

# $L_2$ -constrained Softmax Loss for Discriminative Face Verification

Rajeev Ranjan, Carlos D. Castillo and Rama Chellappa

Center for Automation Research, UMIACS, University of Maryland, College Park, MD 20742  
 {rranjan1, carlos, rama}@umiacs.umd.edu

## Abstract

*In recent years, the performance of face verification systems has significantly improved using deep convolutional neural networks (DCNNs). A typical pipeline for face verification includes training a deep network for subject classification with softmax loss, using the penultimate layer output as the feature descriptor, and generating a cosine similarity score given a pair of face images. The softmax loss function does not optimize the features to have higher similarity score for positive pairs and lower similarity score for negative pairs, which leads to a performance gap. In this paper, we add an  $L_2$ -constraint to the feature descriptors which restricts them to lie on a hypersphere of a fixed radius. This module can be easily implemented using existing deep learning frameworks. We show that integrating this simple step in the training pipeline significantly boosts the performance of face verification. Specifically, we achieve state-of-the-art results on the challenging IJB-A dataset, achieving True Accept Rate of 0.909 at False Accept Rate 0.0001 on the face verification protocol. Additionally, we achieve state-of-the-art performance on LFW dataset with an accuracy of 99.78%, and competing performance on YTF dataset with accuracy of 96.08%.*

## 1. Introduction

Face verification in unconstrained settings is a challenging problem. Despite the excellent performance of recent face verification systems on curated datasets like Labeled Faces in the Wild (LFW) [14], it is still difficult to achieve similar accuracy on faces with extreme variations in viewpoints, resolution, occlusion and image quality. This is evident from the performance of the traditional algorithms on the publicly available IJB-A [16] dataset. Data quality imbalance in the training set is one of the reason for this performance gap. Existing face recognition training datasets contain large amount of high quality and frontal faces, whereas the unconstrained and difficult faces occur rarely. Most of the DCNN-based methods trained with softmax loss for classification tend to over-fit to the high quality data and fail

to correctly classify faces acquired in difficult conditions.

Using softmax loss function for training face verification system has its own pros and cons. On the one hand, it can be easily implemented using inbuilt functions from the publicly available deep learning toolboxes such as Caffe [15], Torch [7] and TensorFlow [1]. Unlike triplet loss [28], it does not have any restrictions on the input batch size and converges quickly. The learned features are discriminative enough for efficient face verification without any metric learning.

On the other hand, the softmax loss is biased to the sample distribution. Unlike contrastive loss [29] and triplet loss [28] which specifically attend to hard samples, the softmax loss maximizes the conditional probability of all the samples in a given mini-batch. Hence, it fits well to the high quality faces, ignoring the rare difficult faces from a training mini-batch. We observe that the  $L_2$ -norm of features learned using softmax loss is informative of the quality of the face [23]. Features for good quality frontal faces have a high  $L_2$ -norm while blurry faces with extreme pose have low  $L_2$ -norm (see Figure 1(b)). Moreover, the softmax loss does not optimize the verification requirement of keeping positive pairs closer and negative pairs far from each other. Due to this reason, many methods either apply metric learning on top of softmax features [27, 3, 24] or train an auxiliary loss [33, 29, 32] along with the softmax loss to achieve better verification performance.

In this paper, we provide a symptomatic treatment to issues associated with the softmax loss. We propose an  $L_2$ -softmax loss that adds a constraint on the features during training such that their  $L_2$ -norm remain constant. In other words, we restrict the features to lie on a hypersphere of a fixed radius. The proposed  $L_2$ -softmax loss has a dual advantage. Firstly, it provides similar attention to both good and bad quality faces since all the features have the same  $L_2$ -norm now, which is essential for better performance in unconstrained settings. Secondly, it strengthens the verification signal by forcing the same subject features to be closer and different subject features to be far from each other in the normalized space. Thus, it maximizes the margin for the normalized  $L_2$  distance or cosine similarity score between negative and positive pairs. Thus, it overcomes the



main disadvantages of the regular softmax loss.

The  $L_2$ -softmax loss also retains the advantages of the regular softmax loss. Similar to the softmax loss, it is a one network, one loss system. It doesn't necessarily require any joint supervision as used by many recent methods [33, 24, 32, 29]. It can be easily implemented using inbuilt functions from Caffe [15], Torch [7] and TensorFlow [1], and converges very fast. It introduces just a single scaling parameter to the network. Compared to the regular softmax loss, the  $L_2$ -softmax loss gains a significant boost in the performance. It achieves new state-of-the-art results on IJB-A dataset, and competing results on LFW and YouTube Face datasets. It surpasses the performance of several state-of-the-art systems, which use multiple networks or multiple loss functions or both. In summary, this paper contributes to the following aspects:

1. We propose a simple, novel and effective  $L_2$ -softmax loss for face verification that restricts the  $L_2$ -norm of the feature descriptor to a constant value  $\alpha$ .
2. We study the variations in the performance with respect to the scaling parameter  $\alpha$  and provide suitable bounds on its value for achieving consistently high performance.
3. The proposed method yields a consistent and significant boost on all the three challenging face verification datasets namely LFW [14], YouTube Face [19] and IJB-A [16]

Moreover, the gains from  $L_2$ -softmax loss are complementary to metric learning (eg: TPE [27], joint-Bayes [3]) or auxiliary loss functions (eg: center loss [33], contrastive loss [29]). We show that applying these techniques on top of the  $L_2$ -softmax loss can further improve the verification performance. Combining with TPE [27],  $L_2$ -softmax loss achieves a record True Accept Rate (TAR) of 0.909 at False Accept Rate (FAR) of 0.0001 on the challenging IJB-A [16] dataset.

