

# Face Recognition Using Deep Multi-Pose Representations

Wael AbdAlmageed<sup>a,0</sup> Yue Wu<sup>a,0</sup> Stephen Rawls<sup>a,0</sup> Shai Harel<sup>c</sup> Tal Hassner<sup>a,c</sup>  
Iacopo Masi<sup>b</sup> Jongmoo Choi<sup>b</sup> Jatuporn Toy Leksut<sup>b</sup> Jungyeon Kim<sup>b</sup> Prem Natarajan<sup>a</sup>  
Ram Nevatia<sup>b</sup> Gerard Medioni<sup>b</sup>

<sup>a</sup>Information Sciences Institute  
University of Southern California  
Marina Del Rey, CA

<sup>b</sup>Institute for Robotics and Intelligent Systems  
University of Southern California  
Los Angeles, California

<sup>c</sup>The Open University  
Raanana, Israel

## Abstract

*We introduce our method and system for face recognition using multiple pose-aware deep learning models. In our representation, a face image is processed by several pose-specific deep convolutional neural network (CNN) models to generate multiple pose-specific features. 3D rendering is used to generate multiple face poses from the input image. Sensitivity of the recognition system to pose variations is reduced since we use an ensemble of pose-specific CNN features. The paper presents extensive experimental results on the effect of landmark detection, CNN layer selection and pose model selection on the performance of the recognition pipeline. Our novel representation achieves better results than the state-of-the-art on IARPA's CS2 and NIST's IJB-A in both verification and identification (i.e. search) tasks.*

## 1. Introduction

Face recognition has been one of the most challenging and attractive areas of computer vision. The goal of face recognition algorithms is to answer the question, *who is this person in a given image or video frame?* Face recognition algorithms generally try to address two problems — identify verification and subject identification. Face verification, as known as the 1 : 1 matching problem [11], answers the question, *are these two people actually the same?*, while face identification, also known as the 1 : N problem [11], answers the question, *who is this person, given a database of faces?*

Labeled Faces in the Wild (LFW) dataset [8] is considered one of the most important benchmarks for face recognition research. Recent advances, especially in applying convolutional neural networks (CNN) to face recognition, enabled researchers of achieving close to 100% recognition rates [6]. However, face recognition problem is far from solved, especially in an uncontrolled environment with extreme pose, illumination, expression and age variations. Indeed, as discussed in [11], state-of-the-art commercial and open-source face recognition systems performed far less than satisfactory on the recently released National Institute of Standards and Technology's (NIST) IARPA Janus Benchmark-A (IJB-A), which is considered much more challenging, than LFW, in terms of the variability in pose, illumination, expression, aging, resolution, etc. [19, 15, 5].

IJB-A dataset has been quickly adopted by the research community. In [19], the authors represent a face image as a feature vector using a trained convolutional neural network and hash this feature vector to achieve fast face search. Chowdhury et al. in [15] fine-tunes a trained base-model of a symmetric bilinear convolutional neural network (BCNN) to extract face features, and trains subject-based SVM classifiers to identify individuals. In [5], Patel et al. use a trained CNN model to represent a face, and additional joint-Bayesian metric learning to assess the similarity between two face representations.

In this paper we present our face recognition pipeline using a novel multi-pose deep face representation. Unlike previous efforts [15, 5, 19] that consider pose variations implicitly, we explicitly leverage the variations of face poses by (1) representing a face with different (aligned and rendered) poses using different pose-specific CNNs and (2) perform face similarity comparisons only using same pose

<sup>0</sup>Equal contributors and corresponding authors at wamageed, yue.wu, rawls@isi.edu

CNNs. To our best knowledge, this is the first attempt to use multi-pose in face recognition. This novel approach is applied to IJB-A dataset and is shown to surpasses the state-of-the-art algorithms [15, 5, 19], without additional domain adaptation or metric learning.

The remainder of this paper is organized as follows. Section 2 discusses the training and testing datasets. In Section 3 we discuss our conceptual single representation along with instances of different pose experts, and the proposed multi-pose representation for face recognition. Evaluation protocols and results of the proposed recognition pipeline are presented in Section 4. We conclude the paper in Section 5.

## 2. Facial Datasets Employed

We use CASIA-WebFace [22] for training and both IJB-A [11] and IARPA’s Janus CS2 for evaluation. Following is a brief description of these datasets.

CASIA-WebFace [22] dataset is the largest known public dataset for face recognition. Specifically, CASIA-WebFace contains 10,575 subjects with a total of 494,414 images. We perform the following data clean up steps before using WebFace for training our pose-specific CNN models — (1) exclude images of all subjects in WebFace that overlap with the testing and evaluation data sets, (2) remove all images of subject with fewer than five images in the dataset and (3) remove images with undetectable faces. Approximately 400,000 images for 10,500 subjects remain of which we use 90% and 10% for training and validation, respectively, of the CNNs.

