VLAD Encoded Deep Convolutional Features for Unconstrained Face Verification

Jingxiao Zheng[1], Jun-Cheng Chen[1], Navaneeth Bodla[1], Vishal M. Patel[2] and Rama Chellappa[1]
1. Center for Automation Research, UMIACS, University of Maryland, College Park, Maryland, 20740
2. Rutgers University, 94 Brett Road, Piscataway, NJ 08854
jxzheng@umiacs.umd.edu, pullpull@cs.umd.edu, nbodla@umiacs.umd.edu, vishal.m.patel@rutgers.edu, rama@umiacs.umd.edu

***Abstract*—We present a method for combining the Vector of Locally Aggregated Descriptor (VLAD) feature encoding with Deep Convolutional Neural Network (DCNN) features for unconstrained face verification. One of the key features of our method, called the VLAD-encoded DCNN (VLAD-DCNN) features, is that spatial and appearance information are simultaneously processed to learn an improved discriminative representation. Evaluations on the challenging IARPA Janus Benchmark A (IJB-A) face dataset show that the proposed VLAD-DCNN method is able to capture the salient local features and yield promising results for face verification. Furthermore, we show that additional performance gains can be achieved by simply fusing the VLAD-DCNN features that capture the local variations with the traditional DCNN features which characterize more global features.**

I. Introduction

Learning invariant and discriminative features from images and videos is one of the central goals of research in many computer vision tasks such as object recognition, object detection and face recognition. Many approaches have been proposed in the literature that extract over-complete and high-dimensional features from images to handle large data variations and noise. For instance, the high-dimensional multi-scale Local Binary Pattern (LBP) [3] features extracted from local patches around facial landmarks is reasonably effective for face recognition. Face representation based on Fisher vector (FV) has also been shown to be effective for face recognition [24], [19], [5]. Some of the other feature encoding methods that have been successfully used in many computer vision applications, include Bag-of-Visual-Words (BoVW) model [7], Vector of Locally Aggregated Descriptor (VLAD) [12] and Super Vector Coding [27].

In recent years, deep convolutional neural networks (DCNN) have demonstrated impressive performances on several computer vision problems such as object recognition [16][25] [10], object detection [8], and face verification [23], [20]. It has been shown that a DCNN model can not only characterize large data variations but also learn a compact and discriminative representation when the size of the training data is sufficiently large.

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. We thank Mr. Rajeev Ranjan [21] for providing the facial landmarks used in the experiments.

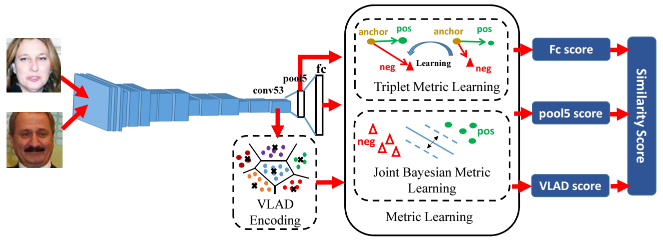


Fig. 1. An overview of the proposed fusion framework to combine the global average pooling, fully-connected layer features and VLAD features for unconstrained face verification.

In particular, several approaches have combined the DCNN features with other feature encoding methods to obtain improved results. Gong *et al.* [9] extracted the multi-scale deep features followed by VLAD for feature encoding and demonstrated promising results for image retrieval and classification tasks. Cimpoi *et al.* [6] proposed a FV-DCNN approach to combine FV with DCNN features for texture recognition.

Motivated by the success of these approaches, we propose a face verification method which essentially combines VLAD encoding method with DCNN features for face verification. Figure 1 gives an overview of the proposed VLAD-DCNN method. Unlike some of the previous approaches [6] and [9], we take the spatial information into consideration when performing VLAD encoding. The proposed DCNN model has fifteen layers and is trained using the CASIA-WebFace dataset [26] of 10,548 subjects. VLAD features are encoded by the feature maps coming out of the last convolutional layer of the network. These feature maps contain useful spatial information ignored by average pooling. The spatial information is encoded by adding two spatial coordinate dimensions into the features. In our method, we also use the average pooling features from the last convolutional layer and the output features from the fully-connected layer. Given an image pair with all three types of feature pairs, discriminative metrics learned from the training set are applied to these pairs to compute three similarity scores. Finally, three scores are normalized by different scaling factors and added to obtain the final similarity score for verification.

