Face Recognition Based on Deep Learning and Data Augmentation

Lam Duc Vu Nguyen[1,2], Van Van Chau[1,2], and Sinh Van Nguyen[1,2(✉)]

[1] School of Computer Science and Engineering, International University, Ho Chi Minh City, Vietnam
[2] Vietnam National University, Ho Chi Minh City, Vietnam
nvsinh@hcmiu.edu.vn

Abstract. Face recognition is one of the most popular applications in video surveillance systems and computer vision. The researches of face recognition in recent years have been shown that their applications are widely used in practice. Particularly, during the pandemic of Covid-19, there were a lot of researches relating to face recognition with and without mask. The accuracy of the face recognition algorithms is depended on technical issues, implemented solutions and models of data processing. In this paper, we propose an improved method for face recognition based on deep learning techniques and data augmentation. Our contribution of the proposed method is focused on the following steps: (1) obtaining and pre-processing data for training dataset based on image processing techniques (i.e. noise removal, mask wearing). (2) Creating a trained model of new dataset based on the Inception Resnet-v1. (3) Building an application for face recognition in timekeeping of a company. We use the two popular face datasets which are open source and publicity available: Casia-WebFace [1] for training and LFW [2] for validation. Comparing the several methods, the accuracy of our method is higher in case with mask and the processing time is very fast in the real time.

Keywords: Face detection · Face recognition · Deep learning · CNN models · Inception-resnet data augmentation

1 Introduction

The research and application of computer vision, computer graphics, image processing and machine learning in recent years brings us a lot of advantages [3–6]. They are applied in many fields like 3D simulation, game industry, medical diagnostics or digital heritages. These knowledge and solutions are also popular and widely used in the systems of security, access control, check-in and check-out, objects tracking and monitoring, identify verification services, etc., and proved usefully and efficiently. The researches in face recognition system have been becoming popular along with development of artificial intelligence and deep learning techniques [7,8]. A facial recognition system is an application capable of matching a human face from a digital image or a video frame of a

T. K. Dang et al. (Eds.): FDSE 2022, CCIS 1688, pp. 560–573, 2022.
https://doi.org/10.1007/978-981-19-8069-5_38

camera with a given image in the database system. Using the machine learning methods and deep learning techniques are best solutions to obtain the high accuracy and adapt real time processing. Transfer learning and image augmentation are additional techniques in image processing and machine learning that can help improving precision of the face recognition systems.

The existing identify systems like fingerprints, palm recognition, ID and passwords, etc., are developed and widely used in practice. However, these methods existed limitations such as fake images of fingers and palms, or missing the user name and password to check-in. Therefore, the needs for using face recognition system is increasing recent years because of its accuracy and necessary in both security system and objects monitoring system.

In order to build a facial recognition system, the training dataset plays an important role in implementation of the system. Numerous facial datasets have been published online so that anyone can download them for free testing. These datasets are very huge, containing millions of images of various peoples. Therefore, researchers can skip the data collection phase and concentrate on training the model. The computer hardware is also getting more and more powerful to be able to train models. Wearing a face mask is now commonly used as part of standard to prevent infection during the pandemic of COVID-19. This is requirements for us in public spaces or on the means of transportation. Therefore, a system that recognizes people with face mask is necessary in any organizations or companies.

The system ought to be real-time and robotized. One suitable solution for these requirements is deep learning. This technology can help us recognize people with face mask automatically. Although the numerous facial datasets are available and free for accessing in the several researches. However, the facial dataset with mask is not available at present. In this research, we apply the image processing techniques to wear a mask on each face of the existing facial data. The dataset is then used for training data of our model. After reviewing the state-of-the-art methods, we propose a method for face recognition in both mask and without mask. Our contribution focus on data creation and data augmentation. The improved points have been shown the accuracy in face recognition with mask.

The remainder of the paper is structured as follows. Section 2 presents the state-of-the-art methods, several applications, tools and techniques for building application in practice. Section 3 describes in detail our proposed method including system design of the application. We present the implementation and obtained results in Sect. 6. Section 5 includes discussion and evaluation. The last section is our conclusion and future work.