## 2. Related Work

In recent years, there has been a significant improvement in the accuracy of face verification using deep learning methods [28, 30, 24, 27, 29, 33]. Most of these methods have even surpassed human performance on the LFW [14] dataset. Although these methods use DCNNs, they differ by the type of loss function they use for training. For face verification, it's essential for the positive subjects pair features to be closer and negative subjects pair features far apart. To solve this problem, researchers have adopted two major approaches.

In the first approach, pairs of face images are input to the training algorithm to learn a feature embedding where

positive pairs are closer and negative pairs are far apart. In this direction, Chopra et al. [5] proposed siamese networks with contrastive loss for training. Hu et al. [13] designed a discriminative deep metric with a margin between positive and negative face pairs. FaceNet [28] introduces triplet loss to learn the metric using hard triplet face samples.

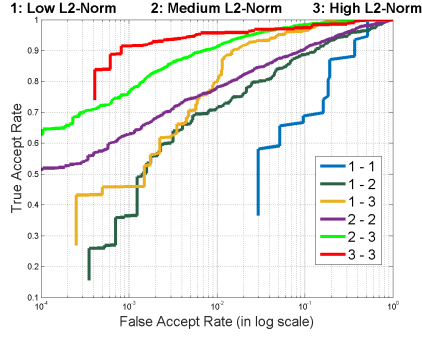
In the second approach, the face images along with their subject labels are used to learn discriminative identification features in a classification framework. Most of the recent methods [29, 30, 24, 37] train a DCNN with softmax loss to learn these features which are later used either to directly compute the similarity score for a pair of faces or to train a discriminative metric embedding [27, 3]. Another strategy is to train the network for joint identification-verification task [29, 32, 33]. Xiong et al. [36] proposed transferred deep feature fusion (TDFF) which involves two-stage fusion of features trained with different networks and datasets. Template adaptation [8] is applied to further boost the performance.

A recent approach [33] introduced center loss to learn better discriminative face features. Our proposed method is different from the center loss in the following aspects. First, we use one loss function (i.e.,  $L_2$ -softmax loss) whereas [33] uses center loss jointly with the softmax loss during training. Second, center loss introduces  $C \times D$  additional parameters during training where  $C$  is the number of classes and  $D$  is the feature dimension. On the other hand, the  $L_2$ -softmax loss introduces just a single parameter that defines the fixed  $L_2$ -norm of the features. Moreover, the center loss can also be used in conjunction with  $L_2$ -softmax loss, which performs better than center loss trained with regular softmax loss (see Section 5.1.4).

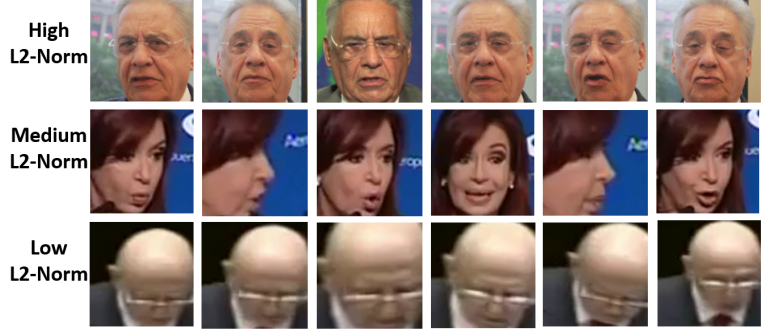
Recently, a few algorithms have used feature normalization during training to improve performance. SphereFace [20] proposes angular softmax (A-softmax) loss that enables DCNNs to learn angularly discriminative features. Another method called DeepVisage [10] uses a special case of batch normalization technique to normalize the feature descriptor before applying the softmax loss. Our proposed method is different as it applies an  $L_2$ -constraint on the feature descriptors enforcing them to lie on a hypersphere of a given radius.