II. Proposed Method

In the training phase, each training image is first passed through a pre-trained DCNN model. Three types of features, the convolutional features (*conv* features) from the last convolutional layer, the average pooled features from the last convolutional layer (*pool* features) and the output features of the fully-connected layer (*fc* features) are extracted. Then, the K-means clustering algorithm is applied on the convolutional features *conv*. VLAD encoding is performed over these local convolutional features to obtain *conv_vlad*. Distance measures are learned for each of these features. In the testing phase, we extract the DCNN features *conv*, *pool* and *fc* and use the learned cluster centers to perform VLAD feature encoding to get *conv_vlad*. We then apply the learned metrics to compute the similarity scores for the three types of features. The final similarity scores are obtained by fusing these features. In what follows, we describe the details of each of these components.

DCNN Network: The DCNN features used in this work are extracted from a deep convolutional neural network with 15 convolutional layers, 5 pooling layers and 2 fully connected layers as shown in Table I. The model is trained using the CASIA-WebFace dataset which contains 10,575 subjects with 494,414 images. (*i.e.*, the images of 10,548 subjects are used for training after removing the overlapping subjects between the CASIA-WebFace and IJB-A datasets.). We use the parametric ReLU (PReLU) [11] as the nonlinear activation function which allows negative responses and usually improves the network performance. The dimensionality of the input layer is $100 \times 100 \times 3$ for the RGB images. We use the output of conv53 layer as the *conv* features. The *pool* features are the output of pool5 layer and the *fc* features are the output of fc6 layer.

VLAD encoding: Since the *pool* features are the average of the *conv* features from the last convolutional layer, it contains the global discriminative information of the appearance with noise reduced by the averaging operation. Each entry of the *fc* features which is the output of the fully-connected layer, shows how the input image looks like the corresponding person in the external training set. Different features in *conv* feature maps correspond to different part of the face. Even though the receptive fields of high level convolutional layers are largely overlapped, especially for deep networks, spatial information is still preserved in *conv* feature maps. Therefore, we use a feature encoding method to incorporate this important information from a feature map into a discriminative feature.

VLAD is a feature encoding and pooling method introduced in [13]. It encodes a set of local features into a high-dimensional vector using the clustering centers provided by methods like the K-means algorithm. For the kth cluster center $\boldsymbol{\mu}_k$, the corresponding VLAD feature is calculated as the sum of the residuals as

$$\mathbf{v}_k = \sum_{i=1}^{N} \alpha_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k) \tag{1}$$

TABLE I
The architecture of DCNN model used in this paper.

Name	Type	Filter Size/Ouput/Stride	#Params
Conv11	convolution	3×3 / 32 / 1	0.28K
Conv12	convolution	3×3 / 64 / 1	18K
Conv13	convolution	3×3 / 64 / 1	36K
Pool1	max pooling	2×2 / 2	
Conv21	convolution	3×3 / 64 / 1	36K
Conv22	convolution	3×3 / 128 / 1	72K
Conv23	convolution	3×3 / 128 / 1	144K
Pool2	max pooling	2×2 / 2	
Conv31	convolution	3×3 / 96 / 1	108K
Conv32	convolution	3×3 / 192 / 1	162K
Conv33	convolution	3×3 / 192 / 1	324K
Pool3	max pooling	2×2 / 2	
Conv41	convolution	3×3 / 128 / 1	216K
Conv42	convolution	3×3 / 256 / 1	288K
Conv43	convolution	3×3 / 256 / 1	576K
Pool4	max pooling	2×2 / 2	
Conv51	convolution	3×3 / 160 / 1	360K
Conv52	convolution	3×3 / 320 / 1	450K
Conv53	convolution	3×3 / 320 / 1	900K
Pool5	avg pooling	7×7 / 1	
Dropout	dropout (40%)		
Fc6	fully connection	10548	3305K
Cost	softmax		
total			6995K

where $\{\mathbf{x}_i\}$ is the set of local features from an image I, α_{ik} is the association of data $\mathbf{x}_i$ to $\boldsymbol{\mu}_k$ with $\alpha_{ik} \geq 0$ and $\sum_{k=1}^{K} \alpha_{ik} = 1$. For hard association, we simply find the nearest neighbor of $\mathbf{x}_i$ among centers $\{\boldsymbol{\mu}_k\}$. As a result,

$$\alpha_{ik} = \begin{cases} 1 & \text{if } \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2 \leq \|\mathbf{x}_i - \boldsymbol{\mu}_l\|_2 \;\forall\; l \neq k \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Then, the overall VLAD feature $\boldsymbol{\Phi}(I)$ for image I is stacked by the residuals for each center as $\boldsymbol{\Phi}(I) = [\cdots \mathbf{v}_k^T \cdots]^T$.

