2D Face Recognition on CNN with Half-Average-Face

Yangfeng Zheng*

School of Data and Computer Science, SUN YAT-SEN UNIVERSITY, 132 East Waihuan Road, Guangzhou Higher Education Mega Center (University Town), Guangzhou 510006, P.R. China

*Corresponding author’s e-mail: zhengyf27@mail2.sysu.edu.cn

Abstract. As the size of training dataset of face recognition models becomes larger and larger, we are interested in a method called Average-Half-Face (AHF), which could halve the size of training samples. The AHF method divides a full face into two halves and then averages them together (reversing the columns of one of the halves). We preprocess the dataset with the method of AHF, and train them on two different models, Eigenfaces and Convolutional Neural Network (CNN). We compare the prediction results with those models trained on the original dataset. Previous researches showed that AHF is superior to Full-Face (FF), while our experiment results further showed that in some cases AHF also boosts CNN. The application of AHF can bring both saving in storage and reduction on training cost time.

1. Introduction

Face recognition has been extensively researched since 1960s [1]. For years of development, it has become one of the most well-known applications currently. Large industries in different countries are benefiting from this technology, from mobile phone human-face detection, to automatic surveillance in public places. However, digital images in recent years are larger and larger, up to several MB. For devices limited in processing power and storage, it is necessary to develop an optimized algorithm to carry on face recognition. The idea of Average-Half-Face (AHF) [2] was proposed to make use of the inherent bilateral symmetry of human faces. Due to the symmetry of human face, the left and right sides of a face have lots of overlapped features, which could be eliminated by averaging the two half-faces. Though this concept initially focused on extracting facial profiles, it has the side effect of decreasing size of images and accelerating training cost time, while giving an even better output than the traditional Full-Face (FF) approach.

However, the dataset [3] for experiments in that paper is relatively symmetric, which is ideal when comparing to real world cases. To simulate a more realistic situation, we employ the AHF and Eigenfaces algorithm on a more asymmetric dataset, the ORL Database of Faces [4]. Figure 1 displays a face image on the ORL Database, and the AHF expression of this image.
Figure 1 (b) shows how AHF averages two half faces, which may not align concisely. If the full face tilts, or the face expression or the face features (like a crooked nose) are asymmetric, the AHF expression would have ghosting. It raises a question: whether the loss of the information due to AHF would lead to a decrease in accuracy? According to the experiment result of Satone’s work [5], the answer might be ‘no’. AHF approach still performs a little better than FF approach in terms of accuracy. As the creator of AHF mentioned, it was still not clear what other factors were leading to the increase in accuracy [2]. As well, using only left or right half-faces, the outputs are worse than that of Full-Face [6]. Therefore, we can assume that the reason behind the increase in accuracy is far from the loss of information due to dimension decrease.

On the other hand, considering the development of CNN’s applications on face recognition, we are interested in the performance of AHF based on CNN. To discover it, we did a cross-validation both horizontally and vertically on PCA and CNN, AHF and FF. That is, we built four models, PCA+AHF, PCA+FF, CNN+AHF, CNN+FF, and then compare PCA+AHF with PCA+FF, PCA+AHF with CNN+AHF, for example. By these experiments, we evaluate the application of AHF on face recognition CNN.

In this paper, we first give a short description on AHF and our face recognition approaches. Next, we carry out related experiments, and then present results that demonstrate the accuracy gains of AHF.

2. Background

2.1. Average-Half-Face

The idea of face recognition using a half face comes from the fact the human face is roughly symmetrical. Zhao and Chellappa [7] developed a face recognition system based on face symmetry, which alleviates the effect of variations in illumination. Ramanathan et al. [8] used the notion of ‘half-faces’ to circumvent the problem of non-uniform illumination on measuring face similarity. Chen [9] combined the idea of half-face with face-template. In 2008, Harguess [2] came up with the idea of AHF. He derived it from the use of the Symmetry Preserving Singular Value Decomposition (SPSVD) [10].

Generally, AHF is featured by this 4-step preprocess:
1. Centralize the face. This is done by finding the location of the tip of the nose. If the face tilts, we centralize the face by finding a straight line that can divide the face into two halves of the same amount of pixels.
2. Partition the image into two equal left and right halves.
3. Reverse the columns of the left half-face.
4. Add two half faces and average them.