## 3. Motivation

We first summarize the general pipeline for training a face verification system using DCNN as shown in Figure 2. Given a training dataset with face images and corresponding identity labels, a DCNN is trained as a classification task where the network learns to classify a given face image to its correct identity label. A softmax loss function is used for training the network, given by Equation 1



(a)



(b)

Figure 1. (a) Face Verification Performance on IJB-A dataset. The templates are divided into 3 sets based on their  $L_2$ -norm. ‘1’ denotes the set with low  $L_2$ -norm while ‘3’ represents high  $L_2$ -norm. The legend ‘x-y’ denote the evaluation pairs where one template is from set ‘x’ and another from set ‘y’. (b) Sample template images from IJB-A dataset with high, medium and low  $L_2$ -norm

$$L_S = -\frac{1}{M} \sum_{i=1}^M \log \frac{e^{W_{y_i}^T f(\mathbf{x}_i) + b_{y_i}}}{\sum_{j=1}^C e^{W_j^T f(\mathbf{x}_i) + b_j}}, \quad (1)$$

where  $M$  is the training batch size,  $\mathbf{x}_i$  is the  $i^{th}$  input face image in the batch,  $f(\mathbf{x}_i)$  is the corresponding output of the penultimate layer of the DCNN,  $y_i$  is the corresponding class label, and  $W$  and  $b$  are the weights and bias for the last layer of the network which acts as a classifier.

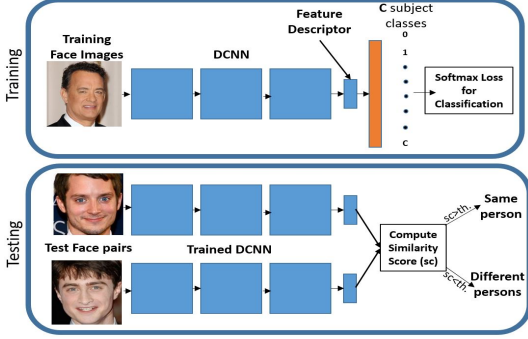


Figure 2. A general pipeline for training and testing a face verification system using DCNN.

At test time, feature descriptors  $f(\mathbf{x}_g)$  and  $f(\mathbf{x}_p)$  are extracted for the pair of test face images  $\mathbf{x}_g$  and  $\mathbf{x}_p$  respectively using the trained DCNN, and normalized to unit length. Then, a similarity score is computed on the feature vectors which provides a measure of distance or how close the features lie in the embedded space. If the similarity score is greater than a set threshold, the face pairs are decided to be of the same person. Usually, the similarity score is computed as the  $L_2$ -distance between the normalized features [28, 24] or by using cosine similarity [33, 3, 25, 27]  $s$ , as given by Equation 2. Both these similarity measures are equivalent and produce same results.

$$s = \frac{f(\mathbf{x}_g)^T f(\mathbf{x}_p)}{\|f(\mathbf{x}_g)\|_2 \|f(\mathbf{x}_p)\|_2} \quad (2)$$

There are two major issues with this pipeline. First, the training and testing steps for face verification task are decoupled. Training with softmax loss doesn’t necessarily ensure the positive pairs to be closer and the negative pairs to be far separated in the normalized or angular space.

Secondly, the softmax classifier is weak in modeling difficult or extreme samples. In a typical training batch with data quality imbalance, the softmax loss gets minimized by increasing the  $L_2$ -norm of the features for easy samples, and ignoring the hard samples. The network thus learns to respond to the quality of the face by the  $L_2$ -norm of its feature descriptor. To validate this theory, we perform a simple experiment on the IJB-A [16] dataset where we divide the templates (groups of images/frames of same subject) into three different sets based on the  $L_2$ -norm of their feature descriptors. The features were computed using Face-Resnet [33] trained with regular softmax loss. Templates with descriptors’  $L_2$ -norm  $< 90$  are assigned to set1. The templates with  $L_2$ -norm  $> 90$  but  $< 150$  are assigned to set2, while templates with  $L_2$ -norm  $> 150$  are assigned to set3. In total they form six sets of evaluation pairs. Figure 1(a) shows the performance of the these six different sets for the IJB-A face verification protocol. It can be clearly seen that pairs having low  $L_2$ -norm for both the templates perform very poor, while the pairs with high  $L_2$ -norm perform the best. The difference in performance between each set is quite significant. Figure 1(b) shows some sample templates from set1, set2 and set3 which confirms that the  $L_2$ -norm of the feature descriptor is informative of its quality.

To solve these issues, we enforce the  $L_2$ -norm of the features to be fixed for every face image. Specifically, we add an  $L_2$ -constraint to the feature descriptor such that it lies on a hypersphere of a fixed radius. This approach has two advantages. Firstly, on a hypersphere, minimizing the

softmax loss is equivalent to maximizing the cosine similarity for the positive pairs and minimizing it for the negative pairs, which strengthens the verification signal of the features. Secondly, the softmax loss is able to model the extreme and difficult faces better, since all the face features have same  $L_2$ -norm.