As shown in [24], spatially encoded local features are useful for face verification. Thus, we augment the original *conv* features with the normalized x and y coordinates as $[\mathbf{a}_{xy}, \frac{x}{w} - \frac{1}{2}, \frac{y}{h} - \frac{1}{2}]^T$, where $\mathbf{a}_{xy}$ is the DCNN descriptor at (x, y), and w and h are the width and height of the *conv* feature map, respectively. By adding the two augmented dimensions, the clustering method will not only cluster the training features in the feature space, but also consider their spatial relationships. The features that are closer in spatial domain will be more likely to be clustered together. Features that are far away will be more likely to be assigned to different clusters.

We chose VLAD as the feature encoding method due to its model simplicity compared to other bag-of-word approaches like FV. It only involves the K-means clustering procedure and a nearest neighbor procedure for hard assignment to a cluster which can be done efficiently using a k-d tree. After finding the nearest neighbor for each feature, the encoding is simply a summation of the feature residues.

Metric Learning: After obtaining the encoded *conv_vlad* features by VLAD, because of their high dimensionality, it is important to project them into a lower-dimensional discriminative space, or learn a similarity metric that is as discriminative as possible. The same metric learning procedure is needed for two other types of features, *pool* and *fc*. For the face verification task, the standard protocol defines the same and different pairs for comparison which can be used to train a discriminative similarity function to improve the verification performance. In this work, we mainly focus on learning two kinds of metrics based on triplet distance embedding method and the joint Bayesian (JB) method.

The triplet distance embedding has been widely used in the literature for different applications [22]. This embedding is obtained by solving the following optimization problem

$$\underset{\mathbf{W}}{\operatorname{argmin}} \sum_{\mathbf{x}_a,\mathbf{x}_p,\mathbf{x}_n\in\mathbb{T}} \max\{0, \alpha + (\mathbf{x}_a - \mathbf{x}_p)^T\mathbf{W}^T\mathbf{W}(\mathbf{x}_a - \mathbf{x}_p) - (\mathbf{x}_a - \mathbf{x}_n)^T\mathbf{W}^T\mathbf{W}(\mathbf{x}_a - \mathbf{x}_n)\}, \quad (3)$$

where $\mathbf{x}_a$, $\mathbf{x}_p$ and $\mathbf{x}_n$ are the anchor feature, positive feature and negative feature in the training triplet set $\mathbb{T}$, respectively. The idea of this embedding is to maximize the gap of the Euclidean distance between the positive pair and the negative pair with the same anchor in a triplet in the embedded space. The optimization problem can be solved using the Stochastic Gradient Descent (SGD) method and the corresponding update step is given by

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta\mathbf{W}_t[(\mathbf{x}_a - \mathbf{x}_p)(\mathbf{x}_a - \mathbf{x}_p)^T + (\mathbf{x}_a - \mathbf{x}_n)(\mathbf{x}_a - \mathbf{x}_n)^T] \quad (4)$$

when the update criterion $\alpha + (\mathbf{x}_a - \mathbf{x}_p)^T\mathbf{W}_t^T\mathbf{W}_t(\mathbf{x}_a - \mathbf{x}_p) - (\mathbf{x}_a - \mathbf{x}_n)^T\mathbf{W}_t^T\mathbf{W}_t(\mathbf{x}_a - \mathbf{x}_n) > 0$ is met. Here, we use a hard negative mining strategy introduced in [22]. Given any anchor feature, the negative feature is chosen as the closest feature in the embedded space to the anchor feature among a random subset of the negative candidates, which is $\mathbf{x}_n = \operatorname{argmin}_{\mathbf{x}\in\mathcal{C}(\mathbf{x}_a)} \|\mathbf{x}_a - \mathbf{x}_n\|_2$, where $\mathcal{C}(\mathbf{x}_a)$ is a random subset of the negative candidates of anchor $\mathbf{x}_a$. Given a testing pair $\mathbf{x}_i$ and $\mathbf{x}_j$, the similarity score is the squared Euclidean distance between two features in the embedded space, which is

$$s(i,j) = \|\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j)\|_2^2 = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W}^T\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j). \quad (5)$$