The objective of IARPA’s Janus program is push the frontiers of unconstrained face recognition. Janus datasets contains images that have full-pose variations, as illustrated by the pose distribution histogram in Figure 2, and provide manual annotations, including facial bounding box, seed landmarks of two-eye and nose base, lighting conditions and some subject attributes, such as ethnicity, gender, etc. These datasets introduce the novel concept of a *template*, which is a collection of images and videos of the same subject. IJB-A and CS2 share 90% of image data. However, they differ in evaluation protocols. In particular, IJB-A includes protocols for both open-set face identification (i.e. 1 : N search) and face verification (i.e. 1 : 1 comparison), while CS2 focuses on closed-set identification. The datasets are divided into gallery and probe subsets. The gallery subset is a collection of templates for subjects to be enrolled in the system, while the probe subset is a collection of templates for unknown subjects for testing and evaluation purposes.

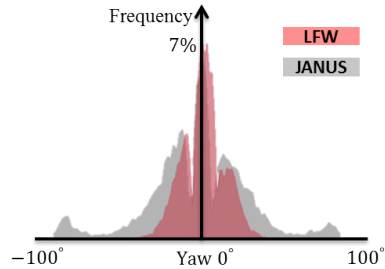


Figure 2: Face pose distribution in LFW and JANUS datasets.

## 3. Face Recognition Pipeline

### 3.1. From Images to Representations

Given an input face image  $X$ , a typical recognition pipeline applies a sequence of processing steps in order to transform the 2D color image into a fixed-dimensional feature vector representation, as shown in Figure 1. To simplify future discussions, we refer to this transformation process as a function  $\text{rep}(\cdot)$ , and the resulting feature representation of an image  $X$  as  $F_X = \text{rep}(X)$ . In the following we discuss the processing steps of  $\text{rep}(\cdot)$ .

#### 3.1.1 Facial Landmark Detection and Face Alignment

A facial landmark detection algorithm,  $\text{lmd}$ , takes a face image  $X$  and estimates  $n$  predefined key points, such as eye corners, mouth corners, nose tip, etc., as shown in Equation (1)

$$\text{lmd}(X) = \begin{bmatrix} P_x^1 & P_y^1 \\ P_x^2 & P_y^2 \\ \vdots & \vdots \\ P_x^n & P_y^n \end{bmatrix} \quad (1)$$

where  $P_x$  and  $P_y$  denote a key point’s coordinate along  $x$  and  $y$  axes, respectively. Depending on the number of defined facial key points, landmarks can be roughly classified into two types — (1) sparse landmark, such as in the 5-point multitask facial landmark dataset [23] and (2) dense landmark, such as in the 68-point 300w dataset [16]. Regardless of the facial landmark detector used, landmark detection produces anchor points of a face as a preprocessing step to face recognition and other face analysis tasks.

Detected landmarks are then used to estimate the roll angel of the face pose. The image is then pose-corrected by rotating the image. Since landmark detection algorithms are often trained with imbalanced datasets, which creates some detection errors, the landmark detection is re-applied to the pose-corrected image, as shown in Figure 1.

In order for downstream processing steps, such as feature extraction and recognition, to work on consistent feature spaces, all face images must be brought to a common

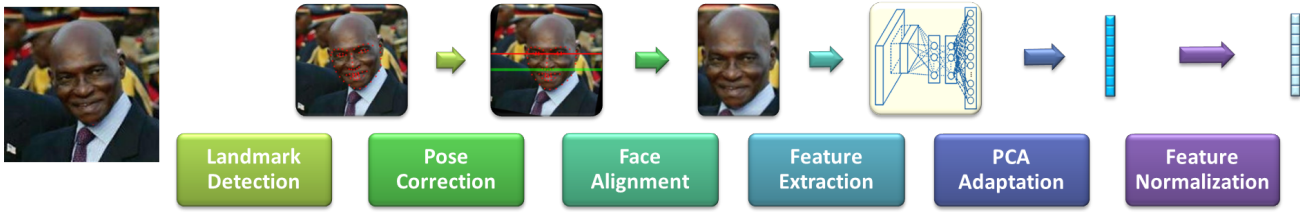


Figure 1: Facial representation pipeline.

coordinate system through face alignment, which reduces pose variations. This is attained by aligning detected landmarks with a set of landmarks from a reference model using distance minimization. If the reference model is in 2D, the process is called in-plane alignment and if it is a 3D model the process is called out-of-plane alignment. In our recognition pipeline, we use both non-reflective similarity transformation for in-plane alignment, and a perspective transformation for out-of-plane alignment.