2 Related Works

In this section, we explore the several methods for face recognition and its application in practice. The two important points that researchers want to obtain are accuracy and time processing. In order to improve accuracy of image classification, Alex Krizhevsky et al. [9] (called AlexNet) presented a new CNN architecture model based on increasing layers with the support of computational power.

The advantage of this CNN architecture is that the model can extract more features from each layer. After that, they are combined to enrich information of the object for prediction step. Moreover, the activation function ReLU is used as an additional computation step to speed-up the training process of the model compared to sigmoid function. However, the limitation come from size selection (11×11) of the first convolutional layer that can be difficult to deal with smaller size of pictures in practice. Karen et al. [10] proposed an idea to build templates using blocks. The VGG Block includes three parts: a convolutional layer, an activation function, and a pooling layer. The order and number of the block are optional. At the end, a fully connected layer and SoftMax is inserted to make prediction. Unlike AlexNet [9], that model use 3×3 filters to help reduce computation, which reduces time and parameters. In contrast, they increase the depth of CNN that lead to increase accuracy of the network model.

In the traditional approach, the neural nework layers are stacked together as thickly as possible to create a model that can learn complex rules. However, a very deep neural network can cause the problem of overfitting. Another problem is vanishing gradient: the gradient is too small, that make it impossible for layers to learn in backpropagation. Furthermore, since the picture's information area changes with each image, choosing the optimal kernel size is crucial. Szegedy et al. [11] suggested a new solution for building neural networks. Instead of being "deeper," the network would become "wider." Different kernels operate on the same input. Then, to form block output, all kernel's output are concatenated along the channel dimension. For example, a block includes four parallel paths. To extract information from different spatial sizes, the first three paths used different convolutional layers from 1×1, 3×3, and 55. The max pooling layer (3×3) is used in the last path. All output must be the same size across the fourth paths, which is a requirement for the concatenation step. Each layer must have appropriate padding. The neural network (as previously indicated) is consuming cost. Therefore, the authors include an extra layer 1×1 convolution before 3×3 and 5×5 convolutions to reduce the number of input channels.

Kaiming et al. [12] presented another solution (namely residual block) to defeat the vanishing/exploding gradient. Instead of making the network wider, this approach introduces a technique called skip connection. It let data skip several layers and connect directly to the output. The implementation of this block is simple. Input x and f(x) are directly added together to create an output of block. This kind of architecture requires x and f(x) to have the same shape. If the result x + f(x) approximates x (can be assumed as x + f (x) = x), the residual block is easy to learn (it means all weights and bias of the layer will be pushed to 0).

Christian et al. [11] presented a method for improving inception deep learning model for image classification. With numerous variants of the inception network were established, each of later versions is better compared to the previous one. In the next research project [13], authors suggested a solution to improve the accuracy and reduction of the computational complexity. Instead of using large filters (e.g. 5×5 or 7×7), it can be expensive in computation, authors suggested to decompose them into smaller filters.

Considering the spatial factorization, the researchers can continue to factorize n × n kernel into a combination of 1 × n and n × 1 kernels. When the number of filters is the same, 1 x n and n × 1 kernels are $(1 \times n + n \times 1)/n^2 = 2/n$ cost of n × n kernel. Therefore, this combination is $1 - 2/n$ cheaper than the original kernel. For instance, a 3×3 convolution is divided into 1×3 convolution followed by a 3×1 convolution. They found this method is $1 - 2/n = 0.3$ cheaper than the single 3×3 convolution. In practice, they figured out that this factorization does not work on early layer, but it produces incredibly good results on the grid of medium size. In order to reduce the grid size, Christian [13] use a pooling layer and a convolutional layer to process and combine them to produce the output of block. From these improvements, many architectural models had been introduced: Inceptionv1, Inceptionv2, Inceptionv3, Inceptionv4. The model in this research is based on Inception Resnet v1. The inception-resnet v1 is introduced in [14]. It is a hybrid network insprired by the inception model and residual model. This combination increases the number of layers while keeping the accuracy and performance.

