Are Current Face Recognition Models Sufficient for Masked-Face Recognition ?

Muhammed Naci Dalkıran

Abstract

Face recognition, which is becoming widespread and gaining importance gradually, is one of the application areas of artificial intelligence. Face recognition systems are used in many places from public institutions to private organizations. However, due to pandemic, Covid 19, people wear masks and masked-face may not be recognized by current face recognition models. In this paper, I will investigate whether current face recognition models are successful to recognize masked-face or not.

1. Introduction

Face recognition is a method to identify or classify a person using individuals' faces. Face recognition systems have been utilized since the 1960s [1]. When it was first started to be used, it was not sufficient since you should manually determine where the mount, eye, etc. are [1]. This approach for face recognition seems to be primitive; however, it contributed ideas, namely finding face's organs automatically. In the 1970s, the face recognition systems used algebra to find the face's organs and recognize it. In the 2010s, people were more interactive in social media and they met 'selfie'. This circumstance contributes to building and accumulating very large face datasets. After the 2010s, face recognition systems used big dataset and recognized faces with high accuracy. Before pandemic, face recognition systems with artificial intelligence were used in different places and areas, such as in airlines, finding criminals and so on. The recent face recognition models are based on neural networks, generally convolutional neural networks are used in face recognition models. However, before the neural network process, faces should be detected and features of the faces should be determined. Actually, face recognition models generally follow this order : face detection, face alignment, extracting feature, and classification.

In the face detection part, the algorithm analyzes the image and controls whether there is a face(s) or not. Algorithms basically control that there are eyes, mount, nose, etc. If there are face's organs, it looks at the ratio among them and it detects face(s) using this ratio.

Fig. 1: Face Detection Example [2]

After face detection, detected face(s) should be aligned. The face alignment is a method which identifies the geometric structure of detected face(s). The method automatically determines the size and shape of a face's organs, components, namely mount, eyes, nose etc.

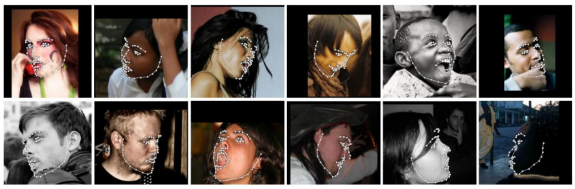


Fig. 2: Face Alignment Examples [3]

After face alignment, features of faces should be extracted. There are different approaches for the feature extraction of face. However, the main aim of the approaches is the reducing computational complexity by representing face's feature as a vector or creating a pattern of face features such as size of eyes, ratio of face components. Also, in this part, the name of the person can be encoded as a number or stored with representation of the image.

After extracting features of the face, faces should be classified to identify the person. classification of faces can be processed by different neural network models. Generally, Convolutional Neural Network (CNN) or its derivatives are used to classify faces for face recognition. Finally, faces are classified and identified according to the input dataset and faces.

These methods have a high accuracy in face recognition for unmasked, normal faces. However, nowadays, there is a pandemic due to SARS - Cov - 2 virus. Because of the pandemic, people have to wear masks to protect themself to not be infected; therefore, current face recognition models might not be successful or have not high accuracy during the face recognition process. In this paper, I will focus on current face recognition models and investigate success of these models in masked-face recognition.

2. Current Face Recognition Models

In this section, I will examine three current face recognition models : Face Recognition with FaceNet Encoding, Face Recognition with FaceNet Embedding and Support Vector Classification (SVC) Models, Face Recognition with DeepFace Ensemble Learning.

2.1. Face Recognition with FaceNet Encoding

The model is open source model; source code and GitHub page link[1]. The pipeline of the model is: Face Detection, Face Alignment, Face Encoding, and Face Classification. I will explain each part of the pipeline below.

The first part of the pipeline is face detection. To detect face, Multi-task Cascaded Convolutional Neural Networks (MTCNN)[2] model is used. MTCNN has three convolutional neural networks which are P-Net, R-Net, and O-Net. With these three CNN, MTCNN model displays faster and more accurate performance.