Specifically, given a set of reference facial landmarks  $\text{lmd}(R)$ , we seek similarity transformation  $T$  to align a face image to the reference model, such that

$$T^* = \underset{T}{\operatorname{argmin}} \|T[\text{lmd}(X) | \mathbf{1}]' - [\text{lmd}(R) | \mathbf{1}]\|_2^2 \quad (2)$$

where  $[\text{lmd}(X) | \mathbf{1}]$  is simply an expansion of  $\text{lmd}(X)$  by adding an all-one vector  $\mathbf{1}$ , and  $T$  is a homogeneous matrix defined by rotation angle  $\theta$ , scaling factor  $s$ , and translation vector  $[t_x, t_y]$ , as shown in Equation(3).

$$T = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Unlike 2D alignment, our 3D alignment relies on a 3D generic face shape model as shown in Figure 3, although we still need the detected facial landmark to estimate initial face shape. Once we successfully fit our generic 3D face shape model to a given face image, we can render face images with arbitrary yaw-pitch-roll parameters (details can be found in [10] and [7]). For example, Figure 3(c) shows rendered face at different yaw and pitch values. Figure 3(d) shows different face images aligned to the same yaw-pitch-roll configuration. Finally, aligned images are cropped to a fixed size of  $160 \times 128$  pixels, which is used in subsequent recognition steps.

### 3.1.2 Feature Extraction and Domain Adaptation

Feature extraction is an essential module in our representation pipeline, for its responsibility to produce features that provide the power for discriminating between

different subjects. Classically, face recognition systems used hand-crafted features. For example, local binary pattern (LBP) [1] and its variants [20] have been effectively used for face recognition, and Gabor wavelet features have also been widely used [21]. Since many of such hand-crafted features involve hyper-parameters, multi-scale, multi-resolution, and/or multi-orientation features have also been useful for face recognition, although they require more computations and occupy more space. Often, hand-crafted features do not require learning from data. However, with the availability of training data, higher-level feature representations could be learned [17]. This learning process still depends on hand-crafted features, rather than raw data.

In contrast, recent advances in deep learning [19, 15, 5, 6] demonstrate that feature representations from raw data can be learned along with the recognition task, where all layers, except the last one, in a deep neural network (DNN) are considered to be feature learning layers. As a result, one may use DNNs for feature extraction. Details of how we learn feature extraction using DNNs are explained in Section 3.2. For now, we deal with a any feature extractor as a black box that takes an aligned and cropped image as as input and produces a high-dimensional feature vector.

The objective of domain adaptation is to close the gap between training and testing data. We use principal component analysis (PCA) to learn the orthogonal transformation from a testing dataset, such that 95% of its original feature variance is kept, because it does not require any labeling and is bounded by a low computational complexity. Although feature dimension reduction is not the main purpose here, possible noise is dropped along with the discarded feature dimensions. Although one may directly use adapted features for matching, additional normalization can be very helpful, such as  $L_2$  and power normalization [14].

### 3.2. Face Representation in Practice

Although one may implement the discussed conceptual pipeline in many different ways, we believe that the two most important components of a face recognition pipeline



Figure 3: Out-of-plane face alignment via rendering: (a) reference generic 3D face shape; (b) face image with estimated 3D face shape; (c) rendered face at different yaw-and-pitch grids; (d) aligned faces at yaw  $45^\circ$  and pitch  $0^\circ$ .

are alignment and feature extraction. Therefore, we mainly focus on the combination of face alignment and face feature extraction, and study the configurations shown in Table 1. In this table, HDLBP stands for *high-dimensional local binary pattern* [1], and *AlexNet*, *VGG16* and *VGG19* refer to CNN architectures of [12], and config-D and E in [18], respectively; “Ref. Model” denotes the reference model used in face alignment, and “avg-all-face-lmd” means that we use the averaged landmark vector of all training data, while “gene-face-yaw@45” means that we use the generic 3D face model at  $45^\circ$  yaw and  $0^\circ$  pitch (see Figure 3(b)). For HDLBP feature extraction, we compute a patch-based LBP histogram of 6,098 predefined facial key points, and then concatenate these histograms into a feature vector.

Table 1: Face representation pipelines.

Pip. Acronym	Alignment	Ref. Model	Feature	Rep. Dim.
HLBP	in-plane	avg-face	HDLBP	100,000
ALEX-AF	in-plane	avg-all-face-lmd	AlexNet	4,000
ALEX-FF	in-plane	avg-frontal-face-lmd	AlexNet	4,000
ALEX-PF	in-plane	avg-profile-face-lmd	AlexNet	4,000
ALEX-FY0	out-of-plane	gene-face-yaw@0	AlexNet	4,000
ALEX-FY45	out-of-plane	gene-face-yaw@45	AlexNet	4,000
VGG16-AF	in-plane	avg-face-lmd	VGG16	4,000
VGG19-AF	in-plane	avg-all-face-lmd	VGG19	4,000
VGG19-FF	in-plane	avg-frontal-face-lmd	VGG19	4,000
VGG19-PF	in-plane	avg-profile-face-lmd	VGG19	4,000
VGG19-FY0	out-of-plane	gene-face-yaw@0	VGG19	4,000
VGG19-FY45	out-of-plane	gene-face-yaw@45	VGG19	4,000
VGG19-FY75	out-of-plane	gene-face-yaw@75	VGG19	4,000

From this point, we concentrate on the discussion of obtaining deep learning features. CASIA-WebFace is used as our standard data set for training and validating different CNN architectures. We preprocessed the dataset in order to obtain aligned and cropped version of the data prior to training the CNNs.