Another metric learning method we use is the JB approach, which has been widely used in the literature for face verification tasks [3][2]. We directly optimize the JB distance measure in a large-margin framework and update the model parameters using SGD as follows

$$\underset{\mathbf{W},\mathbf{V},b}{\operatorname{argmin}} \sum_{i,j} max[1 - y_{ij}(b - (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W}^T\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j) + 2\mathbf{x}_i^T\mathbf{V}^T\mathbf{V}\mathbf{x}_j), 0] \quad (6)$$

where $\mathbf{W}$ and $\mathbf{V} \in \mathbb{R}^{d\times D}$ with d and D as the dimensionality before and after dimension reduction. $b \in \mathbb{R}$ is the threshold, and y_{ij} is the label of a pair: $y_{ij} = 1$ if person i and j are the same and $y_{ij} = -1$, otherwise. Then, one can update $\mathbf{W}$ and $\mathbf{V}$ using the SGD method. The update equations are given as follows:

$$\begin{aligned}
\mathbf{W}_{t+1} &= \begin{cases} \mathbf{W}_t, & \text{if } y_{ij}(b_t - d_{\mathbf{W}_t,\mathbf{V}_t}(\mathbf{x}_i,\mathbf{x}_j)) > 1 \\ \mathbf{W}_t - \gamma y_{ij}\mathbf{W}_t\mathbf{\Gamma}_{ij}, & \text{otherwise,} \end{cases} \\
\mathbf{V}_{t+1} &= \begin{cases} \mathbf{V}_t, & \text{if } y_{ij}(b_t - d_{\mathbf{W}_t,\mathbf{V}_t}(\mathbf{x}_i,\mathbf{x}_j)) > 1 \\ \mathbf{V}_t + 2\gamma y_{ij}\mathbf{V}_t\mathbf{\Lambda}_{ij}, & \text{otherwise,} \end{cases} \\
b_{t+1} &= \begin{cases} b_t, & \text{if } y_{ij}(b_t - d_{\mathbf{W}_t,\mathbf{V}_t}(\mathbf{x}_i,\mathbf{x}_j)) > 1 \\ b_t + \gamma_b y_{ij}, & \text{otherwise,} \end{cases}
\end{aligned} \quad (7)$$

where $d_{\mathbf{W},\mathbf{V}}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W}^T\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j) - 2\mathbf{x}_i^T\mathbf{V}^T\mathbf{V}\mathbf{x}_j$, $\mathbf{\Gamma}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$, $\mathbf{\Lambda}_{ij} = \mathbf{x}_i\mathbf{x}_j^T + \mathbf{x}_j\mathbf{x}_i^T$ and γ is the learning rate for $\mathbf{W}$ and $\mathbf{V}$, and γ_b for the bias b. We use identity matrix to initialize both $\mathbf{W}$ and $\mathbf{V}$ if $d = D$. Otherwise, the projection matrix $\mathbf{P} \in \mathbb{R}^{d\times D}$ for dimension reduction is used for initialization. Both $\mathbf{W}$ and $\mathbf{V}$ are updated only when the constraints are violated. Given a testing pair $\mathbf{x}_i$ and $\mathbf{x}_j$, the similarity score is calculated as

$$s(i,j) = b - (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{W}^T\mathbf{W}(\mathbf{x}_i - \mathbf{x}_j) + 2\mathbf{x}_i^T\mathbf{V}^T\mathbf{V}\mathbf{x}_j. \quad (8)$$

Fusion: In our experiments, we observed that the error patterns for *pool* features, *fc* features, and *conv_vlad* features are different and these features appear to contain complementary information about the face image. As a result, we apply a score level fusion method to fuse these features. We simply sum up the scaled similarity scores obtained from two or more different methods. The scaling factor is important here because the scale of the similarity scores calculated by different metric learning techniques, or even the same technique with different learning parameters, will differ.

For the JB metric, the similarity score is computed by (8). Then, given the matrices $\mathbf{W}$ and $\mathbf{V} \in \mathbb{R}^{d\times D}$ during testing, we calculate

$$scale(\mathbf{W},\mathbf{V}) = \frac{1}{D^2}\|\mathbf{W}^T\mathbf{W}\|_2 + \frac{1}{D^2}\|\mathbf{V}^T\mathbf{V}\|_2. \quad (9)$$

For triplet distance embedding, the similarity score is computed by (5). Then, given the projection matrix $\mathbf{W} \in \mathbb{R}^{d\times D}$ during testing, we calculate

$$scale(\mathbf{W}) = \frac{1}{D^2}\|\mathbf{W}^T\mathbf{W}\|_2 \quad (10)$$

Finally, given two similarity scores of the same pair, s_1 and s_2 from two different models $\mathbf{Model}_1$ and $\mathbf{Model}_2$, the fused similarity score is given by $s_{fusion} = s_1/scale(\mathbf{Model}_1) + s_2/scale(\mathbf{Model}_2)$.