Another research namely FaceNet is introduced by Florian et al. [15]. The purpose is to represent faces in Euclidean space, where the distance can used to compare similarity. The architecture of FaceNet is described as follows. A batch of images fed into a deep architecture (this architecture is designed to turn images into vectors). These vectors are then normalized to unit vectors using the L2 method. The pictures (as vector form) is then processed through triplet loss to distributed embedding the notation of similarity and dissimilarity. To the face-mask recognition, Warot Moungsouy et al. [16,17] proposed a method based on residual inception networks. Authors introduced a masked-face dataset based on the Casia-WebFace dataset [1]. It consists of 2236161 masked-face images. Then, both Casia-WebFace dataset and new dataset are combined to train the model. The proposed method was based on FaceNet using Inception-Resnet-v1 architecture. They test with several model and the best model was the fine-tuned FaceNet with the retrain Inception Block A on the new dataset. This model achieved 99.2% accuracy on masked-face test dataset. They also figure out that adding masked-face image into training data, it improve the accuracy of the model 0.6%

Besides, the transfer learning is known as a machine learning form where a model is built for a specific task and then reused on a second task as the starting point to be modified. It is used in deep learning as a pre-trained model in computer vision and natural language processing tasks to develop neural network models on these problems. The transfer learning is very useful in deep learning problems because most real-world problems usually have billions of labeled data, and this requires complex models. It is an effective technique for optimization, time saving and achieving better performance. Developers can use transfer learning to merge different applications into one. They can quickly train new models for complex applications. Moreover, transfer learning is a good tool to improve the accuracy of computer vision models. At the end of this research work, we present a facial recognition application for attendance system based on a deep

learning model. We utilize transfer learning by using three pre-trained convolution neural networks and train them on our data which contains 10 different classes where each class includes 20 facial images. The three networks showed very high performance in terms of high prediction accuracy and reasonable training time. Therefore, face recognition based on deep learning can greatly improve the recognition speed and accuracy. The last and not lead, many existing API, Libraries, tools and techniques can help us to implement our application like OpenCV, TensorFlow, PyTorch.

3 Our Proposed Method

3.1 Overview

This section present our proposed method. We create an effective model for both recognizing masked faces and without masked faces. As mentioned in the Introduction, the method consist of three steps. In the first step, we obtain the datasets from the publish sources Casia-WebFace [1] and LFW [2]. They are the face images without face masks. Therefore, after pre-processing these data (i.e. noise removal) we use the image processing techniques to wear a mask on each face. In the second step, we use the new datasets for building our training data model based on inception resnet-v1. The last step is creating an application for testing our face recognition system. The detail of each step is described in the following sections.

3.2 Pre-processing Data

Cleaning Data: The dataset Casia-WebFace [1] is used for face verification and face identification tasks including 494414 face images of 10575 peoples. Each image has a size of $250 \times 250 \times 3$ (width $\times$ height $\times$ channel) and saved as a jpg file. To avoid any noise in the image, an algorithm iterates through all images and detects face coordinates. Thus, the face in the image will be extracted from the background. In this section, the MTCNN model is used for face detection. This model includes three nets: P-Net, R-Net and O-Net to process and obtain the coordinates of the bounding box as a rectangle. Finally, we apply an image processing method to format, resize image to have a uniform of width and height, change the color to RGB and save image into a new folder (see Fig. 1). The algorithm (Algorithm 1) below is proposed to clear data.

Creating Mask: After cleaning process, a new dataset of faces is created, however these faces do not have masks. Thus, we now create a mask on each face image. Firstly, a list of mask images will be retrieved from the internet that contain ten different mask images. All of them are processed by using an edited photo application to filter out the background. The purpose is to randomly select masks, thereby creating diverse images when feeding them into the model. The functions of application allows process the picture. Using a certain threshold to