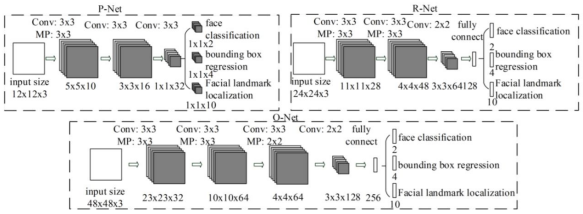


Fig. 3: Structure of CNN models of MTCNN [4]

Group	CNN	300 Times Forward	Accuracy
Group1	12-Net [19]	0.038s	94.4%
Group1	P-Net	0.031s	94.6%
Group2	24-Net [19]	0.738s	95.1%
Group2	R-Net	0.458s	95.4%
Group3	48-Net [19]	3.577s	93.2%
Group3	O-Net	1.347s	95.4%

Fig. 4: Comparison of Speed and Validation Accuracy of CNNs in MTCNN and Previous CNNs [4]

The second part of the pipeline is face alignment. As mentioned before, different face components such as eyes, mount, etc. can be used for face alignment. To align faces, the model uses eyes lines.

The third part of the pipeline is face encoding. In this part, face encodings are extracted from the source image and this process is repeated for each image. The result of this process is 512 encoding, bounding box image(s).

The last part of the pipeline is to classify the faces. In this part, to classify faces, euclidean distance method are used with face encodings. In the previous part of the pipeline, there is a vector output named encodings. This vector actually represents 512 features of the image. The euclidean distance is used to calculate how two image encodings vectors are close to each other. If the output of this calculation is smaller than threshold value, the algorithm accepts these two faces belonging to the same person. The value of threshold is none as a default; but, it can be set.

$$d_{\mathrm{E}}^2(x, y) = \sum_{k=1}^{IJ} \left(x^k - y^k\right)^2.$$

Fig. 5: The Formula of Euclidean Distance

2.2. Face Recognition with FaceNet Embedding and Support Vector Classification (SVC) Models

The model is open source model; source code[3] and tutorial link[4]. The pipeline of the model is: Extract Faces, Face Embeddings, Label Encoding, Face Classification. I will explain each part of the pipeline below.

The first part of the pipeline is the extraction of faces from the image. To extract face(s) from an image, MTCNN is used. The result of MTCNN is a list of bounding box of face(s). To make faces suitable to train them, faces are resized. The required size for this model is 160*160.

[1] https://github.com/paoloripamonti/face-recognition

[2] https://github.com/ipazc/mtcnn

[3] Brownlee, Jason. Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python. Machine Learning Mastery, 2019.

[4] https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/

The second part of the pipeline is face embeddings. Face embeddings are vectors of the faces. These vectors are created according to the features of the faces. Actually, faces represent a feature vector with a face embedding process. To create face embeddings, predict method of FaceNet model is used. Thanks to face embedding process, each face is represented as one dimensional array, size of 128. The size of this array is actually a representation of 128 different features of the face. After the face embedding process, the value of embeddings vectors are rescaled in range one to zero to use distance metric and prepare data for classification.

The third part of the pipeline is label encoding. In this part, all labels, namely name of person in the image, are encoded with a number. This process provides us with opportunities to access the label easily since all names might not be known and tracked; but integer values can.

The last part of the pipeline is to classify the faces. To classify and identify faces, Support Vector Classification (SVC) models are used. In SVC[5], different kernels can be used for multiclass classification. These kernels are : linear, polynomial, radial basis function (rbf). The kernel is a method or function which is used for linear classification for nonlinear problems or classification. As a default rbf are used; however, in the model, linear is used to make classification process faster since rbf works more slowly.

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

Fig. 6: The Formula of RBF Kernel [5]

2.3. Face Recognition with DeepFace Ensemble Learning

The model is open source model; source code and GitHub page link[6]. The pipeline of the model is: Face Detection, Face Alignment, Face Representation, and Face Verification. I will explain each part of the pipeline below.

The first part of the pipeline is face detection. To detect faces, the model utilizes a neural network called ResNet Single Shot-Multibox Detector (SSD) with OpenCV library [6]. The basis of the ResNet SSD is VGG which will be explained later.

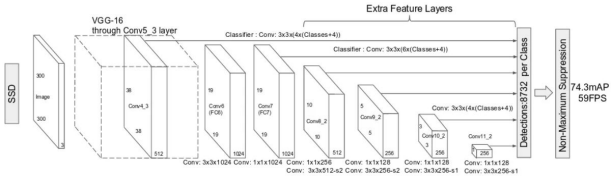


Fig. 7: Structure of ResNet SSD [6]

The second part of the pipeline is face alignment. To align faces, OpenCV's haar cascade method is used. Haar cascade method can be used to align eyes and frontal face[7]. Haar cascade can process 6.50 frames per second [7].