III. Experimental Results

In this section, we evaluate the performance of our method on the recently introduced IARPA Janus Benchmark A (IJB-A) [15] and its extension JANUS Challenge Set 2 (JANUS CS2) unconstrained face verification datasets. Both the IJB-A and JANUS CS2 contain 500 subjects with 5,397 images and 2,042 videos. The IJB-A evaluation protocol consists of verification (1:1 matching) and identification (1:N search). For verification, each of the 10 splits contains around 11,748 pairs of templates with 1,756 positive and 9,992 negative pairs on average. For identification, the protocol also consists of 10 splits which evaluates the search performance. On the other hand, in JANUS CS2, there are about 167 gallery templates and 1763 probe templates. They are used for both identification and verification. The training set for both IJB-A and JANUS CS2 contains 333 subjects, while the test set contains 167 subjects.

In the experiments, each face image is first detected and aligned using the Hyperface method introduced in [21], which

Fig. 2. IJB-A examples. Left 4 are positive pairs and right 4 are negative pairs.

is a multi-task DCNN network that can simultaneously perform face detection, fiducial extraction and gender classification on an input image. Each face is aligned into the canonical coordinate using the similarity transform and seven landmark points (i.e. two left eye corners, two right eye corners, nose tip, and two mouth corners). After alignment, the face image resolution is 125 $\times$ 125 $\times$ 3 pixels. It is then resized into 100 $\times$ 100 $\times$ 3 RGB image and fed into the proposed DCNN network.

The proposed DCNN model is trained with 10,548 subjects and 490,356 face images from the CASIA-WebFace dataset [26] using caffe [14], without finetuning on the JANUS training set. The data is augmented with horizontally flipped faces. For training, we use 128 as the batch size, set the initial negative slope for PReLU to 0.25, and set the weight decay of all convolutional layers to 0 and of the final fully connected layer to 5e-4. Finally, the learning rate is initially set equal to 1e-2 and reduced by half for every 100,000 iterations. The momentum is set equal to 0.9. The snapshot of 720,000th iteration is used for all our experiments.

For each image, the *pool* feature from pool5 layer is 320 dimensional. The *fc* feature from fc6 layer is 10548 dimensional, and the intermediate *conv* feature map from conv53 layer is 7 $\times$ 7 $\times$ 320. The *conv* features from the training set are normalized after taking the square root (with sign preserved). Two additional dimensions are added as extra spatial information. K-means clustering is then applied on the normalized and augmented features with K=16. The features are encoded using the VLAD technique with $(320+2)\times 16 = 5152$ dimensions. After *conv_vlad*, *pool* and *fc* features are extracted, media averaging is applied so that the features coming from the same media (image or video) are averaged.

The training data used for metric learning is the JANUS training set only. Both JB and triplet distance embedding metrics are learned for *conv_vlad* features. Before metric learning, the high dimensional VLAD features are first projected onto a 200-dimensional space by the matrix $\mathbf{P}$ (200 $\times$ 5152) learned using the whitening Principle Component Analysis (WPCA). For JB, the learned matrices $\mathbf{W}$ and $\mathbf{V}$ are both 200 $\times$ 200 ($d = D = 200$). The learning rates γ and γ_β are both set to 1e-2. The margin α is 1e-3. The proportion between positive pairs and negative pairs in the training set is 1:1. For triplet embedding, the learned projection matrix $\mathbf{W}$ is also 200×200. The learning rate γ and margin α are both 1e-3. Hard negatives are chosen from 100 randomly picked negatives for a given anchor. We call the scores obtained by triplet embedding as **A**, and the scores obtained by the JB metric learning as **B**.

For both *pool* and *fc* features, 128-dimensional triplet embeddings are learned. The learning hyperparameters are the same as **A**. We call the scores obtained from *pool* after triplet embedding as **C** and the scores obtained from *fc* after triplet embedding as **D**. Finally, we fused **A** with **D** and **B** with **C**, using the scaling factor calculated by (9) and (10).

For comparison, we also do the same experiment using FV encoding [24] rather than VLAD encoding. Similar to VLAD, for FV we learned a 16-component GMM. The FVs (say *conv_fv*) are computed from *conv* feature maps after square root normalization and spatial encoding. The encoded FVs are of $(320+2)\times 32 = 10304$ dimensions. Triplet embedding with 200 dimensions is learned with the same hyperparameters as **A**, **C** and **D**. We denote this result by **E**.