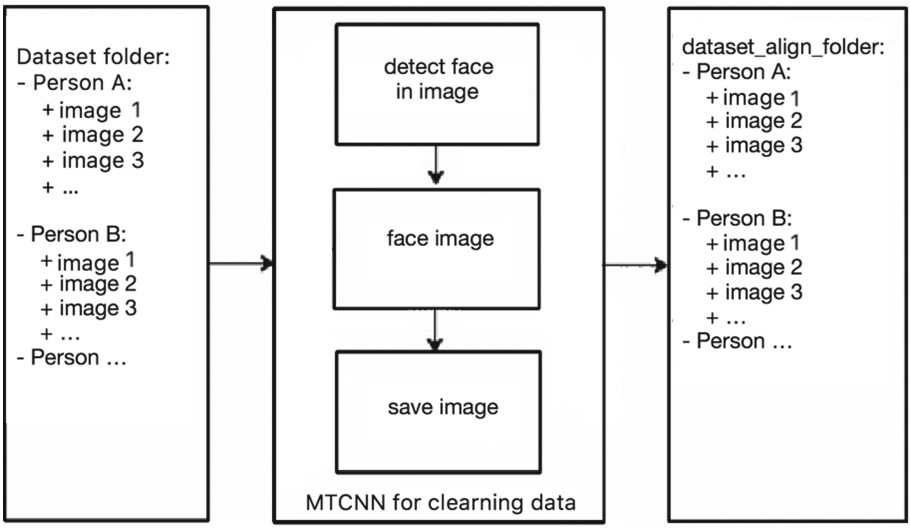

Fig. 1. Using MTCNN to clean data

Algorithm 1. Algorithm for clearing dataset

Input: the dataset from Casia-WebFace [1]
Output: Our new dataset

```
for each image in the dataset do
    image = openImage(each_file)
    list_face_coordinates = MTCNN.detect_faces(image)
    if length(list_face_coordinates) == 1 then
        [xmin, ymin, xmax, ymax] = list_face_coordinates[0]
        face_image = image[ymin:ymax, xmin:xmax]
        face_image = resize(image)
        save(face_image)
    else
        print(the number of faces is incorrect)
        continue
    end if
end for
```

enable the mask separated from the background and obtain a mask image. In order to determine the coordinates points on the face for processing, a technique called facial landmark [18,19] is used. This technique will mark important points of the face according to sixty-eight coordinates. We based on this information to wear mask on the face. Many solutions can be applied to cover a mask on the face like determining points around mouth, nose or chin, etc.

In this case, we map the mask to corresponding points on the face using the homography algorithm. Each face has different features and situation, applying this algorithm gives us a better result because the mask is rotated and resized

based on the shape of face. We want to overlay both cheeks and nose with a mask, so the corresponding points in this case from 1 to 15 and 29. Figure 2 illustrates how the points on the face and the mask are connected. If the coordinates of the face change, the coordinates of the mask also change, so the mask will fit more close to the face (see Fig. 2). The next algorithm (Algorithm 2) show the way to wear a mask on the face:

