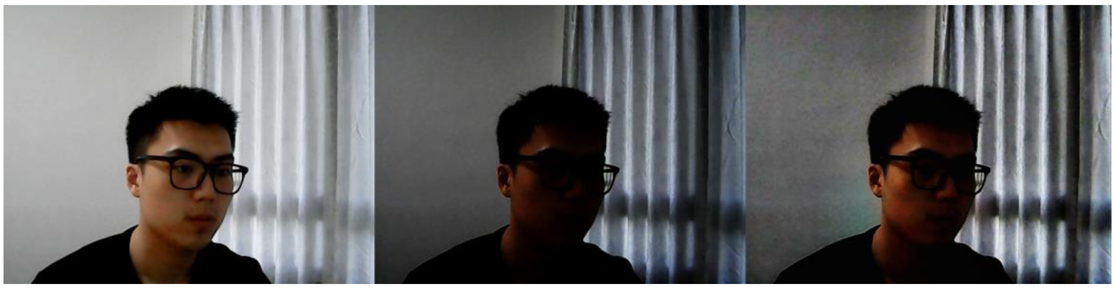


Fig. 4.16. Raw image (left), face detection on image in extremely dark condition (middle), face detection on image with histogram equalization (right).

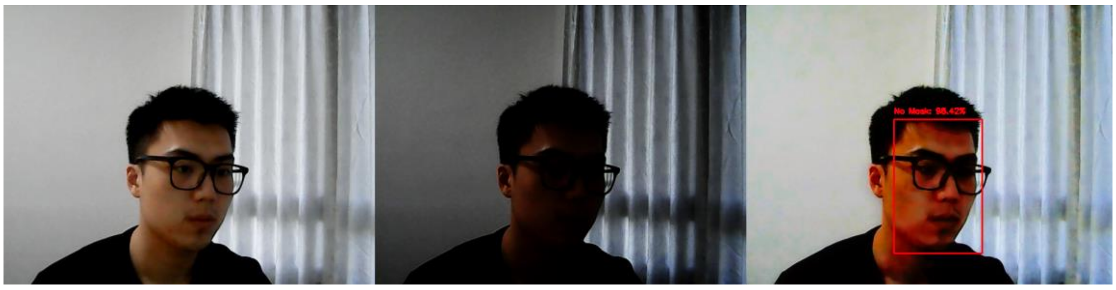


Fig. 4.17. Raw image (left), face detection on image in extremely dark condition (middle), face detection on image with MBLLEN (right).

As a result, we can also apply our system into outdoor area. Even the lighting condition is bad, we can recover the image and do other operations.

4.x.5. SRGAN

SRGAN is useful to recover the details of the image. There are some performance metrics to evaluate the performance of SRGAN, which have been mentioned in 4.x.4 section like PSNR and SSIM. In this project, we focus on the visual feeling of the generated image, which is shown in Fig. 4.18.

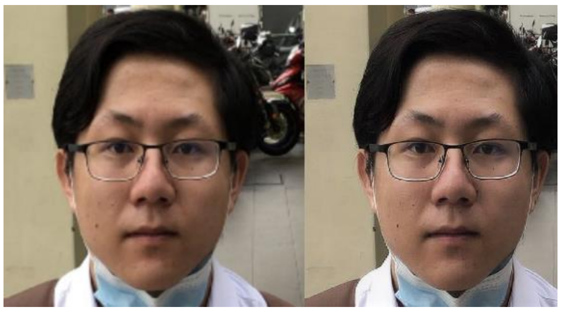


Fig. 4.18. Low resolution image (left), Generated high resolution (4x) image (right).

5. CONCLUSION

Our system can detect if one person is wearing a mask at very high accuracy. Besides, if we find someone takes off his / her mask, we can also do the reidentification. Compare the face feature captured by the camera and feature recorded in the database. Find out who breaks the rule and send an alert email to the safety officer. To make our system robust and can be applied to more extreme scenarios, image enhancement is used to eliminate the noise, improve the lighting condition and increase the image resolution. Advanced deep learning techniques are introduced in our project to make the system perform better. Although there also exists some limitations. The experimental result shows that it is possible to deploy our system to a real scenario for WAM rule-breaker detection.

Reference:

[1] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, Gang Hua. A convolutional neural network cascade for face detection. 2015, computer vision and pattern recognition

[2] Loey, Mohamed, et al. "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic." Measurement 167 (2020): 108288.

[3] Shuo Yang, Ping Luo, Chen Change Loy, Xiaoou Tang. Faceness-Net: Face Detection through Deep Facial Part Responses.

[4] Kaipeng Zhan, Zhanpeng Zhang, Zhifeng L, Yu Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. 2016, IEEE Signal Processing Letters.

[5] HR - P. Hu, D. Ramanan. Finding Tiny Faces. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[6] Face R-CNN - H. Wang, Z. Li, X. Ji, Y. Wang. Face R-CNN. arXiv preprint arXiv:1706.01061, 2017.

[7] SSH - M. Najibi, P. Samangouei, R. Chellappa, L. Davis. SSH: Single Stage Headless Face Detector. IEEE International Conference on Computer Vision (ICCV), 2017.

[8] PyramidBox - X. Tang, Daniel K. Du, Z. He, J. Liu PyramidBox: A Context-assisted Single Shot Face Detector. arXiv preprint arXiv:1803.07737, 2018.

[9] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1701-1708, doi: 10.1109/CVPR.2014.220.

[10] Zhu, Zhenyao, et al. "Recover Canonical-View Faces in the Wild with Deep Neural Networks." Eprint Arxiv (2014).

[11] Schroff, Florian, D. Kalenichenko, and J. Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) IEEE, 2015.

[12] Sun, Yi, et al. "DeepID3: Face Recognition with Very Deep Neural Networks." Computer ence (2015).

[13] Masi, Iacopo, et al. "Do We Really Need to Collect Millions of Faces for Effective Face Recognition?" European Conference on Computer Vision Springer, Cham, 2016.

[14] Tran, Anh Tuan, et al. "Regressing Robust and Discriminative 3D Morphable Models with a very, Deep Neural Network." (2016).

[15] Lichao Huang, Yi Yang, Yafeng Deng, Yinan Yu. DenseBox: Unifying Landmark Localization with End to End Object Detection. 2015, arXiv: Computer Vision and Pattern Recognition

[16] Lv F, Lu F, Wu J, et al. MBLLEN: Low-Light Image/Video Enhancement Using CNNs[C]//BMVC. 2018: 220.

[17] Ledig C, Theis L, Huszár F, et al. Photo-realistic single image super-resolution using a generative adversarial network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4681-4690.

[18] Gharbi M, Chen J, Barron J T, et al. Deep bilateral learning for real-time image enhancement[J]. ACM Transactions on Graphics (TOG), 2017, 36(4): 1-12.

[19] Dong C, Loy C C, He K, et al. Image super-resolution using deep convolutional networks[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 38(2): 295-307.