Regardless of the CNN architecture used, we always use CASIA-WebFace as our training and validation dataset. Depending on the representation pipeline used, we preprocess all CASIA-WebFace data until face alignment. This produces approximately 400,000 samples from 10,500 subjects.

### 3.2.1 Transfer Learning

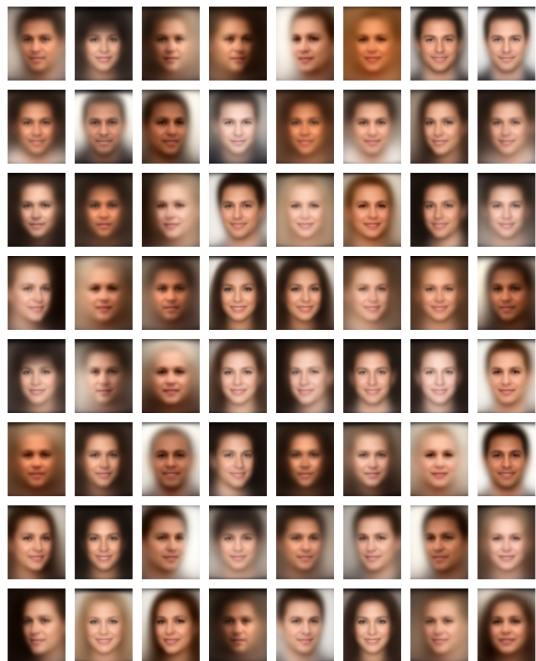
Since the amount of face data in CASIA-WebFace is relatively limited (400,000 images for 10,500 subjects), we initialize our CNNs with pre-trained CNNs using the publicly available models from the ILSVRC2014 image classification task, whose ImageNet dataset contains more than 100 million images for 1000 classes. In order to use a pre-trained model as an initial model for the CASIA-WebFace recognition task, we keep all weights of all CNN layers except those from the last dense layer, since the number of output nodes of the last layer must correspond to the number of subject in WebFace (*i.e.* 10,500) and reinitialize this layer with random weights. After we construct this base model, we begin *transfer learning* process [13] for face recognition in using Caffe library, and obtain new CNN models of ALEX-AF, VGG16-AF, and VGG19-AF, modified to match WebFace subjects and adapted to face feature extraction. Caffe provides pretrained models of AlexNet, VGG16 and VGG19 and automatically performs the required preprocessing, such as resizing an input image to  $224 \times 224$  pixels, subtracting an average image, *etc.* With respect to model training, we use the stochastic gradient descent optimizer with the learning rate starting at  $1e-3$  and reducing it to  $1e-4$  as the convergence plateaus.

To verify the performance of transfer learning, we cluster the output of fc7 layer. Figure 4 shows cluster-wise average faces from the CASIA-WebFace [22] data using raw RGB values and the fc7 layer features of VGG19-AF. It can be seen that VGG19-AF learns many important facial attributes, such as gender, face shape, ethnicity, hair color, *etc.*

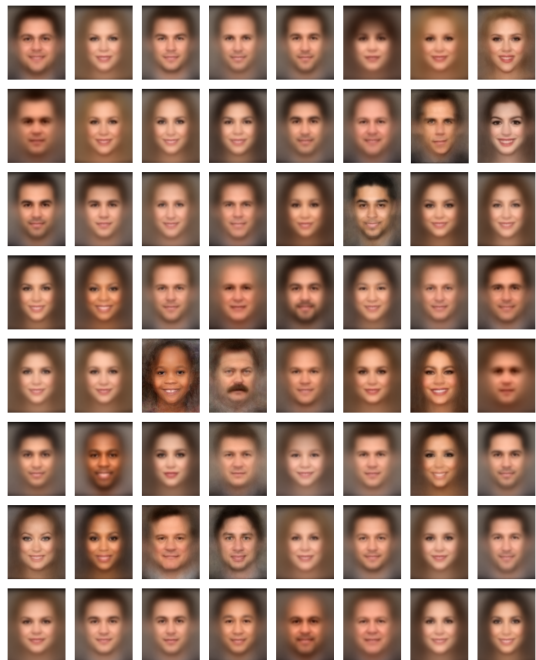
### 3.2.2 Pose-wise Fine Tuning

Once we have our face recognition baseline CNN, we then apply *fine tuning* to learn pose-specific features. For example, we use the ALEX-AF model as our base model, and further train ALEX-FF and ALEX-PF, which focus on near-frontal and near-profile faces, respectively. Based on the ALEX-PF CNN, we can further fine-tune it using ren-





