Enhance Deep Learning Performance in Face Recognition

Ze Lu, Xudong Jiang, Alex Kot
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
e-mail: {zlu008, exdjiang, eackot}@ntu.edu.sg

***Abstract*—Deep convolutional neural networks (CNNs) based face recognition approaches have been dominating the field. The success of CNNs is attributed to their ability to learn rich image representations. But training CNNs relies on estimating millions of parameters and requires a very large number of annotated training images. A widely-used alternative is to fine-tune the CNN that has been pre-trained using a large set of labeled images. However, we show that fine-tuning pre-trained CNNs cannot provide satisfactory face recognition performance when training and testing datasets have large differences. To address this problem, we propose to improve the face recognition performance of CNNs by using non-CNN features. Extensive experiments are conducted on LFW and FRGC databases using the pre-trained CNN model, VGG-Face. Results show that the complementary information contained in non-CNN features greatly improves the face verification rate/accuracy of CNNs on LFW and FRGC databases. Furthermore, we show that non-CNN features are more effective in enhancing the performance of pre-trained CNNs than fine-tuning.**

Keywords-CNNs; fine-tuning; raw pixels; VGG-face

I. INTRODUCTION

Driven by its broad applications in human-computer interaction, homeland security, and entertainment [1-3], face recognition has always been one of the most popular biometric topics in computer vision community [4]. Over the past decade, encouraging improvements have been achieved in face recognition performance by convolutional neural networks (CNNs) [5-7].

CNNs are high-capacity classifiers with very large numbers of parameters that must be learned from millions of training examples [8]. The main power of a CNN lies in its deep architecture [5-7], which allows for extracting a set of discriminating features at multiple levels of abstraction. However, it is impossible to collect millions of annotated images and implement the complex training process for each new face recognition task. In practice, very few people train an entire CNNs from scratch (or full training). Instead, it is common to pretrain a CNNs on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the CNNs either as a fixed feature extractor or an initialization for the task of interest. Computer vision datasets have significant differences in image statistics [9]. When training and testing datasets have differences in viewpoints, image sizes, scene context, illumination, expressions or other factors, the face recognition performance using pre-trained CNNs as a feature extractor will be affected inevitably. Thus the pre-trained CNN model is usually used as an initialization for fine-tuning on images from the application of interest [10-14]. However, our experimental results show that the use of pre-trained deep CNNs with fine-tuning cannot provide satisfactory face recognition performance, especially when there exist big differences between training and targeted face images.

To improve the face recognition performance of pre-trained CNNs with or without fine-tuning, we propose to combine image representations learned by CNNs with non-CNN features. Even though features extracted by CNNs from face images are discriminative, they were designated to solve the classification problem that is strictly restricted. As fine-tuning is still based on the pre-trained CNN model, its performance is constrained by the ability of the underlying pre-trained CNNs. The non-CNN features depict the characteristics of targeted face images from a different perspective, which is complementary to the image representations learned by CNNs. Thus non-CNN features provide an effective way to enhance the recognition performance of CNNs. What's more, the non-CNN features can also be used together with fine-tuning to further boost the face recognition performance. By conducting extensive experiments on LFW and FRGC datasets using the VGG-Face model [7], we show that the face recognition performance of the pre-trained VGG-Face model with or without fine-tuning can be improved significantly by non-CNN features. Furthermore, non-CNN features improve the performance of pre-trained CNNs much more than fine-tuning.

II. CONVOLUTIONAL NEURAL NETWORKS

A deep CNN has multiple layers that progressively compute features from input images. It contains three basic components: convolutional, pooling and fully-connected layers.