The third part of the pipeline is to create face representation. In this part, the model utilizes different models since it is a product of ensemble learning. The representation models used by models are VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace. I will explain each representation model. Addition to differences in the performance of the model, the required input size, output size, and vector structure of the output are differences among representation models.

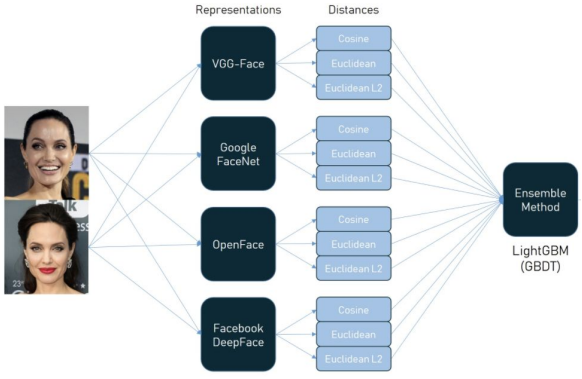


Fig. 8: Structure of ResNet SSD[8]

[5] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[6] https://github.com/serengil/deepface

[7] https://github.com/opencv/opencv/tree/master/data/haarcascades

[8] https://github.com/serengil/deepface

2.3.1. VGG-Face

VGG-Face has 22 layers and 37 deep units. The required input shape of the images are 224x224x3. After the representation process with the VGG-Face neural network, images are represented by 2622 size vectors. Although the image is a rgb image, to make the process faster, inputs are rescaled in range -1 to 1.

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Fig. 9: Layers of VGG-Face[9]

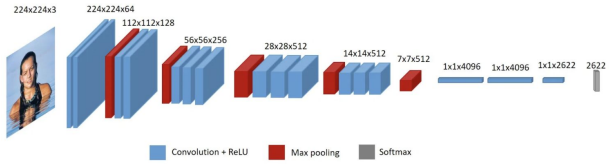


Fig. 10:Visualization of VGG-Face model Structure [8]

2.3.2. Google FaceNet

In the 2.1 and 2.2 sections, FaceNet models and its presentation method was mentioned. Please look at these parts for Google FaceNet.

2.3.3. OpenFace

The OpenFace model is based on inception ResNet v1. It expects 96x96x3 RGB image as input size and its output is 128 size vector which represents face's feature.

2.3.4. Facebook DeepFace

Facebook DeepFace has 8 layered CNN. As seen in Figure 11, there is a structure of Facebook DeepFace model, numbers refers layer number, C refers convolution layer, M refers max pooling layer, and L refers locally connected layer Although, it is not deeper than others, it has a huge number of parameters. You can see the parameter table in Figure 12. In the F7 layer, representations of the faces are created; thus, F8 is nor required for the ensemble learning model. 152x152 size of facial image is required as an input and its output which is representation of face's feature is 4096 dimensional vector.

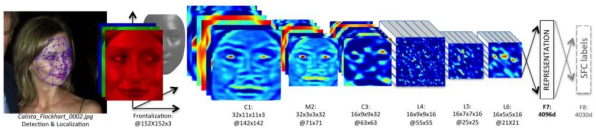


Fig. 11:Visualization of Facebook DeepFace model Structure [9]

Model	Num of Params
VGG-Face	145,002,878
Facenet	22,808,144
OpenFace	3,743,280
DeepFace	137,774,071

Fig. 12: Comparison of Number of Parameters Models have [10]

The last part of the pipeline is to verify and identify faces. Until this part, representation of the face, namely face features' vectors, are created. In this part, faces are verified and identified by using distance metrics which are cosine, euclidean, and euclidean L2. The euclidean distance was explained in part 2.1, please look at this section. I will explain cosine similarity metric. According to the distance and similarity metric results, the face(s) is identified and the model indicates whose face in the dataset, database or input data, is closer to the given face.

The cosine similarity is a metric which ensures to measure how two metrics are similar to each other. Assume there is an angle between two vectors which are a and b. The value of cos(theta) is our cosine similarity metric. If the value is higher than threshold, this means that these two vectors, actually face, are very similar, even the same. The formulation is shown in figure 13.

$$Cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\ \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\ \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a}\cdot\vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

Fig. 13: Comparison of Number of Parameters Models have [11]

[9] http://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/parkhi15.pdf

3. Experiment

To investigate whether current face recognition models are successful to recognize masked-face or not, I conducted an experiment. In the experiment, two datasets, which are Real World Fasked Face Recognition Dataset and LFW Simulated Masked Face Recognition Dataset, and three current face recognition models explained in section 2 are used. You can access the dataset here[10]. I made two experiments: firstly, I used unmasked face as train dataset and masked face as test dataset, secondly, I used masked face dataset as both train and test dataset. In this part of the paper, I will explain the dataset I used and the process of the experiment.