A baseline method is also implemented. It is based on a 10-layer DCNN network introduced in [4] which is fine tuned on the JANUS training set. For face alignment, the fiducial extraction method proposed in [17] is used. A triplet similarity embedding method introduced in [22] is applied on the pool5 output features of the network to produce the final similarity scores. We denote this method by **F**. We also compare our methods with two recent methods [1] and [18]. The verification and identification results corresponding to different methods on the CS2 and IJB-A datasets are shown in Table II, III and Figure 3.

To clarify the notation again, in the following tables and figures, **A** is *conv_vlad* with triplet embedding. **B** is *conv_vlad* with JB metric. **C** is *pool* with triplet embedding. **D** is *fc* with triplet embedding. **E** is *conv_fv* with triplet embedding and **F** is the baseline method. **B+C** and **A+D** correspond to two kinds of score level fusion.

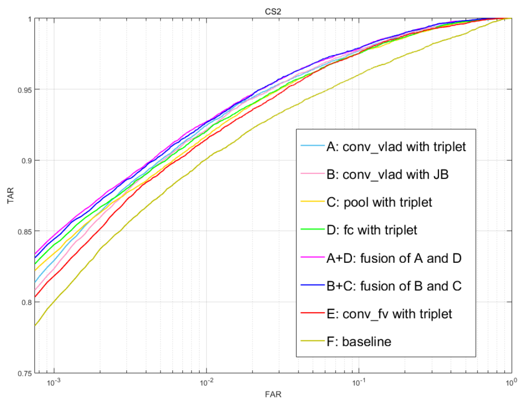
TABLE II
CS2 AND IJB-A VERIFICATION RESULTS

CS2 and IJB-A Verification Results						
FAR	CS2			IJB-A (1:1)		
	1e-3	1e-2	1e-1	1e-3	1e-2	1e-1
A	82.92%	92.44%	97.71%	73.64%	87.65%	96.16%
B	82.34%	92.14%	97.76%	73.31%	87.11%	96.17%
C	83.42%	91.71%	97.53%	77.09%	88.21%	96.18%
D	84.04%	92.05%	97.52%	**77.88%**	88.70%	96.22%
B+C	84.43%	92.66%	**97.90%**	76.62%	88.70%	96.56%
A+D	**84.69%**	**92.72%**	97.85%	77.36%	**88.85%**	**96.66%**
E	81.83%	91.46%	97.53%	72.94%	86.63%	95.80%
F	80.02%	90.09%	96.02%	72.05%	85.52%	94.54%
[1]	-	89.7%	95.9%	-	78.7%	91.1%
[18]	82.4%	92.6%	-	72.5%	88.6%	-

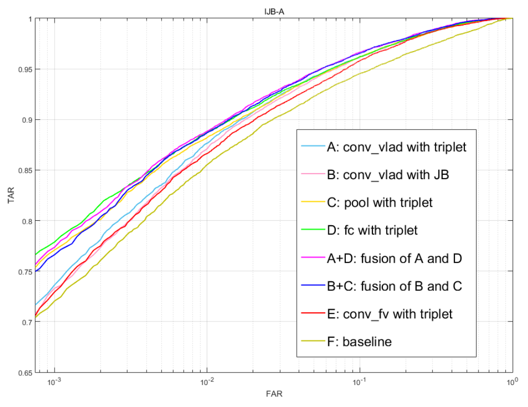
TABLE III
CS2 AND IJB-A IDENTIFICATION RESULTS

CS2 and IJB-A Identification Results						
FAR	CS2			IJB-A		
	rank 1	rank 5	rank 10	rank 1	rank 5	rank 10
A	89.20%	94.80%	96.00%	90.40%	95.30%	96.30%
B	89.10%	94.70%	95.90%	90.40%	95.20%	96.20%
C	89.30%	94.60%	95.90%	90.50%	95.20%	96.50%
D	89.70%	94.70%	96.00%	90.90%	95.30%	96.60%
B+C	89.90%	95.00%	96.40%	91.00%	95.70%	96.80%
A+D	**90.20%**	95.10%	96.40%	**91.30%**	95.60%	96.90%
E	88.80%	94.60%	96.10%	90.00%	95.20%	96.60%
F	87.40%	93.70%	95.30%	88.31%	94.60%	96.00%
[1]	86.5%	93.4%	94.9%	84.6%	92.7%	94.7%
[18]	89.8%	**95.6%**	**96.9%**	90.6%	**96.2%**	**97.7%**

From the tables and curves we can see that before fusion, **D** has the best IJB-A verification result. The best CS2 at 1e-2



(a)



(b)