## 4. Proposed Method

The proposed  $L_2$ -softmax loss is given by Equation 3

$$\begin{aligned} & \text{minimize} \quad -\frac{1}{M} \sum_{i=1}^M \log \frac{e^{W_{y_i}^T f(\mathbf{x}_i) + b_{y_i}}}{\sum_{j=1}^C e^{W_j^T f(\mathbf{x}_i) + b_j}} \\ & \text{subject to} \quad \|f(\mathbf{x}_i)\|_2 = \alpha, \quad \forall i = 1, 2, \dots, M, \end{aligned} \quad (3)$$

where  $\mathbf{x}_i$  is the input image in a mini-batch of size  $M$ ,  $y_i$  is the corresponding class label,  $f(\mathbf{x}_i)$  is the feature descriptor obtained from the penultimate layer of DCNN,  $C$  is the number of subject classes, and  $W$  and  $b$  are the weights and bias for the last layer of the network which acts as a classifier. This equation adds an additional  $L_2$ -constraint to the regular softmax loss defined in Equation 1. We show the effectiveness of this constraint using MNIST [17] data.

### 4.1. MNIST Example

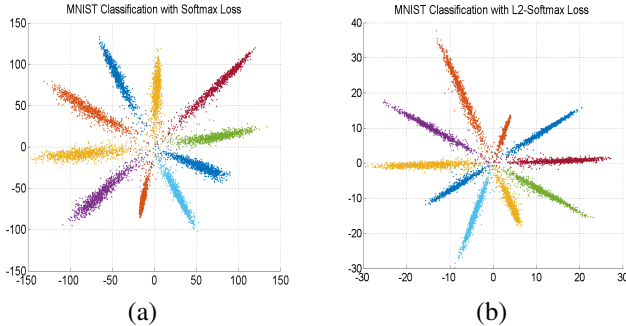


Figure 3. Visualization of 2-dimensional features for MNIST digit classification test set using (a) Softmax Loss. (b)  $L_2$ -Softmax Loss

We study the effect of  $L_2$ -softmax loss on the MNIST dataset [17]. We use a deeper and wider version of LeNet mentioned in [33], where the last hidden layer output is restricted to 2-dimensions for easy visualization. For the first setup, we train the network end-to-end using the regular softmax loss for digits classification with number of classes = 10. For the second setup, we add an  $L_2$ -normalize layer and scale layer to the 2-dimensional features which enforces the  $L_2$ -constraint described in Equation 3 (seen Section 4.2 for details). Figure 3 depicts the 2-D features for different classes for MNIST test set containing 10,000 digit images. Each of the lobes shown in the figure represents 2-D features of unique digits classes. The features for the second setup were obtained before the  $L_2$ -normalization layer.

Table 1. Accuracy on MNIST test set in (%)

	Softmax Loss	$L_2$ -Softmax Loss
Accuracy	98.88	99.05

We find two clear differences between the features learned using the two setups discussed above. First, the intra-class angular variance is large when using the regular softmax loss, which can be estimated by the average width of the lobes for each class. On the other hand, the features obtained with  $L_2$ -softmax loss have lower intra-class angular variability, and are represented by thinner lobes. Second, the magnitudes of the features are much higher with the softmax loss (ranging upto 150), since larger feature norms result in a higher probability for a correctly classified class. In contrast, the feature norm has minimal effect on the  $L_2$ -softmax loss since every feature is normalized to a circle of fixed radius before computing the loss. Hence, the network focuses on bringing the features from the same class closer to each other and separating the features from different classes in the normalized or angular space. Table 1 lists the accuracy obtained with the two setups on MNIST test set.  $L_2$ -softmax loss achieves a higher performance, reducing the error by more than 15%. Note that these accuracy numbers are lower compared to a typical DCNN since we are using only 2-dimensional features for classification.

### 4.2. Implementation Details

Here, we provide the details of implementing the  $L_2$ -constraint described in Equation 3 in the framework of DCNNs. The constraint is enforced by adding an  $L_2$ -normalize layer followed by a scale layer as shown in Figure 4.

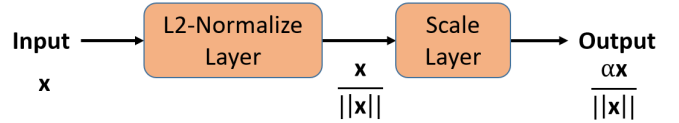


Figure 4. We add an  $L_2$ -normalize layer and a scale layer to constrain the feature descriptor to lie on a hypersphere of radius  $\alpha$ .

This module is added just after the penultimate layer of DCNN which acts as a feature descriptor. The  $L_2$ -normalize layer normalizes the input feature  $\mathbf{x}$  to a unit vector given by Equation 4. The scale layer scales the input unit vector to a fixed radius given by the parameter  $\alpha$  (Equation 5). In total, we just introduce one scalar parameter ( $\alpha$ ) which can be trained along with the other parameters of the network.

$$\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2} \quad (4)$$

$$\mathbf{z} = \alpha \cdot \mathbf{y} \quad (5)$$



The module is fully differentiable and can be used in the end-to-end training of the network. At test time, the proposed module is redundant, since the features are eventually normalized to unit length while computing the cosine similarity. At training time, we backpropagate the gradients through the  $L_2$ -normalize and the scale layer, as well as compute the gradients with respect to the scaling parameter  $\alpha$  using the chain rule as given below.