(a)



(b)

Figure 4: GMM 64 Clustering CASIA-WebFace data according to (a) raw RGB-values, and (b) VGG19-AF fc7 feature.

dered faces aligned at  $0^\circ$  yaw, and obtain ALEX-FY0 CNN. The essence behind this CNN training is that we always only move one step forward. For example, we do not use AlexNet as the base model for ALEX-FF because AlexNet’s training data contains objects from different poses.

Similarly, we do not use the near-profile model VGG19-PF as the base model for VGG19-FY0, because the near-frontal CNN model VGG-FF is more appropriate in the sense that rendered faces at  $0^\circ$  yaw are also frontal. Table 2 gives the full set of descriptions of how we learn these CNNs for face recognition. At the end of each fine tuning process, the last (i.e. classification) layer of the CNN is discarded and the CNN is used as a feature extractor by concatenating the outputs of one or more layers of the CNN. Note that we use different versions of preprocessed CASIA-WebFace data to train different CNNs. Specifically, *all real* refers to all CASIA-WebFace images, *real frontal* and *real profile* only refer to those whose yaw angles are close to  $0^\circ$  and  $75^\circ$ , respectively. Figure 5 shows the averaged images of different training partitions.

Table 2: Deep learning features for face recognition

Pip. Acronym	Learning Type	Base CNN Model	Training Partition
ALEX-AF	transfer	AlexNet	all real
ALEX-FF	finetune	ALEX-AF	real frontal
ALEX-PF	finetune	ALEX-AF	real profile
ALEX-FY0	finetune	ALEX-FF	rendered yaw0
ALEX-FY45	finetune	ALEX-PF	rendered yaw45
VGG16-AF	transfer	VGG16	all real
VGG19-AF	transfer	VGG19	all real
VGG19-FF	finetune	VGG19-AF	real frontal
VGG19-PF	finetune	VGG19-AF	real profile
VGG19-FY0	finetune	VGG19-FF	rendered yaw0
VGG19-FY45	finetune	VGG19-PF	rendered yaw45
VGG19-FY75	finetune	VGG19-FY45	rendered yaw75

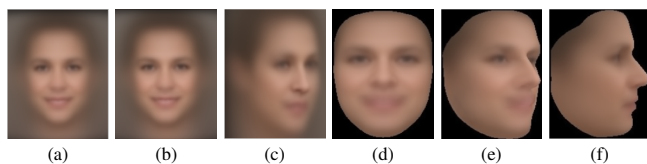


Figure 5: Averaged faces of different training partitions. (a) all real, (b) real frontal, (c) real profile, (d) rendered yaw0, (e) rendered yaw45, and (f) rendered yaw75.

### 3.3. Multi-Modal Representation for Recognition

Without loss of generality, assume that we have  $k$  distinctive representations of a single input facial image  $X$ , namely,

$$R(X) = \{\text{rep}_1(X), \text{rep}_2(X), \dots, \text{rep}_k(X)\}' \quad (4)$$

where each representation is obtained by applying the conceptual pipeline of Figure 1 using a different feature extractor. When this general multi-modal representation only involves pose-specific models, i.e. “fine tuned” models in Table 2, we call this representation a **multi-pose** representation.

Once we define the face representation of Equation (4), we compute the similarity between two face images in two

steps — (1) compare a similarity score between features from the same representation pipeline, and (2) fuse similarities scores across different representation pipelines, as shown in Figure 6. Although one may use various ways

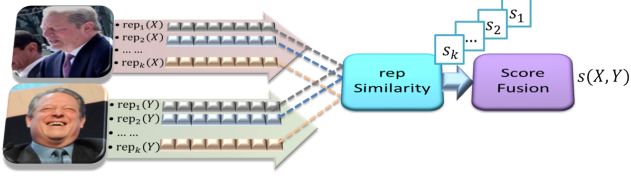


Figure 6: Facial matching overview.

to compute feature similarities and to fuse a set of scores, we simply use direct representation-to-representation (i.e. feature-to-feature) comparison in pair-wise fashion to avoid having to construct new classifiers based on the target (i.e. testing) data set. Specifically, we use Equation (5) to compute the pair-wise image similarity score

$$\text{sim}(X, Y) = \text{fuse}(\{\text{rsim}(\text{rep}_j(X), \text{rep}_j(Y))\}_{j=1}^k) \quad (5)$$

where we use a cosine similarity metric defined in Equation (6) to quantify the similarity between two features, and use softmax weights to fuse different scores, as shown in Equation (7), with the bandwidth parameter  $\beta = 10$ .