The convolutional layer is the core building block of a CNN. It detects local features at all locations of the input image by learning a filter bank. Every filter is small along the width and height directions, but extends through the full depth of the input volume. We use $x_i^{(l)}$ to represent the i-th feature map at layer *l*. The convolutional layer computes as following:

978-1-5090-6238-6/17/$31.00 ©2017 IEEE

$$x_j^{(l+1)} = s(\sum_i F_{ij}^{(l)} * x_i^{(l)} + b_j^{(l)}) \quad (1)$$

where $F_{ij}^{(l)}$ denotes the filter that connects feature map $x_i^{(l)}$ to output map $x_j^{(l+1)}$ at layer l, $b_j^{(l)}$ is the bias for the j-th output feature map, $s(\cdot)$ is some element-wise non-linear function and * denotes the discrete 2D convolution. The output feature maps along the depth dimension are stacked and produce the output volume.

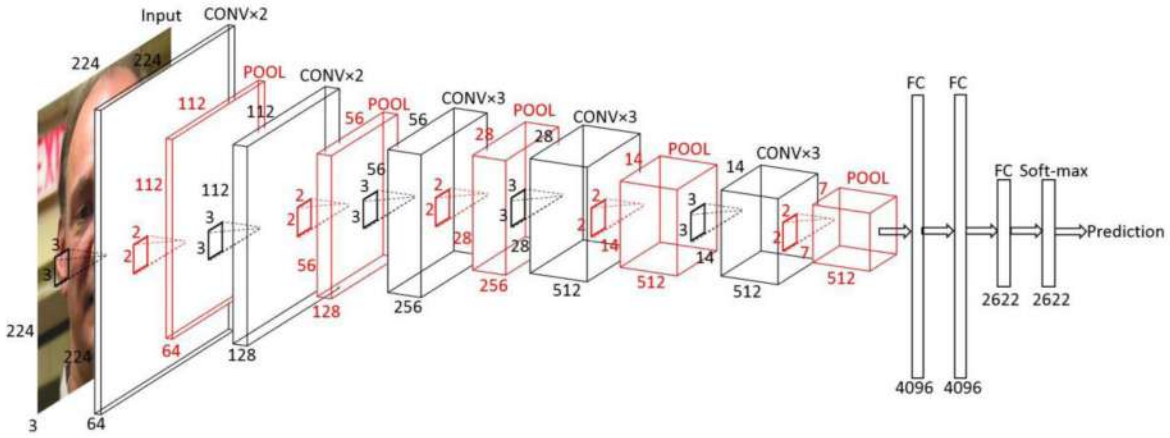

Figure 1. VGG-Face architecture, CONV indicates convolutional layers, POOL indicates pooling layers and FC indicates fully-connected layers. For each layer, the filter size, number of filters, size of resulted feature maps are also indicated.

The pooling layer is periodically inserted in-between successive convolutional layers in a CNN. Its function is to progressively reduce the spatial size of the representations to reduce the amount of parameters and computation in the network, and hence to control overfitting.

In a fully-connected layer, output units have full connections to all input units. Their activations can hence be computed with a matrix multiplication followed by a bias offset, specifically:

$$x^{(l+1)} = s(W^{(l)} x^{(l)} + b^{(l)}) \quad (2)$$

where $x^{(l)}$ is the vectored input from layer l, $W^{(l)}$ and $b^{(l)}$ are the parameters of the fully-connected layer at layer l.

In order to provide training signals (class scores), the last layer of CNNs is normally associated with some loss, and the training for CNNs can be done by doing gradient descent on the parameters with respect to the loss.

III. FINE-TUNING

Training a deep CNN from scratch (or full training) is not without complications [15]. First, CNNs require a large amount of labeled training data. Second, the training process requires extensive computational and memory resources. Third, training a deep CNN is often complicated due to overfitting and convergence issues. Therefore, it is common to pretrain a CNN on a very large dataset, and then use the pre-trained CNN model either as a fixed feature extractor or an initialization for the task of interest. As for the former method, we remove the last fully-connected layer outputting class scores and use the rest of the pre-trained model as a fixed feature extractor.

An alternative way is to fine-tune the weights of the pre-trained network by continuing the back-propagation process on targeted images. Fine-tuning is a procedure based on the concept of transfer learning [16]. It has been used successfully in many applications such as [11-13].