$$\begin{aligned}
\frac{\partial l}{\partial y_i} &= \frac{\partial l}{\partial z_i} \cdot \alpha \\
\frac{\partial l}{\partial \alpha} &= \sum_{j=1}^D \frac{\partial l}{\partial z_j} \cdot y_j \\
\frac{\partial l}{\partial x_i} &= \sum_{j=1}^D \frac{\partial l}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i} \\
\frac{\partial y_i}{\partial x_i} &= \frac{\|\mathbf{x}\|_2^2 - x_i^2}{\|\mathbf{x}\|_2^3} \\
\frac{\partial y_j}{\partial x_i} &= \frac{-x_i \cdot x_j}{\|\mathbf{x}\|_2^3}
\end{aligned} \tag{6}$$

#### 4.3. Bounds on Parameter $\alpha$

The scaling parameter  $\alpha$  plays a crucial role in deciding the performance of  $L_2$ -softmax loss. There are two ways to enforce the  $L_2$ -constraint: 1) by keeping  $\alpha$  fixed throughout the training, and 2) by letting the network to learn the parameter  $\alpha$ . The second way is elegant and always improves over the regular softmax loss. But, the  $\alpha$  parameter learned by the network is high which results in a relaxed  $L_2$ -constraint. The softmax classifier aimed at increasing the feature norm for minimizing the overall loss, increases the  $\alpha$  parameter instead, allowing it more freedom to fit to the easy samples. Hence,  $\alpha$  learned by the network forms an upper bound for the parameter. A better performance is obtained by fixing  $\alpha$  to a lower constant value.

On the other hand, with a very low value of  $\alpha$ , the training doesn't converge. For instance,  $\alpha = 1$  performs very poorly on the LFW [14] dataset, achieving an accuracy of 86.37% (see Figure 7). The reason being that a hypersphere with small radius ( $\alpha$ ) has limited surface area for embedding features from the same class together and those from different classes far from each other.

Here, we formulate a theoretical lower bound on  $\alpha$ . Assuming the number of classes  $C$  to be lower than twice the feature dimension  $D$ , we can distribute the classes on a hypersphere of dimension  $D$  such that any two class centers are at least  $90^\circ$  apart. Figure 5(a) represents this case for  $C = 4$  class centers distributed on a circle of radius  $\alpha$ . We assume the classifier weights ( $W_i$ ) to be a unit vector pointing in the direction of their respective class centers. We ignore the bias term. The average softmax probability  $p$  for correctly classifying a feature is given by Equation 7

$$\begin{aligned}
p &= \frac{e^{W_i^T X_i}}{\sum_{j=1}^4 e^{W_j^T X_i}} \\
&= \frac{e^\alpha}{e^\alpha + 2 + e^{-\alpha}}
\end{aligned} \tag{7}$$

Ignoring the term  $e^{-\alpha}$  and generalizing it for  $C$  classes, the average probability becomes:

$$p = \frac{e^\alpha}{e^\alpha + C - 2} \tag{8}$$

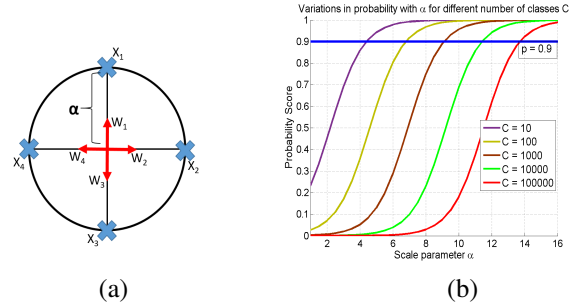


Figure 5. (a) 2-D visualization of the assumed distribution of features (b) Variation in Softmax probability with respect to  $\alpha$  for different number of classes  $C$

Figure 5(b) plots the probability score as a function of the parameter  $\alpha$  for various number of classes  $C$ . We can infer that to achieve a given classification probability (say  $p = 0.9$ ), we need to have a higher  $\alpha$  for larger  $C$ . Given the number of classes  $C$  for a dataset, we can obtain the lower bound on  $\alpha$  to achieve a probability score of  $p$  by using Equation 9.

$$\alpha_{low} = \log \frac{p(C-2)}{1-p} \tag{9}$$

## 5. Results

We use the publicly available Face-Resnet [33] DCNN for our experiments. Figure 6 shows the architecture of the network. It contains 27 convolutional layers and 2 fully-connected layers. The dimension of the feature descriptor is 512. It utilizes the widely used residual skip-connections [12]. We add an  $L_2$ -normalize layer and a scale layer after the fully-connected layer to enforce the  $L_2$ -constraint on the descriptor. All our experiments are carried out in Caffe [15].