$$\text{rsim}(\text{rep}(X), \text{rep}(Y)) = \frac{\langle \text{rep}(X), \text{rep}(Y) \rangle}{\|\text{rep}(X)\| \cdot \|\text{rep}(Y)\|} \quad (6)$$

$$\text{fuse}(\{s_1, s_2, \dots, s_k\}) = \frac{\sum_{i=1}^k s_i \cdot \exp(\beta \cdot s_k)}{\sum_{i=1}^k \exp(\beta \cdot s_k)} \quad (7)$$

As mentioned before, since the IJB-A dataset uses the notion of *templates*, we need to compare the similarity between two templates  $\mathbb{X}$  and  $\mathbb{Y}$  instead of two images. Therefore, we use one more step of softmax fusion over all pair-wise scores from images in two templates, as shown in Equation (8).

$$\text{tsim}(\mathbb{X}, \mathbb{Y}) = \text{fuse}(\{\text{sim}(X, Y) | X \in \mathbb{X}, Y \in \mathbb{Y}\}) \quad (8)$$

### 3.4. Time Complexity

We use a single NVIDIA Tesla K40 GPU for training and testing. Fully training and fine tuning a VGG19-AF-like CNN model takes approximately one week with fully preprocessed images. Testing a single IJB-A/CS2 data split with one CNN model takes roughly one hour using the proposed facial representation pipeline (see Fig. 1). The most time consuming step in testing is PCA adaptation and it costs about 20 minutes for a single IJB-A/CS data split.

## 4. Experimental Evaluation

### 4.1. Metrics and Protocols

IJB-A dataset contains two types of protocols, namely *search* and *compare*. The *search* protocol measures the accuracy of open-set and closed-set search among  $N$  gallery templates in terms of the true acceptance rate (TAR) at various false acceptance rates (FAR) as well as receiver operating characteristic (ROC) plots. The *compare* protocol measures the verification accuracy between two templates. However, IJB-A carefully designed challenging template pairs by ensuring that subjects of templates have the same gender, and that their skin colors do not differ more than one level. Metrics of rank-1, rank-5, and the missing rate correspond to false alarm rates of  $\frac{1}{10}$  and  $\frac{1}{100}$ . Detailed descriptions of evaluation protocols and metrics can be found in [11].

### 4.2. Experiment Overview

In the rest of this section, we report the results of several experiments based on our proposed face representation and matching scheme. Since each experiment may have its own baseline, comparing performances across different experiments may be inaccurate. Although CS2 and IJBA both contains 10 splits of data, we only report averaged scores across these 10 splits to save space. Because CS2 and IJBA provide many associated attributes for a given face image, we do use provided subject face bounding boxes and seed landmarks in most of the following experiments, except those that explicitly state not to.

### 4.3. Selecting CNN Layers for Feature Extraction

As in many recognition problems, feature plays a core role in face recognition. Given a trained face recognition CNN, we may treat each layer as a feature, and also a collection of features from different layers. Therefore, it is interesting to investigate which combination of features is the best for face recognition. In this experiment, we exhaustively try all possible layer combinations of the last six layers of each CNN architecture, and report face recognition performance on selected combinations with reasonable performance in Table 3, where ‘x’ indicates that this layer of feature is used.

As one can see, the same collection of features means something different for ALEX-AF and VGG19-AF. Especially, we notice that when we only use the fc7 layer, the second-to-last dense layer, it is the best among all possible layer combinations for VGG19-AF, but not for ALEX-AF. This result clearly confirms that an optimal feature can be a set of features from different layers. From now on, all of our future experiments on ALEX-\* representations will be based on the feature combination of (pool5, fc7, fc8, prob),

Table 3: DNN Features for Face Recognition in CS2

Layer Name	CNN Feature of Layer Combinations									
prob	x	x	x	x						x
fc8	x				x	x	x	x	x	x
fc7		x			x		x		x	
fc6			x					x		
pool5				x		x	x	x		
Metric	ALEX-AF									
TAR@FAR=0.01	.572	.403	.374	.457	.572	.687	.688	.662	<b>.703</b>	.549
FAR@TAR=0.85	.039	.058	.060	.102	.042	.452	.044	.041	<b>.046</b>	.038
RANK@10	.875	.844	.820	.741	.867	.885	.878	.868	<b>.883</b>	.872
Metric	VGG19-AF									
TAR@FAR=0.01	.816	.724	.548	.749	.815	.788	.789	.753	.748	<b>.816</b>
FAR@TAR=0.85	.018	.018	.034	.030	.017	.022	.023	.028	.027	<b>.017</b>
RANK@10	.909	.892	.879	.913	.909	.921	.919	.916	.913	<b>.909</b>

while VGG\* representations are based on the feature combination of (fc8).