Several pre-trained CNN architectures have been proposed in the literature and some have been shown to produce better results than the most advanced state-of-the-art face recognition methods. In our paper, we adopt the VGG-Face deep architecture [7] shown on Fig. 1. The VGG-Face model was learned from a large face dataset containing 982,803 web images of 2622 celebrities and public figures. It can be used as a feature extractor for classifying any arbitrary face image by running the image through the entire network. The 4096-dimensional vector of the first fully-connected layer 'fc6' is used as the feature.

To do fine-tuning, we remove the final softmax layer and initialize a new one with random values. The new softmax classifier is trained from scratch using the back-propagation algorithm with data from our new database. In addition, the number of output feature maps of the last fully-connected layer should be changed to the number of classes in the new dataset. We divide the training set of the new database into a smaller training set with 80% of the training data and a validation set with the remaining 20% of the training data. The training process will be terminated when the highest accuracy on the validation set is observed. Our implementation is based on the open source deep learning library Caffe [17], and we run the experiment using a NVIDIA K80 GPU, which is a dual GPU board that combines 24 GB of memory.

IV. ENHANCE THE DEEP LEARNING PERFORMANCE BY NON-CNN FEATURES

The good performance achieved by pre-trained CNNs is strictly restricted to applications which have similar face images to the pre-training datasets. In practice, different

applications have significant variations in viewpoints, image sizes, scene context, illumination, expressions or other factors. Although fine-tuning can offer some help in improving the performance of pre-trained CNNs, an effective fine-tuning needs to fine-tune all layers of the pre-trained models when the variations between the source and target applications are significant. However, when the pre-trained CNNs are fine-tuned too much, the limited number of fine-tuning data might cause the problem of overfitting. Given the "data-hungry" nature of CNNs and the difficulty of collecting large-scale image datasets, the applicability of CNNs to tasks with limited amount of training data appears as an important and open problem.

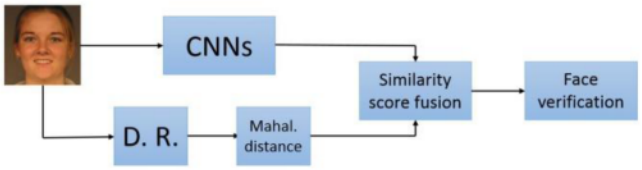

Figure 2. Framework of the proposed method, D. R. indicates dimension reduction and Mahal. indicates Mahalanobis.

Here, we propose to use non-CNN features to enhance the face recognition performance of CNN models based on the framework shown on Fig.2. Although there are many different handcrafted features for image recognition [18-19], raw pixel intensity values keep the most original appearance information of face images, which are used in many recent face recognition works [20-21]. The RGB color space describes color by the same three components: red, green, and blue. It is a fundamental and commonly used color space. In order to provide complementary and raw information of the targeted face images to image representations learned by CNNs, we combine raw pixels of face images in the RGB color space with image representations learned by CNNs.

To get similarity scores from CNN features, we feed face images to the VGG-Face model as shown on Fig. 1. The 4096-dimensional vector obtained at the first fully-connected layer 'fc6' by eq. 2 is taken as the CNN feature. Suppose x_1 and x_2 are two learned feature vectors for the gallery and query face images, respectively, we use the following equation to calculate their similarity score:

$$s^{CNN} = \frac{x_1^t \cdot x_2}{|x_1| \times |x_2|} \tag{3}$$

where t indicates transpose, $\cdot$ indicates the inner product and $|\cdot|$ indicates L2 norm.