### 5.1. Baseline experiments

In this subsection, we experimentally validate the usefulness of the  $L_2$ -softmax loss for face verification. We form two subsets of training dataset from the MS-Celeb-1M [9] dataset: 1) MS-small containing 0.5 million face images with the number of subjects being 13403, and 2) MS-large containing 3.7 million images of 58207 subjects. The

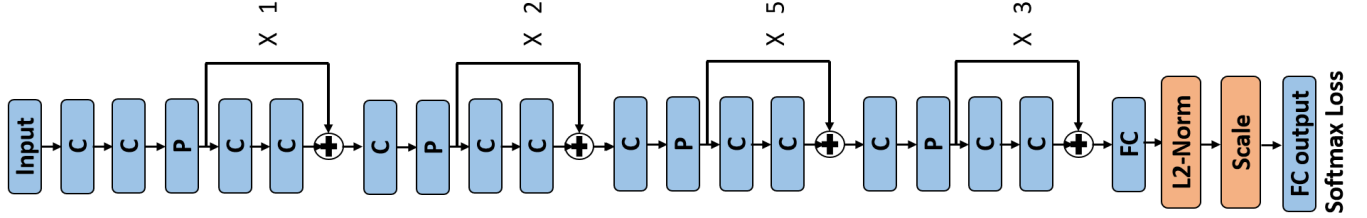


Figure 6. The Face-Resnet architecture [33] used for the experiments. **C** denotes Convolution Layer followed by PReLU [11] while **P** denotes Max Pooling Layer. Each pooling layer is followed by a set of residual connections, the count for which is denoted alongside. After the fully-connected layer (FC), we add an  $L_2$ -Normalize layer and Scale Layer which is followed by the softmax loss.

dataset was cleaned using the clustering algorithm mentioned in [18]. We train the Face-Resnet network with softmax loss as well as  $L_2$ -softmax loss for various  $\alpha$ . While training with MS-small, we start with a base learning rate of 0.1 and decrease it by  $1/10^{th}$  after  $16K$  and  $24K$  iterations, upto a maximum of  $28K$  iterations. For training on MS-large, we use the same learning rate but decrease it after  $50K$  and  $80K$  iterations upto a maximum of  $100K$  iterations. A training batch size of 256 was used. Both softmax and  $L_2$ -softmax loss functions consume the same amount of training time which is around 9 hours for MS-small and 32 hours for MS-large training set respectively, on two TITAN X GPUs. We set the learning multiplier and decay multiplier for the scale layer to 1 for trainable  $\alpha$ , and 0 for fixed  $\alpha$  during the network training. We evaluate our baselines on the widely used LFW dataset [14] for the unrestricted setting, and the challenging IJB-A dataset [16] on the 1:1 face verification protocol. The faces were cropped and aligned to the size of  $128 \times 128$  in both training and testing phases by implementing the face detection and alignment algorithm mentioned in [25].

### 5.1.1 Experiment with small training set

Here, we compare the network trained on MS-small dataset using our proposed  $L_2$ -softmax loss, against the one trained with regular softmax loss. Figure 7 shows that the regular softmax loss attains an accuracy of 98.1% whereas the proposed  $L_2$ -softmax loss achieves the best accuracy of 99.28%, thereby reducing the error by more than 62%. It also shows the variations in performance with the scale parameter  $\alpha$ . The performance is poor when  $\alpha$  is below a certain threshold and stable with  $\alpha$  higher than the threshold. This behavior is consistent with the theoretical analysis presented in Section 4.3. From the figure, the performance of  $L_2$ -Softmax is better for  $\alpha > 12$  which is close to its lower bound computed using equation 9 for  $C = 13403$  with a probability score of 0.9.

A similar trend is observed for 1:1 verification protocol on IJB-A [16] as shown in Table 2, where the numbers denote True Accept Rate (TAR) at False Accept Rates (FAR) of 0.0001, 0.001, 0.01 and 0.1. Our proposed approach

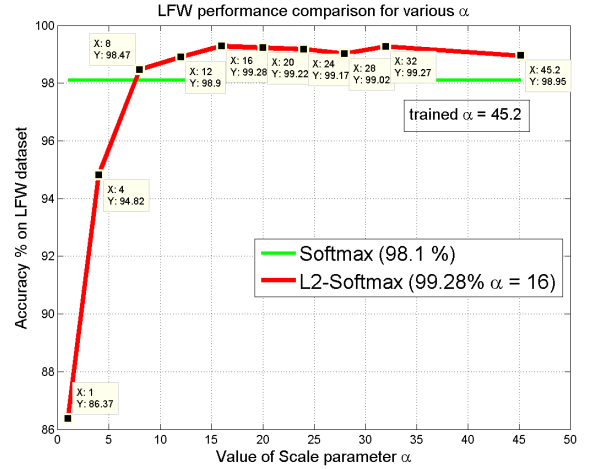


Figure 7. The red curve shows the variations in LFW accuracy with the parameter  $\alpha$  for  $L_2$ -softmax loss. The green line is the accuracy using the usual softmax loss.

improves the TAR@FAR=0.0001 by 19% compared to the baseline softmax loss. The performance is consistent with  $\alpha$  ranging between 16 to 32. Another point to note is that by allowing the network to learn the scale parameter  $\alpha$  by itself results in a slight decrease in performance, which shows that having a tighter constraint is a better choice.