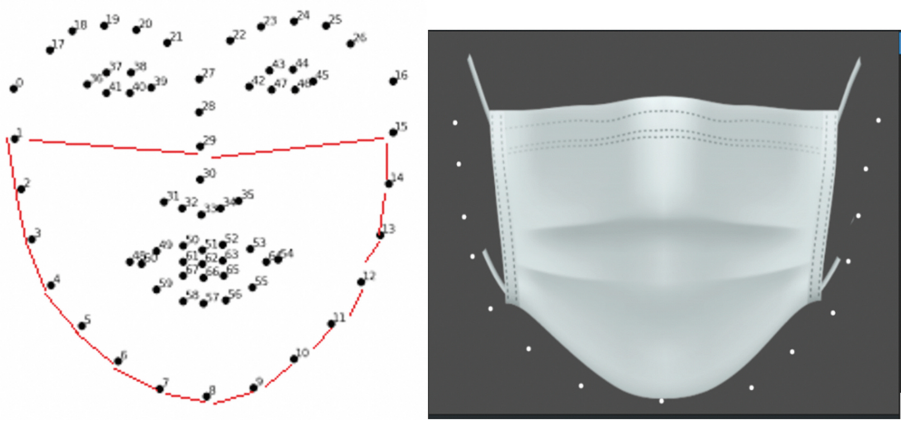

Fig. 2. Creating a mask for the face

3.3 Creating Trained Model

In this section, we apply the FaceNet concept to create our model. As mentioned in [15], the picture is converted into 128-D embedding using triplet loss function. In the very first model, triplet loss function is used to train the model, but the result is bad. The loss value decreases gradually and the model eventually corrupts. The problems come from batch size and the label of each batch. The Casia-WebFace dataset has 10575 labels with a batch size of 512. Therefore, there is a small chance for images of the same person to exist in one batch, while triplet loss function requires triplets (anchor, positive, and negative). For this reason, we apply the FaceNet concept to convert image into embedding; but other factors like loss function and model architecture will be changed. After training, we will remove the last layer (dense net 10575) and add an l2 normalizer to the model. The final model for embedding is described as in Fig. 3. We use the Inception-resnet-v1 [14] to train our model. The model is added more Batch Normalization Layers to normalize input data and modify the last layer to get the 128-element embedding. This model is a stack of Stem, Inception-Resnet-A, Reduction A, Inception-Resnet-B, Reduction-B, Inception-Resnet-C. The cross-entropy is chosen to train the model, so the last layer must encode the image into an n-number vector, where n is equal to the number of people in the dataset.

Algorithm 2. Algorithm for wearing a mask on the face

Input: The data without mask
Output: The data with mask

```
for each face image in the dataset do
    image = openImage(each_file)
    xminM, xmaxM, yminM, ymaxM //M: mouth
    landmarks = detect_68_landmarks_and_mouth(image)
    if (landmarks is not null) and (len(landmarks) = 68) then
        coord_points_face = ([point[0], point[1]] for points in landmarks[1:16])
        coord_points_face.append(landmarks[29][0], landmarks[29][1])
        mask_image = choose_a_random_mask()
        coord_points_mask = get_coord_points_mask()
        matrix_HG = findHG(coord_points_mask, coord_points_face)
        mask_image_HG = cv2.warpPerspective(mask_image,matrix_HG,imageSize)
        image_with_mask = wear_mask_HG(image, mask_image_HG)
        save(image_with_mask)
    else
        mask_image = choose_a_random_mask()
        mask_image = resize(xmaxM - xminM, ymaxM - yminM)
        image&mask = wear_mask_default(image, mask_image, xminM, yminM)
        save(image&mask)
    end if
end for
```

Due to the requirement of cross entropy, dense net has 10575 units before going to loss function because there are 10575 different persons in the dataset and one image belongs to only one person. But we do not want to classify 10575 people in a dataset or train again each time we add a new person. Therefore, we keep a dense net with 128 units to learn the features of the face before going to a dense net of 10575 (see Fig. 4).

Training Process: Although the dataset has 10575 people, but the image number of each person are not the same that lead to generate noise. To solve this problem, for each epoch, a training dataset must be created in which each person has the same number of images. After that, each person includes fifteen images are chosen from the "no mask" dataset and fifteen images are chosen from the "mask" dataset. We apply processing techniques to augment the data. This will help the model predict better in different conditions (e.g. angle, light, size). Because the data is processed at random, we make a double of data to increase the amount of data in one epoch. Currently, we have 634500 ($30 \times 10575 \times 2$) images in the training dataset. We use several functions like change contrast, change brightness, flip, and crop an image, then combine them together to generate new images.

Prediction Process: This model was inspired by the FaceNet model. It means the model learns to distinguish between different people and group the images