3.1. Datasets

Before the pandemic, there was nearly no dataset consisting of masked faces; however, after the pandemic, people started wearing masks and the masks covered their faces to a great extent. Because of this change, face recognition models' accuracy might be influenced. To make face recognition model's accuracy higher, data scientists and machine learning engineers started needing a new dataset for feeding their models to get higher accuracy. Wang and et. al. are one of the team that work on creating masked-face datasets [12]. I used their dataset and I made some changes on their dataset.

3.1.1. Real World Fasked Face Recognition Dataset (RMFRD)

Originally, RMFRD contains approximately 5,000 masked faces of 525 people and about 90,000 nominal faces. In the masked face part of the data set, there are unknown people who are not labeled in the unmasked part of the dataset. Also, there are empty folders and masked people who wear sun-glasses. Some issues might occur because of these circumstances. Firstly, people already wear masks and If they also wear sun-glasses, accuracy of the experiment might be affected. Since some part of the face is covered by a mask and there are a few parts to classify the people. Secondly, an empty folder is useless.Thirdly, If there are unknown people, this affects accuracy of the moldes since they are accepted as a failure and this lead model in the wrong way. To deal with these issues, I iterate the masked-face part of the dataset and I delete the empty folder, unknown folder, and image including the sun-glasses person. After my changes, the dataset consists of approximately 1338 masked faces of 352 people and 90468 nominal faces of 460 people.

As mentioned before, in my first experiment, I fed models with unmasked faces and tested it with a masked face. To conduct this part of the experiment, I used RMFRD. Indeed, I splitted the dataset according to the masked-unmasked face. The train dataset consists of unmasked-faces which are 90468 faces of 460 people and the test dataset consists of masked-faces which are 1338 masked faces of 352 people.

3.1.2. LFW Simulated Masked Face Dataset (LFW - SMFRD)

This dataset is obtained by Labeled Faces in the Wild (LFW) dataset[11]. LFW dataset consists of famous people images which are collected from web-sites. In the LFW - SMFRD has the same images; however, the face of the people in the LFW is simulated with a mask. The dataset consists of 13117 faces of 5713 people.

As mentioned before, in my second experiment, I fed models with masked faces and tested it with a masked face. To conduct this part of the experiment, I used LFW - SMFRD. Indeed, I splitted the dataset myself as a train and test dataset. Actually, I iterated the dataset and If a person has more than three face images, I chose one face image as a test dataset for each three face image the person has. In this way, I created my test dataset. The train dataset consists of 13027 masked faces of 5713 people and the test dataset consists of 70 masked faces of 48 people.

3.2. Process of The Experiment

To conduct my experiment, I used three current models which are Face Recognition with FaceNet Encoding, Face Recognition with FaceNet Embedding and SVC's Linear Kernel Models, Face Recognition with DeepFace Ensemble Learning (see 2. section). I used RMFRD dataset for unmasked-masked face recognition and SMFRD for masked-masked face recognition.

I utilized Kaggle, its API, and my local computer. Kaggle is an online community which generally consists of data scientists and machine learning practitioners. It provides users with opportunities to utilize their GPU, memory, and CPU. Also, people can

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

[11] http://vis-www.cs.umass.edu/lfw/

publish their work, notebooks, dataset, questions etc. on this community.

GPU requires for Most of the computer vision and face recognition process to make the process faster. Therefore, I used Kaggle and its GPU support for Face Recognition with FaceNet Encoding and Face Recognition with FaceNet Embedding and SVC's Linear Kernel Models. For the Face Recognition with DeepFace Ensemble Learning model, I used the CPU of my computer since there occurred an error on Kaggle and I did not figure it out . My notebooks are following:

For Face Recognition with FaceNet Encoding model with RMFRD[12], with SMFRD[13].

For Face Recognition with FaceNet Embedding and SVC's Linear Kernel model with RMFRD[14], with SMFRD[15].

For Face Recognition with DeepFace Ensemble Learning model with RMFRD[16], with SMFRD[17].

All notebooks are open source and notebooks on Kaggle have descriptions.