This preprocess aims to decrease the number of columns of an image, while the loss of information
is relatively little. Due to the fact that the two sides of a human face are symmetric, the mean-squared error of the difference of two half faces is low. To minimize this error, we need to centralize the image first. Then, we reverse the left half-face so that two halves are aligned before averaging. After that, we carry out the operation of averaging, which was proved a better way than just keeping the left half or the right half [2].

After preprocessing, everything is quite the same as what are indicated in a traditional Eigenfaces algorithm [11].

Satone [5] modified the plain AHF algorithm by adding steps of wavelet transform. She achieved an impressive promotion on error rate on the ORL database.

2.2. Eigenfaces
Eigenfaces [11] came up early and has played an important role in face recognition and detection. Eigenfaces is considered as a use of principal components analysis (PCA). It projects face images onto the principal components of the original set of training images, in order to extract features and decrease dimension. Then face images of individuals are compared (by Euclidean distance, typically) on this subspace of principal components.

Eigenfaces has been proved a powerful tool to carry out face recognition. Lawrence [12] performed experiments using eigenfaces algorithm on the ORL database and reached a 10.5% recognition error rate when the number of eigenfaces he used was between 175 and 199.

Several extensions or related methods are available. Multilinear principal components analysis (MPCA) [13] applies PCA to tensors or multilinear arrays. Furthermore, MPCA could be extended by using Linear Discriminant Analysis (LDA) [14] on the projected multilinear arrays to perform feature selection. Independent component analysis (ICA) [15], a generalization of PCA, tries to identify high-order statistical relationships between pixels to form a better set of basis vectors.

2.3. Convolutional Neural Network
Though Neural Network (NN) begin to catch everyone’s eyes not long ago, its researches of application on face recognition started at an early time. In an early comparison [16] between eigenfaces, elastic matching [17] and NN, NN did not perform well, and was upper bounded by eigenfaces. However, with an approach of convolutional neural network (CNN) [12], NN did have an astonishing performance on the ORL database, beating eigenfaces and Hidden Markov models [18].

After years of development, CNN has become a standard method in computer vision. Due to the availability of very large scale training datasets [19], and computational resources eligible to train a multi-layers CNN, CNN performs much better than the past. With the help of the Deep Neural Network (DNN) architecture, the state-of-the-art of face recognition has been significantly advanced [20]. In recent years, Deep Convolutional Neural Network has been applied on mobile devices [21], showing the necessity to improve the processing power of mobile devices and to optimize face recognition algorithms.

3. Methodology

3.1 ORL database
The ORL database [4] we used in this paper, contains a set of faces taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge, U.K. There are totally 400 images, while there are 40 individuals and 10 images for each one. Unlike the Yale face database [3], there are more variations, including facial expression, illumination and face angles. All the images were taken against a dark homogeneous background. Each image is greyscale with a resolution of 92 X 112.

3.2 Input Resizing
Each image in ORL database would be firstly resized from 92 X 112 to 46 X 56 (halved width and halved height), in order to accelerate training. Then, a half face image, as its name indicates, has a
halved width $46 \times 0.5 = 23$. The size of it is $23 \times 56$. To compare with half face directly, we will also scale the FF image to $32 \times 40$, keeping the aspect ratio while the amount of pixels is almost the same as that of half face images. The $23 \times 56$ input and $32 \times 40$ input have a similar training speed.

To conclude, we set 3 different input sizes: $46 \times 56$ (full face), $23 \times 56$ (half face) and $32 \times 40$ (full face).

3.3 Eigenfaces
There is no special detail in our Eigenfaces code. It projects face images onto the principal components of the original set of training images, in order to extract features and decrease dimension. Then face images of individuals are compared (by Euclidean distance, typically) on this subspace of principal components. The document of OpenCV [22] is referred to be checked.

3.4 CNN
Our model is quite the same as keras’s exemplary CNN model to recognize the MNIST database [23], which has a good performance on simple small-size image datasets. Table 1 is an overlook of this model.