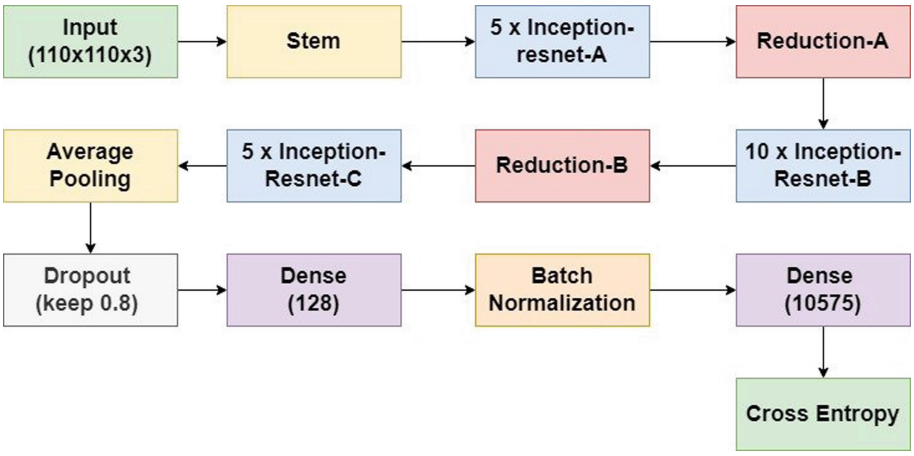

Fig. 3. Stem block

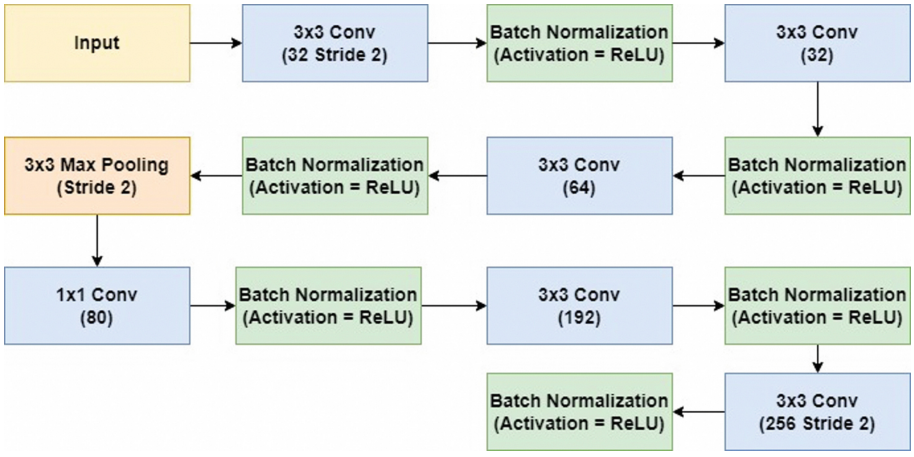

Fig. 4. Our proposed architectural model

of the same people into the same cluster. This approach is more efficient and we do not need to train our model again each time a new image is added. In detail, we generate an embedding of a new person and save it. To recognize an image, the model converts the image to embedding and uses a function to measure the similarity of this embedding with database embedding. The obtained result is a pair with the most similarities.

3.4 Building an Application

In this section, we build an application for testing our model. It is served as a real application to test and evaluate our model. Our application is performed

Fig. 5. Our application for testing and evaluating

on a single Laptop (MacBook Pro 8-Core CPU, 14-Core GPU) and its camera. The user interface of the application is created as follows (see Fig. 5).

4 Implementation and Results

In this section, we implement our proposed method and application for face recognition system. We perform our method by using Python, TensorFlow. The model and data are trained on Google Collab Pro. The application is then built based on our model to describe how it is used in the face recognition system. We implement all functions based on Python API to process data, to augment images, to train the training data. The Inception-Resnet v1 is implemented using Keras. To wear a mask on each face image, we use the API "FacialMask-DataSet.py". The output of this step is presented as follows: The face detection is performed by using existing source code in [20]. The obtained result is matched with data in the database (see Fig. 6 and Fig. 7)

Fig. 6. Wearing a mask on each face image [1]