Table 2. TAR on IJB-A 1:1 Verification Protocol @FAR

	0.0001	0.001	0.01	0.1
softmax	0.553	0.730	0.881	0.957
$L_2$ -softmax ( $\alpha=8$ )	0.257	0.433	0.746	0.953
$L_2$ -softmax ( $\alpha=12$ )	0.620	0.721	0.875	0.970
$L_2$ -softmax ( $\alpha=16$ )	0.734	0.834	<b>0.924</b>	0.974
$L_2$ -softmax ( $\alpha=20$ )	0.740	0.820	0.922	0.973
$L_2$ -softmax ( $\alpha=24$ )	<b>0.744</b>	0.831	0.912	0.974
$L_2$ -softmax ( $\alpha=28$ )	0.740	<b>0.834</b>	0.922	<b>0.975</b>
$L_2$ -softmax ( $\alpha=32$ )	0.727	0.831	0.921	0.972
$L_2$ -softmax ( $\alpha$ trained)	0.698	0.817	0.914	0.971

### 5.1.2 Experiment with large training set

We train the network on the MS-large dataset for this experiment. Figure 8 shows the performance on the LFW dataset. Similar to the small training set, the  $L_2$ -softmax loss significantly improves over the baseline, reducing the error by 60% and achieving an accuracy of 99.6%. Similarly, it improves the TAR@FAR=0.0001 on IJB-A by more than 10% (Table 3). The performance of  $L_2$ -softmax is consistent with  $\alpha$  in the range 40 and beyond. Unlike, the small set training, the self-trained  $\alpha$  performs equally good compared to fixed  $\alpha$  of 40 and 50. The theoretical lower bound on  $\alpha$  is not of much use in this case since improved performance is achieved for  $\alpha > 30$ . We can deduce that as the number of subjects increases, the lower bound on  $\alpha$  is less reliable, and the self-trained  $\alpha$  is more reliable with performance. This experiment clearly suggests that the proposed  $L_2$ -softmax loss is consistent across the training and testing datasets.

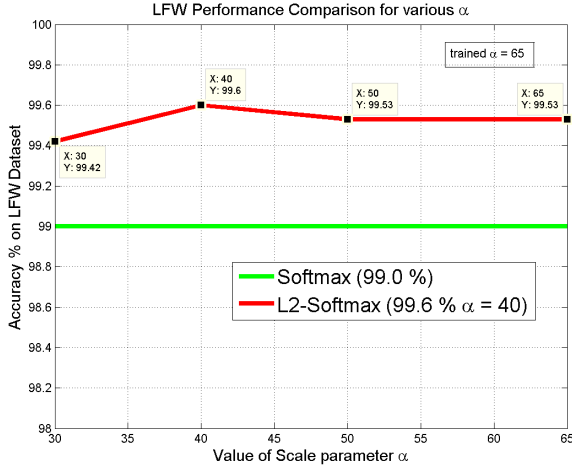


Figure 8. The red curve shows the variations in LFW accuracy with the parameter  $\alpha$  for  $L_2$ -softmax loss. The green line is the accuracy using the softmax loss.

Table 3. TAR on IJB-A 1:1 Verification Protocol @FAR

	0.0001	0.001	0.01	0.1
softmax	0.730	0.851	0.926	0.972
$L_2$ -softmax ( $\alpha=30$ )	0.775	0.871	0.938	0.978
$L_2$ -softmax ( $\alpha=40$ )	0.827	0.900	0.951	<b>0.982</b>
$L_2$ -softmax ( $\alpha=50$ )	<b>0.832</b>	<b>0.906</b>	<b>0.952</b>	0.981
$L_2$ -softmax ( $\alpha$ trained)	<b>0.832</b>	0.903	0.950	0.980

### 5.1.3 Experiment with a different DCNN

To check the consistency of our proposed  $L_2$ -softmax loss, we apply it on the All-In-One Face [25] instead of the Face-Resnet. We use the recognition branch of the All-In-One

Face to fine-tune on the MS-small training set. The recognition branch of All-In-One Face consists of 7 convolution layers followed by 3 fully-connected layers and a softmax loss. We add an  $L_2$ -normalize and a scale layer after the 512 dimension feature descriptor. Figure 9 shows the comparison of  $L_2$ -softmax loss and the base softmax loss on LFW dataset. Similar to the Face-Resnet, All-In-One Face with  $L_2$ -softmax loss improves over the base softmax performance, reducing the error by 40%, and achieving an accuracy of 98.82%. The improvement obtained by using All-In-One Face is smaller compared to the Face-Resnet. This shows that residual connections and depth of the network generate better feature embedding on a hypersphere. The performance variation with scaling parameter  $\alpha$  is similar to that of Face-Resnet, indicating that the optimal scale parameter does not depend on the choice of the network.