4. Results

In this section, I will mention results, namely scores, of the models for two dataset. Face Recognition with FaceNet Encoding model does not indicate train accuracy and loss; but test accuracy. Face Recognition with FaceNet Embedding and SVC's Linear Kernel model does not indicate loss accuracy; but, train and test accuracy. Face Recognition with DeepFace Ensemble Learning model does not indicate both test and train loss and accuracy; but, dataframe. I will show results of Face Recognition with FaceNet Encoding and Face Recognition with FaceNet Embedding and SVC's Linear Kernel model as a table. Additionally, I will mention results of Face Recognition with DeepFace Ensemble Learning model as an image and frame.

4.1. Results of Models with RMFRD

Model	Train Accuracy	Test Accuracy
Face Recognition with FaceNet Encoding	**None**	**~16.0**
Face Recognition with FaceNet Embedding and SVC's Linear Kernel	**~91.7**	**~19.5**

Table 1: Results of Face Recognition with FaceNet Encoding and Face Recognition with FaceNet Embedding and SVC's Linear Kernel Models on RMFRD

As it can be seen in table 1, even if the train's accuracy is high , about 92 percent, the test's accuracy is very low when compared with the train's accuracy. Also, It can be said that Face Recognition with FaceNet Embedding and SVC's Linear Kernel model is more successful than Face Recognition with FaceNet Encoding model when predicting masked faces with unmasked face inputs.

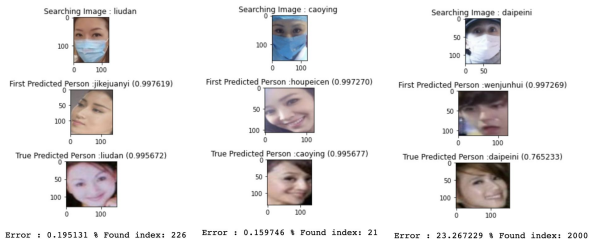


Fig. 14: Some Results of Face Recognition with DeepFace Ensemble Learning model on RMFRD

In figure 14, some results of the Face Recognition with DeepFace Ensemble Learning model on SMFRD can be seen. In the results, there are searching people with names, the first predicted people by model with names, and who the model should predict with names. At the bottom, there are error percentages which indicate error between the images predicted and should be. Addition to error percentage, there are index values which indicate which prediction people predicted true by the model.

For the first person named Liudan in figure 14, the model predicted her as Jikenjuanyi with about 99.7 percent similarity. The model gives predicted people

[12] https://www.kaggle.com/muhammeddalkran/masked-unmasked-face-recognition
[13] https://www.kaggle.com/muhammeddalkran/face-recognition-with-lfw-smfrd
[14] https://www.kaggle.com/muhammeddalkran/masked-face-recognition-with-rmfrd
[15] https://www.kaggle.com/muhammeddalkran/masked-face-recognition-with-lfw-smfrd
[16] https://github.com/dalkiran70/Face-Recognition-with-DeepFace
[17] https://github.com/dalkiran70/Face-Recognition-with-DeepFace

whose similarity metrics are higher than threshold in the train dataset as a list, namely dataframe. Actual person who should predict is the 226th person in the list. The model predicted her as Liudan with 99.5 percent similarity. The error percentage between these two predictions is about 0.19 percent. Although this error percentage seems very low, true prediction of the model is the 226th person in the predicted list.

4.2. Results of Models with SMFRD

Model	Train Accuracy	Test Accuracy
Face Recognition with FaceNet Encoding	**None**	**~54.1**
Face Recognition with FaceNet Embedding and SVC's Linear Kernel	**~46.3**	**~51.1**

Table 2: Results of Face Recognition with FaceNet Encoding and Face Recognition with FaceNet Embedding and SVC's Linear Kernel Models on SMFRD

As it can be seen in table 1, even if the test's accuracy is higher than the test's accuracy. Also, It can be said that Face Recognition with FaceNet Embedding and SVC's Linear Kernel model is less successful than Face Recognition with FaceNet Encoding model when predicting masked faces with masked face inputs.