As for the similarity score of non-CNN features, we first apply PCA to raw pixels of face images for dimension reduction. PCA is a standard technique for dimensionality reduction and has been applied to a broad class of computer vision problems. It has been explained in [22-23] that PCA plays an important role in the classification. It alleviates the overfitting problem or improves the generalization capability by removing the subspace spanned by eigenvectors corresponding to the small eigenvalues while keeping the principal components. Let $z \in R^N$ be a column vector of raw pixels, whose covariance matrix is Σ_z. The PCA factorizes the covariance matrix Σ_z into the following form:

$$\Sigma_z = \Phi \Lambda \Phi^t \tag{4}$$

where Φ is an orthogonal eigenvector matrix and Λ is a diagonal eigenvalue matrix. PCA reduces the dimension of the vector z by:

$$z' = P^t z \tag{5}$$

where P is the matrix of the m largest eigenvectors in Φ and z' is the resulting m-dimensional vector.

After dimension reduction, we use the Mahalanobis distance to compute similarity scores between low-dimensional feature vectors instead of the Euclidean distance. To start with, we calculate the within-class scatter matrix Σ_z^w of z as following:

$$\Sigma_z^w = \sum_{i=1}^{p} \sum_{j=1}^{q_i} (z_{ij} - \overline{z}_i)(z_{ij} - \overline{z}_i)^t \tag{6}$$

where i indicates the number of classes, j indicates the number of images for class i and $\overline{z}_i$ is the mean vector for samples of class i. Then we factorize the within-class scatter matrix Σ_z^w using eq. 4 to obtain the eigenvector matrix Φ^w and the eigenvalue matrix Λ^w. After that, we calculate the whitening matrix Q of Σ_z^w by

$$Q = \Phi^w (\Lambda^w)^{-\frac{1}{2}} \tag{7}$$

The whitened low-dimensional feature vector is

$$z'' = Q^t z' = Q^t P^t z \tag{8}$$

The Euclidean distance computed from z'' is the Mahalanobis distance of z'. Suppose we have two feature vectors z''_1 and z''_2 for gallery and query face images, respectively, their similarity score s^{nCNN} is also computed by eq. 3.

To fuse s^{CNN} and s^{nCNN} for face recognition, we adopt the following weighted sum rule:

$$s = w^{CNN} s^{CNN} + w^{nCNN} s^{nCNN} \tag{10}$$

where w^{CNN} and w^{nCNN} indicate weights for CNN similarity score and non-CNN similarity score, respectively. The

resulting similarity score s can be used for subsequent Nearest-neighbor classification. For different face recognition applications, the similarity between pre-training images and targeted images is different. Thus the performance of pre-trained CNNs on different databases also varies. We determine the value of w^{CNN} and w^{nCNN} by the similarity between VGG images and the targeted images.

V. EXPERIMENTS

Extensive experiments are conducted on LFW [24] and FRGC [25] databases to evaluate the proposed method for face recognition tasks.

A. Databases

1) LFW database: The LFW database has been widely used as a benchmark dataset to evaluate face recognition algorithm. It consists of 13,233 images of 5,749 subjects. All images are collected from the internet with large pose, illumination and facial-expression variations, as well as occlusions. We report the averaged results over 10 folds of View 2. More precisely, 3000 positive pairs (two images of the same subject) and 3000 negative pairs (two images of different subjects) are divided into 10 subsets. In each fold, 9 subsets are used as the training set and the hold-out subset is used for testing. The task is to determine whether an image pair comes from the same subject or not. Fig. 3 shows some cropped images of the LFW database. We set the value of w^{CNN} *and* w^{nCNN} to be 0.9 and 0.1 on LFW database.

Figure 3. Some cropped images of the LFW database.

Figure 4. Some cropped images of the FRGC database.

2) FRGC database: There are 12,776 training images, 16,028 controlled target images, and 8,014 uncontrolled query images in FRGC database. The controlled images have good image quality, while the uncontrolled images display poor image quality, such as large illumination variations, low resolution, and blurring. The controlled images have good quality, while the uncontrolled images display poor quality, such as large illumination variations, low resolution, and blurring. These uncontrolled factors pose grand challenges to face recognition performance. FRGC Experiment 4 has been reported to be the most challenging FRGC experiment, so it is chosen to assess the face recognition performance in our experiments. The face recognition performance is reported by using the face verification rate (FVR) at the false accept rate (FAR) of 0.1%. Fig. 4 shows some cropped images of the FRGC database. We set the value of w^{CNN} *and* w^{nCNN} to be 0.5 and 0.5 on FRGC database.