Fig. 3. Results on the JANUS CS2 and IJB-A datasets. (a) the average ROC curves for the JANUS CS2 verification protocol and (b) the average ROC curves for IJB-A verification protocol over 10 splits.

result is achieved by **A**, which implies that VLAD encoding does extract more information from the features maps of the last convolutional layer, instead of directly doing average pooling. Its IJB-A performance is also comparable with **C** and **D**. After fusing **B+C**, CS2 at 1e-2 increases about 0.9% from **C**. IJB-A at 1e-2 also has a 0.5% gain, which shows the the effectiveness of our scaled fusion strategy. After fusing **A+D**, we obtain the best results on both CS2 at 1e-2 and IJB-A at 1e-2. Also, **A** performs better than **E** at both CS2 1e-2 and IJB-A 1e-2 with a gap of about 1%. All of the above results show that based on DCNN features in this scenario, VLAD encoding is very competitive and is superior to FV encoding.

Also, it is obvious that the performance of **F** is much lower than others based on features from the new 15-layer network. This is because the 15-layer network is five layers deeper than the one presented in [4]. Also the new features are extracted from the face images aligned using Hyperface [21], which is a more powerful face detection and fiducial extraction method. Thus finetuning is not that effective when the alignment is of high quality.

Compared to [1] and [18], our methods performs consistently better for verification task on both CS2 and IJB-A. For identification task at Rank 5 and 10, our performance is slightly lower but still comparable to [18]. It is because in [18] the CASIA WebFace dataset is expanded to over 2.4 Million images for training using 3D synthesized image. But our model is trained using the original CASIA dataset without any augmentation.

The reason that FV does not perform as well as VLAD using the DCNN features is that the 2nd order statistics are not useful in this scenario. The bag-of-words method is usually designed for low-level local features like SIFT, SURF or HoG, which are basically histograms. In these cases, for a set of histograms of local features, both the 1st order (mean of the histograms) and the 2nd order (variance of every entry of the histograms) statistics contain discriminative information. But for DCNN features, the 2nd order statistics are much less important than the 1st order ones. Different from the traditional local features, the DCNN features extracted from the high level layers themselves are already very discriminative features. As mentioned above, they are not as local as the traditional local features since their receptive fields in the input image are getting bigger as the network is getting deeper. Therefore, the variance of the set of DCNN features from the same image is more likely to contain noise than useful information since these DCNN features are from receptive fields with large overlaps. When computing FV, we need to scale each entry of the feature according to its variance and aggregate these shifted and scaled features together as

$$\mathbf{\Phi}_{ik}^{(1)} = \frac{1}{N\sqrt{w_k}} \sum_{p=1}^{N} \alpha_k(\mathbf{v}_p) \left(\frac{\mathbf{v}_{ip} - \boldsymbol{\mu}_{ik}}{\boldsymbol{\sigma}_{ik}} \right), \quad (11)$$

$$\mathbf{\Phi}_{ik}^{(2)} = \frac{1}{N\sqrt{2w_k}} \sum_{p=1}^{N} \alpha_k(\mathbf{v}_p) \left(\frac{(\mathbf{v}_{ip} - \boldsymbol{\mu}_{ik})^2}{\boldsymbol{\sigma}_{ik}^2} - 1 \right). \quad (12)$$

If the variances are not reliable enough, it will degrade the performance.

In contrast, since the DCNN features are robust and discriminative, the 1st order statistics still contain important information (even more robust after taking the average over the neighborhood). VLAD only considers the 1st order statistics and will not be affected by the noise variance. Thus compared to FV encoding, VLAD encoding preserves useful information.

To examine the above assumption, we design another experiment based on the verification protocols of CS2 and IJB-A Split 1. We first learn a GMM of 16 components based on the training set of Split 1. Then we randomly choose one position in the 7×7 feature map of *conv* features. Given an image, instead of average pooling the 49 320-dimensional local features or performing VLAD encoding to get the output features, we directly pick the 320×1 local feature at the chosen position from the $7 \times 7 \times 320$ *conv* features and consider it as the representation of this image. In this way, every image is directly represented by the local features at a certain position in the feature map. Then we encode them in two ways. One follows VLAD encoding by subtracting the features by their nearest GMM mean as $\mathbf{x}_{vlad} = \mathbf{x} - \mathbf{m}_{nn}$ without encoding the variance information. The other method mimics the FV encoding by subtracting the features by their nearest GMM mean and dividing by the corresponding standard deviation, which is $\mathbf{x}_{fv} = (\mathbf{x} - \mathbf{m}_{nn})/\boldsymbol{\sigma}_{nn}$. The verification results are calculated on CS2 and IJB-A Split 1 using the cosine distance. The experiments are repeated 10 times. Then we average the results for both encoding methods.