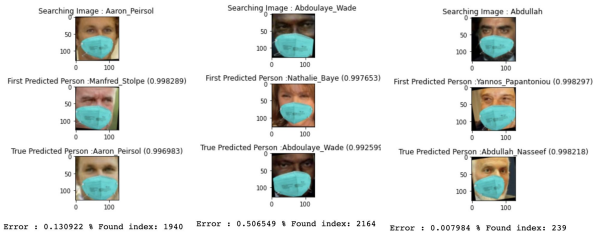


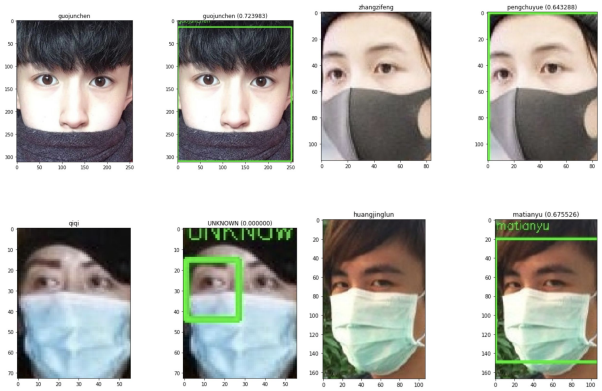
Fig. 15: Some Results of Face Recognition with DeepFace Ensemble Learning model on SMFRD

In figure 15, some results of the Face Recognition with DeepFace Ensemble Learning model on SMFRD can be seen. In the results, there are searching people with names, the first predicted people by model with names, and who the model should predict with names. At the bottom, there are error percentages which indicate error between the images predicted and should be. Addition to error percentage, there are index values which indicate which prediction people predicted true by the model.

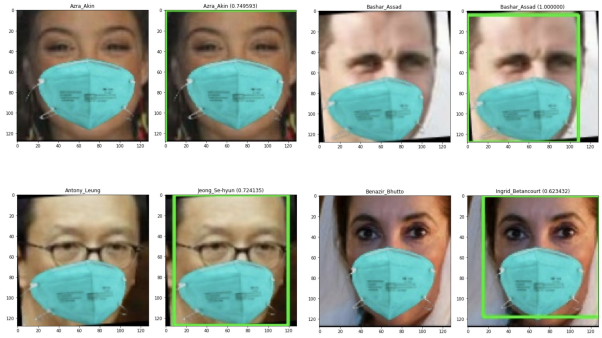
For the first person named Aaron Peirsol in figure 14, the model predicted him as Manfred Stolpe with about 99.8 percent similarity. The model gives predicted people whose similarity metrics are higher than threshold in the train dataset as a list, namely dataframe. Actual person who should predict is the 1940th person in the list. The model predicted him as Aaron Peirsol with 99.6 percent similarity. The error percentage between these two predictions is about 0.13 percent. Although this error percentage seems very low, true prediction of the model is the 1940th person in the predicted list.

4.3. Examples

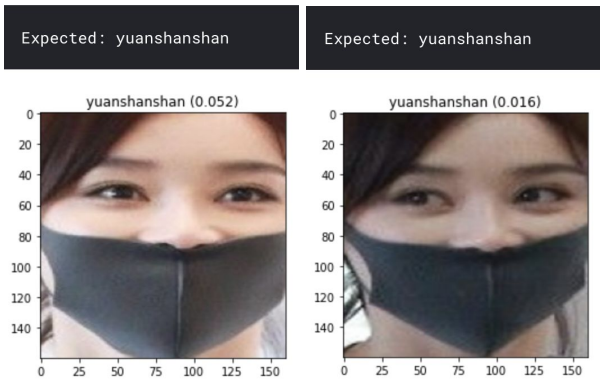
4.3.1. Examples of Face Recognition with FaceNet Encoding model with RMFRD