Furthermore, we compare four face recognition pipelines based on different features; specifically HDLBP, ALEX-AF, VGG16-AF, and VGG19-AF as shown in Table 4. It is clear that deep learning features outperform classic HDLBP features by a large margin, even though we spend an almost comparable amount of time for grid searching over the HDLBP hyper-parameter space.

Table 4: Feature Influences for Face Recognition in CS2

Metric	HDLBP	ALEX-AF	VGG16-AF	VGG19-AF
TAR@FAR=0.01	.274	.703	.779	.816
TAR@FAR=0.10	.511	.906	.918	.929
FAR@TAR=0.85	.680	.046	.025	.017
FAR@TAR=0.95	.930	.275	.228	.210
RANK@1	.398	.665	.739	.773
RANK@5	.596	.834	.862	.880
RANK@10	.689	.883	.895	.909

#### 4.3.1 Effect of Landmark Detection Algorithm

We mainly focus on evaluating facial landmark performance for face recognition. Our baseline of face recognition uses the ALEX-AF representation with cosine similarity and softmax score fusion, and the dataset used is CS2. Four state-of-the-art facial landmarks are used — (1) DLIB [9] with 68 points, (2) FPS3K [4] with 68 points, (3) TDCNN [23] with 5 points, and (4) CLNF with 68 points [2, 3] and its variant CLNFs that use seed landmarks provided in CS2 [11] for the estimation of an initial face shape. Note that we use DLIB, FPS3K and TDCNN out-of-shelf, and because they do not provide an interface to set the initial facial landmarks, we cannot report their performance with seed landmarks. Detailed results are listed in Table 5. All of these landmark detectors share the same set of face bounding boxes. In general, different landmark detectors do not make a huge performance difference compared to different features, and this implies that CNN features attain a certain level of spatial invariance. On the other hand, initial seed

landmarks do help to largely improve face recognition performance.

Table 5: Landmark Influence for Face Recognition in CS2

Metric	DLIB	FPS3K	TDCNN	CLNF	CLNF-s
TAR@FAR=0.01	.689	.708	.692	.682	.703
TAR@FAR=0.10	.894	.903	.906	.878	.906
FAR@TAR=0.85	.056	.049	.049	.062	.046
FAR@TAR=0.95	.249	.237	.228	.504	.275
RANK@1	.658	.636	.608	.661	.665
RANK@5	.821	.804	.803	.807	.834
RANK@10	.871	.861	.861	.862	.883

Table 6: Representation Influences for Face Recognition on CS2

Metric	AF	-FF	-PF	-FY0	-FY45	-FY75	quadruple	quintuple
ALEX- Arch.								
TAR@FAR=0.01	.703	.635	.332	.754	.797	.802	.814	.814
TAR@FAR=0.10	.906	.729	.443	.913	.935	.937	.939	.941
FAR@TAR=0.85	.046	.376	.558	.036	.020	.019	.017	.017
FAR@TAR=0.95	.275	.485	.661	.249	.151	.149	.143	.126
RANK@1	.665	.576	.263	.706	.751	.753	.781	.799
RANK@5	.834	.694	.367	.844	.883	.882	.898	.906
RANK@10	.883	.723	.419	.889	.918	.920	.928	.936
VGG19- Arch.								
TAR@FAR=0.01	.816	.688	.364	.806	.858	.860	.873	.897
TAR@FAR=0.10	.929	.738	.453	.930	.948	.948	.950	.959
FAR@TAR=0.85	.017	.489	.659	.020	.009	.009	.006	.003
FAR@TAR=0.95	.210	.616	.763	.171	.112	.111	.101	.065
RANK@1	.733	.655	.305	.769	.817	.802	.854	.865
RANK@5	.880	.723	.397	.881	.914	.913	.927	.934
RANK@10	.909	.743	.439	.912	.936	.936	.946	.949

#### 4.4. Single versus Multi-Pose Representations

We investigate the influence of using three different numbers of representations, namely *single*, *quadruple*, and *quintuple*. Specifically, *single* uses only one of \*-AF, \*-FF, \*-PF, \*-FY0, \*-FY45, \*-FY75 representations, *quadruple* uses the representation tuple of (\*-FF, \*-PF, \*-FY0, \*-FY45) and *quintuple* uses the representation tuple of (\*-FF, \*-PF, \*-FY0, \*-FY45, \*-FY75), where \* denotes either ALEX or VGG19 CNN architecture. As shown in Table 8, face recognition performance significantly improves as the number of pose representations increases, regardless of whether we use ALEX architecture or VGG19 architecture.