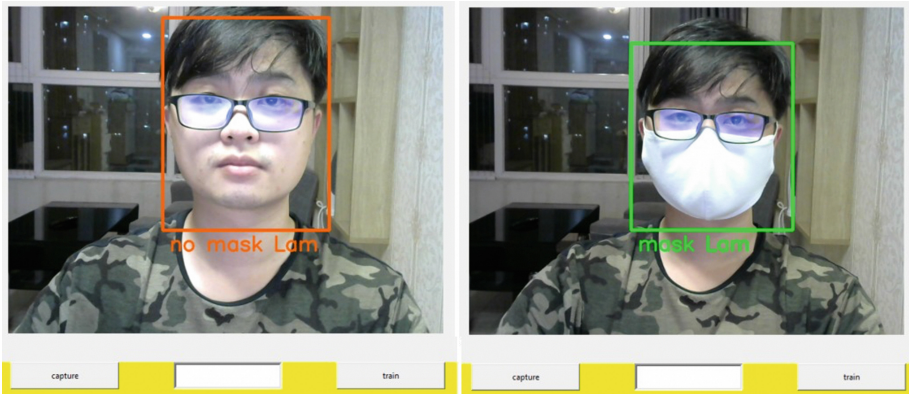

Fig. 7. Face recognition is worked well with both masked face and non-masked face

5 Discussion and Evaluation

In order to evaluate our model, we test and run with many times of epochs. After running 49 epochs (see Fig. 8a), The loss value decrease dramatically in the first ten epochs; after that, this value decreases slowly and is stable from 35 to 49. The goal is to evaluate how our model can process both the face with and without mask. We combine the "no mask LFW" and "mask LFW" into one dataset. The obtained result is plotted in Fig. 8b. The accuracy increases after each epoch. Starting from the epoch twenty, accuracy increase insignificantly, staying around 90%.

To compare with the existing method, the FaceNet-PyTorch [21] in python library includes several versions of the FaceNet model. In this research work, we use the Casia-Webface version. This model is trained with the same dataset that we used. The dataset LFW and its variants will be reused in this experiment.

Our model will be the model at epoch 49, which is the latest model. At first, two models will convert images from the "no mask LFW" dataset into database embedding. Each label will have only one embedding. Then two models will predict data in two cases: LFW (with mask) and LFW (without mask). In this

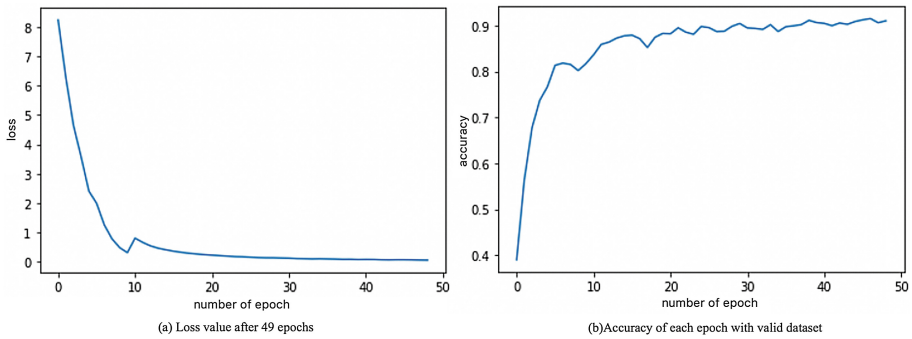

Fig. 8. Evaluation of the stable and accuracy of our model

test we do not use a threshold to maximize the accuracy of two models, so the unknown answer will be zero (see Table 1).

Table 1. Comparison of the precision, recall, accuracy and F1 between our model and the existing methods

Model (training dataset)	Data type	Precision	Recall	Accuracy	F1
Facenet pytorch (Casia-Webface)	Without mask	0.9875	0.9961	0.9758	0.9918
	With mask	0.5860	0.1697	0.1632	0.2632
Facenet pytorch (VGGFace2)	Without mask	**0.9969**	**0.9993**	**0.9957**	**0.9981**
	With mask	0.7024	0.4844	0.4772	0.5734
Our model	Without mask	0.9740	0.9896	0.9268	0.9817
	With mask	**0.9644**	**0.9817**	**0.8992**	**0.9730**

Comparing to other models, the accuracy of our model is better in case with mask and a little lower in case without mask. In order to improve for both cases (with and without mask), we can increase the number of epochs to run until meet the accuracy of our expectation.

6 Conclusion and Future Work

In this research work, we explore the several methods for face recognition and their application in practice. The methods are based on machine learning techniques are very popular and suitable for almost cases in practical applications nowadays. We proposed a method that is based on the Inception Resnet-v1 to implement our model. The obtained results have been shown the accuracy of our model (see Table 1). It is higher than the existing methods in case with mask. It can be adapted with real context of pandemic of Covid-19 at present. Besides, we built successful an application that will be applied in the company for timekeeping. This application can help the company to check-in, check-out

and control their staffs every days. The work in the future can be extended an applied in other security systems as mentioned in the Introduction.