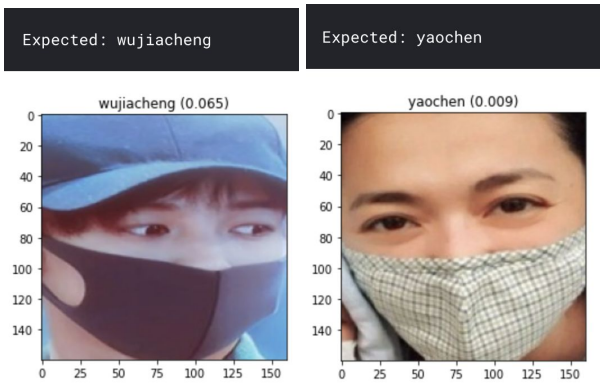


4.3.2. Examples of Face Recognition with FaceNet Encoding model with SMFRD

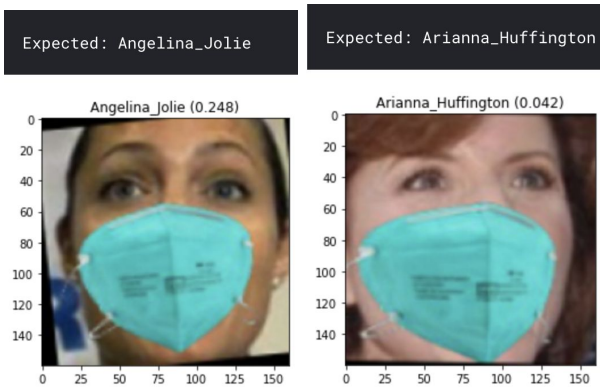


4.3.3. Face Recognition with FaceNet Embedding and SVC's Linear Kernel model with RMFRD

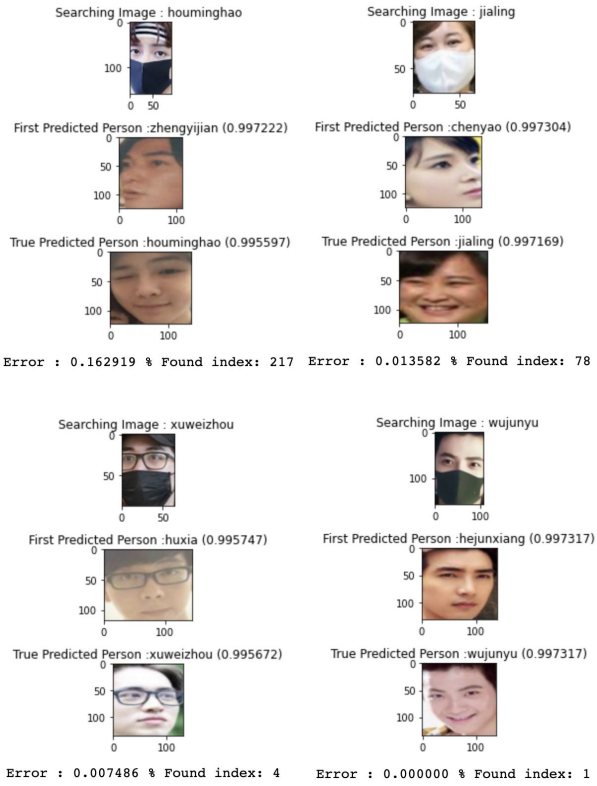




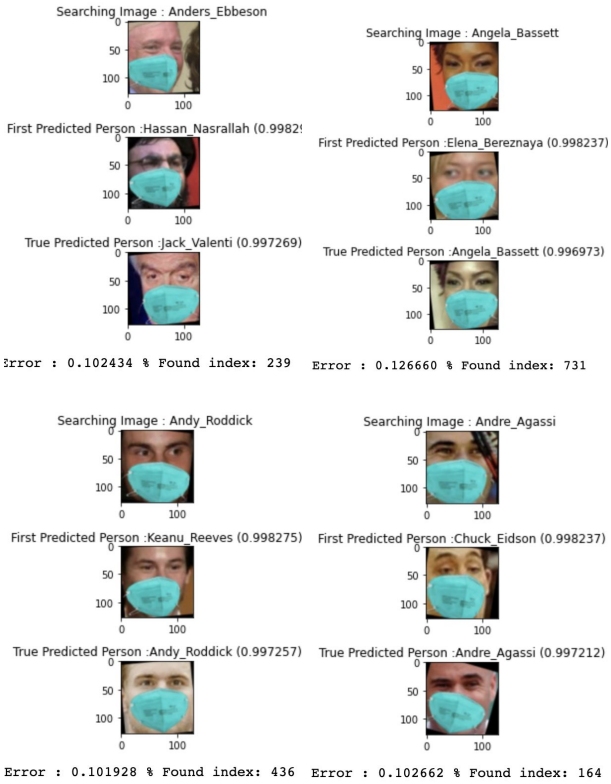
4.3.4. Face Recognition with FaceNet Embedding and SVC's Linear Kernel model with SMFRD



4.3.5. Face Recognition with DeepFace Ensemble Learning model with RMFRD



4.3.6. Face Recognition with DeepFace Ensemble Learning model with RMFRD



5. Discussion

After conducting experiments, we have results and we can analyze them. In this section, RMFRD's results and SMFRD's results will be analyzed.

Models with RMFRD have low test accuracies. One reason for this circumstance is that the dataset has a considerable number of classes which is 460. If the number of classes in the database is decreased, accuracy might increase. Since when looked at the result of DeepFace results, It can be considered that because of the number of classes, similarity metrics are very close; however, the model predicts the wrong person.