#### 4.5. Comparing to State-of-the-Art Methods

We now compare our VGG19-quintuple representation, namely (VGG19-FF, -PF, -FY0, -FY45, -FY75), to state-of-the-art methods [19, 15, 5], along with the baseline GOTS and COTS from [11] on the IJB-A dataset. It is clear that our multi-pose representation-based recognition pipeline outperforms other state-of-the-art methods. It is worthy to mention that the algorithm presented in [5] involves both fine tuning on IJB-A data and uses metric learning using labeled IJB-A training data, while our algorithm is data-agnostic and is used out of the box on both CS2 and IJB-A without target domain specific tuning.

Table 7: Results on CS2

Metric	COTS	GOTS	FV[5]	DCNN-all[5]	[19]	Ours
TAR@FAR=0.01	.581	.467	.411	.876	.733	.897
TAR@FAR=0.10	.761	.675	.704	.973	.895	.959
RANK@1	.551	.413	.381	.838	.820	.865
RANK@5	.694	.517	.559	.924	.929	.934
RANK@10	.741	.624	.637	.949	-	.949

Table 8: 1:N Results on IJB-A

Metric	COTS	GOTS	[15]	DCNN-all[5]	[19]	Ours
	1:N (Search Protocol)					
TAR@FAR=0.01	.406	.236	-	-	.733	.876
TAR@FAR=0.10	.627	.433	-	-	.895	.954
RANK@1	.443	.246	.588	.860	.820	.846
RANK@5	.595	.375	.797	.943	.929	.927
RANK@10	-	-	-	-	-	.947
	1:1 (Verification Protocol)					
TAR@FAR=0.01	-	-	-	-	-	.787
TAR@FAR=0.10	-	-	-	-	-	.911

## 5. Conclusion

We introduced a multi-pose representation for face recognition, which is a collection of face representations learned from specific face poses. We show that this novel representation significantly improves face recognition performance on IJB-A benchmark compared not only to the single best CNN representations but also those state-of-the-art methods that heavily rely on supervised learning, such gallery fine-tuning and metric learning.

## Acknowledgment

This research is mainly based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA’s 2014-14071600011. 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 ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon. We thank the NVIDIA Corporation for the donation of the Tesla K40 GPU.

## References

- [1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041, 2006. 3, 4
- [2] T. Baltrušaitis, P. Robinson, and L.-P. Morency. 3d constrained local model for rigid and non-rigid facial tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2610–2617. IEEE, 2012. 7
- [3] T. Baltrušaitis, P. Robinson, and L.-P. Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 354–361. IEEE, 2013. 7
- [4] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. *International Journal of Computer Vision*, 107(2):177–190, 2014. 7
- [5] J.-C. Chen, V. M. Patel, and R. Chellappa. Unconstrained face verification using deep cnn features. *arXiv preprint arXiv:1508.01722*, 2015. 1, 2, 3, 7, 8
- [6] C. Ding and D. Tao. Robust face recognition via multimodal deep face representation. *Multimedia, IEEE Transactions on*, 17(11):2049–2058, Nov 2015. 1, 3
- [7] T. Hassner, S. Harel, E. Paz, and R. Enbar. Effective face frontalization in unconstrained images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4295–4304, June 2015. 3
- [8] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007. 1
- [9] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1867–1874. IEEE, 2014. 7
- [10] I. Kemelmacher-Shlizerman and R. Basri. 3d face reconstruction from a single image using a single reference face shape. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2):394–405, Feb 2011. 3
- [11] B. F. Klare, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, M. Burge, and A. K. Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. *algorithms*, 13:4, 2015. 1, 2, 6, 7
- [12] 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. 4
- [13] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1717–1724. IEEE, 2014. 4
- [14] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision—ECCV 2010*, pages 143–156. Springer, 2010. 3
- [15] A. RoyChowdhury, T.-Y. Lin, S. Maji, and E. Learned-Miller. Face identification with bilinear cnns. *arXiv preprint arXiv:1506.01342*, 2015. 1, 2, 3, 7, 8
- [16] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 397–403. IEEE, 2013. 2
- [17] K. Sikka, T. Wu, J. Susskind, and M. Bartlett. Exploring bag of words architectures in the facial expression domain. In



*Computer Vision—ECCV 2012. Workshops and Demonstrations*, pages 250–259. Springer, 2012. 3

- [18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4
- [19] D. Wang, C. Otto, and A. K. Jain. Face search at scale: 80 million gallery. *arXiv preprint arXiv:1507.07242*, 2015. 1, 2, 3, 7, 8
- [20] L. Wolf, T. Hassner, and Y. Taigman. Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10):1978–1990, Oct 2011. 3
- [21] S. Xie, S. Shan, X. Chen, and J. Chen. Fusing local patterns of gabor magnitude and phase for face recognition. *Image Processing, IEEE Transactions on*, 19(5):1349–1361, 2010. 3
- [22] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 2, 4
- [23] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *Computer Vision—ECCV 2014*, pages 94–108. Springer, 2014. 2, 7