References

1. Yi, D., Lei, Z., Liao, S., Li, S.Z.: Learning face representation from scratch (2014). https://arxiv.org/abs/1411.7923
2. Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. In: Workshop on Faces in Real Life Images: Detection, Alignment, and Recognition (2008)
3. Nguyen, S.V., Tran, H.M., Maleszka, M.: Geometric modeling: background for processing the 3D objects. Appl. Intell. **51**(8), 6182–6201 (2021). ISSN: 1573–7497
4. Van Nguyen, S., Le, S.T., Tran, M.K., Tran, H.M.: Reconstruction of 3D digital heritage objects for VR and AR applications. J. Inf. Telecommun. **6**(3), 254–269 (2021). https://doi.org/10.1080/24751839.2021.2008133. ISSN: 2475–1839
5. Van Nguyen, S., Nguyen, D.A., Pham, L.Q.S.: Digitalization of administrative documents - a digital transformation step in practice. In: 8th NAFOSTED Conference on Information and Computer Science (NICS), pp. 519–524. IEEE (2021). 978-1-6654-1001-4/21/$31.00
6. Van Nguyen, S., Tran, H.M., Le, T.S.: Application of geometric modeling in visualizing the medical image dataset. SN Comput. Sci. **1**(5), 1–15 (2020). https://doi.org/10.1007/s42979-020-00266-0
7. Suganthi, S.T., Ayoobkhan, M.U.A., Venkatachalam, K.V., Bacanin, N., Stepan, H., Pavel, T.: Deep learning model for deep fake face recognition and detection. PeerJ Comput. Sci. **8**, e881 (2022). https://doi.org/10.7717/peerj-cs.881
8. Teoh, K.H., Ismail, R.C., Naziri, S.Z.M., Hussin, R., Isa, M.N.M., Basir, M.S.S.M.: Face recognition and identification using deep learning approach. J. Phys. Conf. Ser. **1755**, 012006 (2021)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional. J. Commun. ACM **60**(6), 84–90 (2017). https://doi.org/10.1145/3065386
10. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015). http://arxiv.org/abs/1409.1556
11. Szegedy, C., et al.: Going deeper with convolutions (2014). https://doi.org/10.48550/arXiv.1409.4842
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), pp. 770–778 (2016)
13. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016). https://doi.org/10.1109/CVPR.2016.308
14. Szegedy, C., Ioffe, S., Vanhouke, V., Alemi, A.A.: Inception-v4, inception-ResNet and the impact of residual connections on learning. In: AAAI 2017 Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, pp. 4278–4284 (2017)
15. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, pp. 815–823 (2015). https://doi.org/10.1109/CVPR.2015.7298682

16. Moungsouy, W., Tawanbunjerd, T., Liamsomboon, N., Kusakunniran, W.: Face recognition under mask-wearing based on residual inception networks. Appl. Comput. Inf. (2022). https://doi.org/10.1108/ACI-09-2021-0256
17. Masood, S., Ahsan, U., Munawwar, F., Rizvi, D.R., Ahmed, M.: Scene recognition from image using convolutional neural network. J. Procedia Comput. Sci. **167**, 1005–1012. https://doi.org/10.1016/j.procs.2020.03.400. ISSN 1877–0509
18. Sun, K., et al.: High-resolution representations for labeling pixels and regions (2019). arXiv:1904.04514v1 [cs.CV]
19. Liang, S., Zhou, Z., Guo, Y., Gao, X., Zhang, J., Bao, H.: Facial landmark disentangled network with variational autoencoder. J. Appl. Math. **37**(2), 290–305 (2022)
20. AIZOOTECH. Github FaceMaskDetection. Accessed July 2022. https://github.com/AIZOOTech/FaceMaskDetection
21. Timesler, Facenet-pytorch. https://github.com/timesler/facenet-pytorch