Another reason might be that models are trained with unmasked faces and they learn features of the unmasked area of the face. During the classification process, models identify masked faces as an unmasked face and due to this issue, they might misclassify masked faces. Since models do not know what a masked-face is, what is a mask, or where a mask is in the face.

Moreover, masks might cover a huge part of the face and models may not create a face's feature representation correctly due to the area covered by the mask. This might cause a classification problem, since models identify them according to face representation vectors. For example, Face Recognition with FaceNet Encoding model has a appr. 16 percent accuracy; however, it correctly predict guojunchen, in section 4.3.1, as appr. 72 percent. When looked at the predicted face, the person's mask does not cover the person's nose. This might increase accuracy since the model can represent and utilize the nose feature of the face.

Models with SMFRD have higher test accuracies; however, they can predict only half of the test face correctly. One reason for higher accuracy might be that models were fed with masked faces and they tried to predict masked faces. This circumstance ensures that all face representations do not include face's features where they are covered by masks.

Moreover, people in the RMFRD have a far eastern phenotype and the variety is less; nonetheless, although SMFRD consists of 5713 people, there is a lot of variety. This can be seen in the section 4.3. Nevertheless, the ratio of the number of faces per person in SMFRD is much fewer than RMFRD's. There is a dilemma, models with RMFRD can represent different features of the person since they have more face; however, their accuracy is less than models with SMFRD.

6. Conclusion

Current face recognition models are more efficient for masked face recognition when they are fed by masked faces. If these models are fed by unmasked faces, their accuracies are dramatically decreasing. Although models fed by masked faces have more accuracy, they can predict appr. only half of the faces. This may cause a problem since for example, one person has no permission to access a place; however, masked-face recognition systems predict him/her as another person who has permission. Actually, because of the low accuracy of current face recognition systems for masked faces, security holes may arise.

To deal with security holes due to wearing masks, the first stage of face recognition, namely face detection, may be transformed from face detection to masked face detection. In this way, models are able to detect masked face and extract face features which are not covered by masks.

References

[1] Dharaiya, D., 2020. Face Recognition Technology Past, Present, And Future. [online] ReadWrite. Available at: https://readwrite.com/2020/03/12/history-of-facial-recognition-technology-and-its-bright-future/ [Accessed 3 September 2020].

[2] Brownlee, J., 2020. How To Perform Face Detection With Deep Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/ [Accessed 4 September 2020].

[3] GitHub. 2020. 1Adrianb/Face-Alignment. [online] Available at: <https://github.com/1adrianb/face-alignment> [Accessed 4 September 2020].

[4] Zhang, K., Zhang, Z., Li, Z. and Qiao, Y., 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Processing Letters, 23(10), pp.1499-1503.

[5] Albon, C., 2020. SVC Parameters When Using RBF Kernel. [online] Chrisalbon.com. Available at: <https://chrisalbon.com/machine_learning/support_vector_machines/svc_parameters_using_rbf_kernel/> [Accessed 5 September 2020].

[6] Serengil, S., 2020. Deep Face Detection With Opencv In Python - Sefik Ilkin Serengil. [online] Sefik Ilkin Serengil. Available at: <https://sefiks.com/2020/08/25/deep-face-detection-with-opencv-in-python/> [Accessed 5 September 2020].

[7] Serengil, S., 2020. Face Alignment For Face Recognition In Python Within Opencv - Sefik Ilkin Serengil. [online] Sefik Ilkin Serengil. Available at: <https://sefiks.com/2020/02/23/face-alignment-for-face-recognition-in-python-within-opencv/> [Accessed 5 September 2020].

[8] Serengil, S., 2020. Deep Face Recognition With VGG-Face In Keras | Sefiks.Com. [online] Sefik Ilkin Serengil. Available at: <https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/> [Accessed 5 September 2020].

[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] Serengil, S., 2020. Face Recognition With Facebook Deepface In Keras - Sefik Ilkin Serengil. [online] Sefik Ilkin Serengil. Available at: <https://sefiks.com/2020/02/17/face-recognition-with-facebook-deepface-in-keras/> [Accessed 6 September 2020].

[11] Prabhakaran, S., 2020. Cosine Similarity - Understanding The Math And How It Works? (With Python). [online] Machine Learning Plus. Available at: <https://www.machinelearningplus.com/nlp/cosine-similarity/> [Accessed 6 September 2020].

[12] Wang, Zhongyuan, et al. "Masked face recognition dataset and application." arXiv preprint arXiv:2003.09093 (2020).