Table 1. The CNN Layers.

| Layer   | Type           | Properties | Activation |
|---------|----------------|------------|------------|
| Conv2d_1 | Convolutional2D | 32@3*3    | ReLU       |
| Conv2d_2 | Convolutional2D | 64@3*3    | ReLU       |
| Pooling2D_1 | MaxPooling2D | pool size: 2*2 |             |
| Dropout_1 | Dropout        | drop rate: 0.25 |         |
| Flatten_1 | Flatten       |            |            |
| Dense_1   | Full_Connected | 128       | ReLU       |
| Dropout_2 | Dropout        | drop rate: 0.5 |         |
| Dense_2   | Full_Connected | 40        | Softmax    |

4. Results and Discussions
As we mentioned, there are 400 training images totally, and 10 for each of 40 subjects. From the 10 images of each subject, we randomly take $x$ as trainset, and as $(10 - x)$ testset. We trained several models on the trainset and tested them on the testset. Table 2 shows the accuracy of these models.

On CNN, we fix the learning rate at 1.0, and both train 90 epochs. The evaluation metric is accuracy.

Table 2 shows the accuracy of each models, while $x$ alters in the range from 3 to 9 ($x = 3$, $x = 5$, $x = 7$, $x = 9$). Every model was cross-validated.

Table 2. The overall accuracy of different models+input shapes.

| Model                | $x=3$ | $x=5$ | $x=7$ | $x=9$ |
|----------------------|-------|-------|-------|-------|
| Eigenfaces+FF        | 0.874 | 0.942 | 0.966 | 0.983 |
| Eigenfaces+HAF       | 0.893 | 0.947 | 0.969 | 0.990 |
| Eigenfaces+FF(half_size) | 0.876 | 0.944 | 0.966 | 0.984 |
| CNN+FF               | 0.751 | 0.917 | 0.954 | 0.970 |
| CNN+HAF              | 0.799 | 0.944 | 0.967 | 0.993 |
| CNN+FF (half_size)   | 0.735 | 0.912 | 0.957 | 0.980 |

Generally, with the increase of $x$, the accuracy increases on every model. Due to the limited size of testset, the accuracy does not look consistent on some $x$. For example, when $x = 3$, CNN+FF surpasses CNN+FF (half_size). However, when $x = 5$, CNN+FF (half_size) is better. In fact, their performances are almost the same, and the accuracy fluctuates because the small size of testset. Overall, all have good performance when $x$ is big. When $x \geq 7$, every model has an accuracy score more than 0.950. However, we can see that the HAF-related models are quite stable. No matter it is
Eigenfaces+HAF or CNN+HAF, it has a higher accuracy score on every $x$ than its counterparts (FF and FF (half_size)). We also noticed that only the CNN+HAF performed not worse than its Eigenfaces counterpart. The other two, FF and FF (half_size), were beaten by their Eigenfaces versions. CNN+HAF even predicted better than Eigenfaces+HAF when $x = 9$, proving its potential to recognize faces in an efficient way.

On the other hand, CNN often has a large scale of parameters, and the time spent on training/testing the CNN is usually long. In a CNN, the theoretical time complexity of all convolutional layers given by a related work [24] is:

$$O\left(\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right)$$

Here $l$ is the index of a convolutional layer, and $d$ is the depth (number of convolutional layers). $n_l$ is the number of filters in the $l$th layer. $n_{l-1}$ is also known as the number of input channels of the $l$th layer. $s_l$ is the spatial size (length) of the filter. $m_l$ is the spatial size of the output feature map. Reducing the size of input by half, the time costed on training/testing convolutional layers would be four times less if the computational overhead is not omitted.

In our experiments, Table 3 shows how many times were cost on every model. HAF-related models have the least time cost, and FF-related (half_size) models are the second. The FF-related models are the worst. CNN+HAF took about 43.1% time compared with CNN+FF. It could be concluded that the time cost and the input size are positively related. Therefore, we can see the advantage of adapting traditional full-size images to halved ones to train models. In spite of shorter training time, HAF performs better than FF. It is inspiring because we are able to train on some certain datasets with less time but get better performance.

| Model               | Cost_time(s) |
|---------------------|--------------|
| Eigenfaces+FF       | 0.682        |
| Eigenfaces+HAF      | 0.671        |
| Eigenfaces+FF(half_size) | 0.681    |
| CNN+FF              | 201.431      |
| CNN+HAF             | 86.774       |
| CNN+FF(half_size)   | 96.590       |

Figure 2 shows the learning curves of CNN on different input shapes. We can see that FF, HAF and FF (half_size) are all about to converge around 60 epochs.