The objective of this experiment is to see whether encoding the 2nd order statistics of our DCNN features will reduce the quality of these local features. Since the FV feature is the aggregation of encoded local features, if the performance of encoded local features decreases, it will very likely affect the performance of FV features. We evaluated the performance of each set of these encoded local features with cosine distance. The verification results averaged over 10 sets of local features on CS2 and IJB-A Split 1 are shown in Table IV.

TABLE IV
CS2 and IJB-A Verification Results of Encoded Local Features

CS2 and IJB-A Verification Results of Encoded Local Features						
FAR	CS2			IJB-A (1:1)		
	1e-3	1e-2	1e-1	1e-3	1e-2	1e-1
$\mathbf{x}_{vlad}$	**50.33%**	**67.77%**	**84.22%**	**41.26%**	**62.26%**	**80.07%**
$\mathbf{x}_{fv}$	49.71%	67.43%	84.18%	40.63%	61.55%	79.81%

From the table we can see that the VLAD-like encoded local DCNN features perform consistently better than FV-like encoded local DCNN features, which supports our assertion that the 2nd order statistics of our DCNN features contain little discriminative information. It also explains why FV features' performance is not as good as VLAD features' in the dataset we used. But since there are different types of DCNN architectures, we need to perform more experiments to check if our assumption is also true for features from the other DCNN networks. This is left for future work.

IV. Conclusion

In this paper, we introduced a DCNN-based approach for face verification using VLAD feature encoding, two types of metric learning techniques and score level fusion. The experiments on the challenging JANUS dataset show the effectiveness of VLAD encoding of DCNN features and the superior performance after score level fusion. We also compare the performance of VLAD and FV encoding on the JANUS dataset. What we can conclude is that VLAD encoding works better than FV encoding in our senario because of the noisy 2nd order statistics used by FV. For future work, we plan to design an end-to-end VLAD-DCNN network with the siamese structure for face verification. Also, the analysis of VLAD and FV encoding for DCNN features from other deep network architectures will be continued, which will help understand and introduce new feature pooling techniques.

References

[1] W. AbdAlmageed, Y. Wu, S. Rawls, S. Harel, T. Hassner, I. Masi, J. Choi, J. Lekust, J. Kim, P. Natarajan, R. Nevatia, and G. Medioni. Face recognition using deep multi-pose representations. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, Lake Placid, NY, 2016.

[2] X. D. Cao, D. Wipf, F. Wen, G. Q. Duan, and J. Sun. A practical transfer learning algorithm for face verification. In *IEEE International Conference on Computer Vision*, pages 3208–3215. IEEE, 2013.

[3] D. Chen, X. D. Cao, L. W. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *European Conference on Computer Vision*, pages 566–579. 2012.

[4] J. Chen, V. M. Patel, and R. Chellappa. Unconstrained face verification using deep CNN features. *CoRR*, abs/1508.01722, 2015.

[5] J.-C. Chen, S. Sankaranarayanan, V. M. Patel, and R. Chellappa. Unconstrained face verification using fisher vectors computed from frontalized faces. In *IEEE International Conference on Biometrics: Theory, Applications and Systems*, 2015.

[6] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision*, pages 1–30, 2016.

[7] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[9] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. *Multi-scale Orderless Pooling of Deep Convolutional Activation Features*, pages 392–407. Springer International Publishing, Cham, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[12] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010.

[13] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1704–1716, Sept. 2012.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[15] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[17] A. Kumar, R. Ranjan, V. M. Patel, and R. Chellappa. Face alignment by local deep descriptor regression. *CoRR*, abs/1601.07950, 2016.

[18] I. Masi, A. T. an Trãn, J. T. Leksut, T. Hassner, and G. Medioni. Do we really need to collect millions of faces for effective face recognition? *arXiv preprint arXiv:1603.07057*, 2016.

[19] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[20] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. *British Machine Vision Conference*, 2015.

[21] R. Ranjan, V. M. Patel, and R. Chellappa. HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition, Mar. 2016.

[22] S. Sankaranarayanan, A. Alavi, and R. Chellappa. Triplet similarity embedding for face verification. *CoRR*, abs/1602.03418, 2016.

[23] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*, 2015.

[24] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In *British Machine Vision Conference*, 2013.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[26] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *CoRR*, abs/1411.7923, 2014.

[27] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *European Conference on Computer Vision*, pages 141–154, 2010.