B. Results

1) Results on LFW database: The face verification accuracy is shown on Table I for LFW database. VGG indicates the pre-trained VGG-Face model and nCNN indicates the non-CNN feature. Because there is no label information to use for fine-tuning the pre-trained VGG-Face model on LFW, we only test the performance of pre-trained VGG-Face model. Empirical studies [26] in face recognition proved that a minimum face resolution between 32×32 and 64×64 is required for stand-alone recognition algorithms. So the image size of 32×32 and 64×64 is used. According to our experiments, the image size of 140×140 produces the best face verification accuracy when using the pre-trained VGG-Face model on LFW database. Thus the image size of 140×140 is also used.

TABLE I. RESULTS ON LFW DATABASE

Image size	Face verification accuracy (%)	
	VGG	*VGG+nCNN*
32×32	85.62	**86.85**
64×64	91.97	**92.35**
140×140	97.43	**97.45**

2) Results on FRGC database: The face verification rates at the FAR of 0.1% are shown on Table II for FRGC database. Our experiments show that the image size of 211×201 produces the best face verification performance when using the pre-trained VGG-Face model on FRGC images. Thus the image size of 32×32, 64×64 and 211×201 are used.

FVGG indicates the fine-tuned VGG-Face model. To do fine-tuning, we set the base learning rate to 0.001 and it drops by a factor of 0.5. In order to magnify the number of training samples per user, we perform the following data augmentation, for each training sample of size m × m, we extract all possible crops of size (m-2) × (m-2). Each crop is also flipped along its vertical axis yielding a total of 2 × 3 × 3=18 crops. The crops are then re-sized back to 224×224 and used for training the CNNs. The batch size is set to be 50. We run the back-propagation algorithm for around 4,000 iterations.

TABLE II. RESULTS ON FRGC DATABASE

Image Size	Face verification rate (%) at FAR of 0.1%			
	VGG	*VGG+nCNN*	*FVGG*	*FVGG+nCNN*
32×32	32.84	**82.61**	44.97	**83.32**
64×64	45.32	**84.11**	75.34	**90.11**
211×201	88.86	**95.37**	92.50	**96.00**

C. *Analysis*

From the experimental results shown on Table I and Table II, we can observe that the face verification performance of the pre-trained VGG-Face model is unsatisfactory on LFW and FRGC datasets. Even after fine-tuning, the performance of CNNs is not good enough. By using non-CNN features, the face verification performance of both pre-trained and fine-tuned CNN models can be improved significantly. Simple raw pixels of RGB only marginally improve the face verification accuracy of CNNs on LFW database, because the VGG-face model was initially trained and tuned to achieve state-of-the-art results on LFW database [7].

What's more, the performance improvement of pre-trained models achieved by non-CNN features is much larger than that achieved by fine-tuning. The improvement made by non-CNN features can be significant when the targeted images have large variations with pre-training images.

VI. CONCLUSION

In this paper, we show that pre-trained CNN models and widely used fine-tuned CNN models cannot provide satisfactory face recognition performance under the case that training and testing datasets have big differences. To address this problem, we propose to combine non-CNN features with image representations learned by CNNs. Extensive experiments are conducted on LFW and FRGC databases using the pre-trained CNN model, VGG-Face. Results show that non-CNN features can greatly improve the face verification performance of pre-trained CNNs and fine-tuned CNNs.

REFERENCES

[1] J. Lai and X.D. Jiang, "Class-wise Sparse and Collaborative Patch Representation for Face Recognition," IEEE Trans. Image Processing 25.7 (2016): 3261-3272.