5. Conclusion
In this paper, we presented an experimental evaluation of the adaptation of AHF on CNN. The models were evaluated using different number of training images and test images by cross-validation. The experiment showed the idea of Half-Average-Face could also work well on CNN, and it brought improvements on both prediction accuracy and training time.
However, to be applied on real production, AHF should be experimented further on more datasets. Actually, the ORL database is relatively small, and the images are not as various as daily images. Therefore, the application of AHF might be limited on some certain areas. In some fields, like facial identification, in which users would take several front-face images as trainsets, AHF could be a good adaptation.

Last but not least, the reason why AHF can lead to a better performance on face recognition is still mysterious. In the former work [6], AHF performed well on traditional methods. We showed further that AHF also performed well on CNN. We guessed that AHF helped the model to extract some features better, at least on the datasets our work and related works used.

**References**

[1] https://www.facefirst.com/blog/brief-history-of-face-recognition-software/

[2] Harguess J, Gupta S, Aggarwal J K 2008 3D face recognition with the average-half-face, 2008 19th International Conference on Pattern Recognition (Tampa) p 1-4

[3] Yale University 2002 http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

[4] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[5] Satone M P and Kharate G K 2012 Face Recognition Based on PCA on Wavelet Subband of Average-Half-Face Journal of Information Processing Systems 8(3) 483-494

[6] Harguess J and Aggarwal J K 2009 A case for the average-half-face in 2D and 3D for face recognition IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (Miami) 7-12

[7] Zhao W and Chellappa R 2000 Illumination-insensitive face recognition using symmetric shape-from-shading Proceedings IEEE Conference on Computer Vision and Pattern Recognition (Hilton Head Island) 1 286-293

[8] Ramanathan N, Chellappa R and Roy Chowdhury A K R 2004 Facial similarity across age, disguise, illumination and pose International Conference on Image Processing 3 1999-2002

[9] Chen W, Sun T, Yang X and Wang L 2009 Face detection based on half face-template 9th International Conference on Electronic Measurement & Instruments (Beijing) 4 54-4-58

[10] Shah M, D. Sorensen C 2006 A Symmetry Preserving Singular Value Decomposition SIAM Journal on Matrix Analysis and Applications 28 749 - 769

[11] Turk M A and Pentland A P 1991 Face recognition using eigenfaces IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Maui) 586-591

[12] Lawrence S, Giles C L, Tsoi A C and Back A D 1997 Face recognition: a convolutional neural-network approach IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition 8 98-113

[13] Lu H, Plataniotis K N and Venetsanopoulos A N 2008 MPCA: Multilinear Principal Component Analysis of Tensor Objects IEEE Transactions on Neural Networks 19(1) 18-39

[14] Etemad K and Chellappa R 1997 Discriminant analysis for recognition of human face images Proc. Ist Intl.Conf.of Audio-and Video-based Biometric Person Authentication 14 1724-1733

[15] Bartlett M S, Movellan J R and Sejnowski T J 2002 Face recognition by independent component analysis IEEE Transactions on Neural Networks 13(6) 1450-1464

[16] Zhang J, Yan Y and Lades M 1997 Face recognition: eigenface, elastic matching, and neural nets Proceedings of the IEEE 8(9) 1423-1435

[17] Lades M, Vorbruggen J, Buhmann J, Lange J, Malburg C V D and Wurtz R 1993 Distortion invariant object recognition in the dynamic link architecture IEEE Transactions on Computers 42(3) 300–31

[18] Samaria F S1994 Face recognition using hidden Markov model Trinity College, Univ. Cambridge, Cambridge, U.K., 1994.

[19] Parkhi O M, Vedaldi A, Zisserman A2015 Deep Face Recognition British Machine Vision Conference 41.1-41.12
[20] Sun Y, Liang D, Wang X, et al 2015 DeepID3: Face Recognition with Very Deep Neural Networks Computer Science, 2015.

[21] Amos B, Ludwicz, Satyanarayanan M 2016 OpenFace: A general-purpose face recognition library with mobile applications

[22] https://docs.opencv.org/2.4/modules/contrb/doc/facerec/facerec_tutorial.html#algorithm-description

[23] https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

[24] He K, Sun J 2015 Convolutional neural networks at constrained time cost Computer Vision and Pattern Recognition IEEE 5353-5360