[2] Z. Lu, X.D. Jiang, and A.C. Kot. "A Color Channel Fusion Approach for Face Recognition." IEEE Signal Processing Letters 22.11 (2015): 1839-1843.

[3] Z. Lu, X.D. Jiang, and A.C. Kot. "An effective color space for face recognition." IEEE Intl. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2016.

[4] X.D. Jiang and J. Lai, "Sparse And Dense Hybrid Representation via Dictionary Decomposition for Face Recognition," IEEE Trans. Pattern Analysis and Machine Intelligence 37.5 (2015): 1067-1079

[5] T. Yaniv, et al. "Deepface: Closing the gap to human-level performance in face verification." Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition. 2014.

[6] Y. Sun, X. Wang, and X. Tang. "Deep learning face representation from predicting 10,000 classes." Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition. 2014.

[7] P. Omkar, A. Vedaldi, and A. Zisserman. "Deep face recognition." British Machine Vision Conference. Vol. 1. No. 3. 2015.

[8] O. Maxime, et al. "Learning and transferring mid-level image representations using convolutional neural networks." Proceedings of the IEEE conf. computer vision and pattern recognition. 2014.

[9] T. Antonio, and A. A. Efros. "Unbiased look at dataset bias." IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR), 2011.

[10] Z. Liu, et al. "Deep learning face attributes in the wild." Proceedings of the IEEE Intl. Conf. Computer Vision. 2015.

[11] F., Sachin Sudhakar, M. J. Saberian, and Li-Jia Li. "Multi-view face detection using deep convolutional neural networks." Proceedings of the ACM Intl. Conf. Multimedia Retrieval, 2015.

[12] X.D. Jiang, A. and Wah, "Constructing and Training Feed-Forward Neural Networks for Pattern Classification," Pattern Recognition 36.4 (2003): 853-867.

[13] K. Zhang, et al. "Gender and smile classification using deep convolutional neural networks." Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition Workshops. 2016.

[14] X. Peng, et al. "Towards Facial Expression Recognition in the Wild: A New Database and Deep Recognition System." Proceedings of the IEEE Conf. Computer Vision and Pattern Recognition Workshops. 2016.

[15] E. Dumitru, et al. "The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training." AISTATS. Vol. 5. 2009.

[16] B. Yoshua. "Deep Learning of Representations for Unsupervised and Transfer Learning." ICML Unsupervised and Transfer Learning 27 (2012): 17-36.

[17] Y. Jia, et al. "Caffe: Convolutional architecture for fast feature embedding." Proceedings of the ACM intl. conf. Multimedia, 2014.

[18] X.D. Jiang, "Extracting Image Orientation Feature by Using Integration Operator," Pattern Recognition, 40.2 (2007): 705-717.

[19] X.D. Jiang, "On Orientation and Anisotropy Estimation for Online Fingerprint Authentication," IEEE Trans. Signal Processing, 53.10 (2005): 4038- 4049.

[20] J.Y. Choi, M.R. Yong and K. N. Plataniotis. "Color face recognition for degraded face images." IEEE Trans. Systems, Man, and Cybernetics, Part B (Cybernetics) 39.5 (2009): 1217-1230.

[21] P. Shih, and C. Liu. "Comparative assessment of content-based face image retrieval in different color spaces." Intl. Journal of Pattern Recognition and Artificial Intelligence 19.07 (2005): 873-893.

[22] X.D. Jiang. "Linear subspace learning-based dimensionality reduction." IEEE Signal Processing Magazine 28.2 (2011): 16-26.

[23] X.D. Jiang. "Asymmetric principal component and discriminant analyses for pattern classification." IEEE Trans. Pattern Analysis and Machine Intelligence 31.5 (2009): 931-937.

[24] G.B. Huang, et al. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Vol. 1. No. 2. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[25] P. J. Phillips, et al. "Overview of the face recognition grand challenge." IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2005.

[26] Y.M. Lui, et al. "A meta-analysis of face recognition covariates." Biometrics: Theory, Applications, and Systems (2009): 1-8.