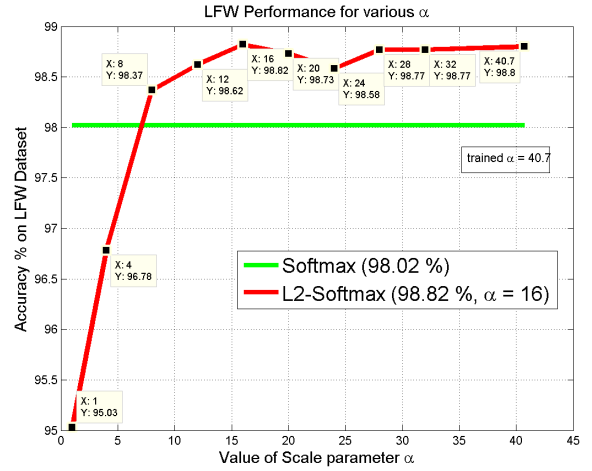


Figure 9. The red curve shows the variations in LFW accuracy with the parameter  $\alpha$  for  $L_2$ -softmax loss. The green line is the accuracy using the softmax loss.

### 5.1.4 Experiment with auxiliary loss

Similar to softmax loss, the  $L_2$ -softmax loss can be coupled with auxiliary losses such as center loss, contrastive loss, triplet loss, etc. to further improve the performance. Here we study the performance variation of  $L_2$ -softmax loss when coupled with the center loss. We use the MS-small dataset for training the networks. Table 4 lists the accuracy obtained on LFW dataset by different loss functions. The softmax loss performs the worst. The center loss improves the performance significantly when trained in conjunction with the softmax loss, and is comparable to the  $L_2$ -softmax loss. Training center loss with the  $L_2$ -softmax loss gives the best performance of 99.33% accuracy. This shows that  $L_2$ -softmax loss is as versatile as the regular softmax loss and can be used efficiently with other auxiliary loss functions.

Table 4. Accuracy on LFW (%)

softmax loss	98.10
center loss [33] + softmax loss	99.23
$L_2$ -softmax loss	99.28
center loss [33] + $L_2$ -softmax loss	<b>99.33</b>

## 5.2. Comparison with recent methods

We compare our algorithm with recently reported face verification methods on LFW [14], YouTube Face [34] and IJB-A [16] datasets. We crop and align the images for all these datasets by implementing the algorithm mentioned in [25]. We train the Face-Resnet (FR) with  $L_2$ -softmax as well as regular softmax loss using the MS-large training set. Additionally, we train ResNet-101(R101) [12] and ResNeXt-101(RX101) [35] deep networks for face recognition using MS-large training set with  $L_2$ -softmax loss. Both R101 and RX101 models were initialized with parameters pre-trained on ImageNet [26] dataset. A fully-connected layer of dimension 512 was added before the  $L_2$ -softmax classifier. The scaling parameter was kept fixed with a value of  $\alpha = 50$ . Experimental results on different datasets show that  $L_2$ -softmax works efficiently with deeper models.

The LFW dataset [14] contains 13,233 web-collected images from 5749 different identities. We evaluate our model following the standard protocol of unrestricted with labeled outside data. We test on 6,000 face pairs and report the experiment results in Table 5. Along with the accuracy values, we also compare with the number of images, networks and loss functions used by the methods for their overall training. The proposed method attains the state-of-the-art performance with the RX101 model, achieving an accuracy of 99.78%. Unlike other methods which use auxiliary loss functions such as center loss and contrastive loss along with the primary softmax loss, our method uses a single loss training paradigm which makes it easier and faster to train.

YouTube Face (YTF) [34] dataset contains 3425 videos of 1595 different people, with an average length of 181.3 frames per video. It contains 10 folds of 500 video pairs. We follow the standard verification protocol and report the average accuracy on splits with cross-validation in Table 5. We achieve the accuracy of 96.08% using  $L_2$ -softmax loss with RX101 network. Our method outperforms many recent algorithms and is only behind DeepVisage [10] which uses larger number of training samples, and VGG Face [24] which further uses a discriminative metric learning on YTF.

The IJB-A dataset [16] contains 500 subjects with a total of 25,813 images including 5,399 still images and 20,414 video frames. It contains faces with extreme viewpoints, resolution and illumination which makes it more challenging than the commonly used LFW dataset. Given a template containing multiple faces of the same individual, we gener-

Table 5. Verification accuracy (in %) of different methods on LFW and YTF datasets.

Method	Images	#nets	One loss	LFW	YTF
Deep Face [30]	4M	3	No	97.35	91.4
DeepID-2+ [29]	-	25	No	99.47	93.2
FaceNet [28]	200M	1	Yes	99.63	95.12
VGG Face [24]	2.6M	1	No	98.95	<b>97.3</b>
Baidu [19]	1.3M	1	No	99.13	-
Wen et al. [33]	0.7M	1	No	99.28	94.9
NAN [37]	3M	1	No	-	95.72
DeepVisage [10]	4.48M	1	Yes	99.62	<b>96.24</b>
SphereFace [20]	0.5M	1	Yes	99.42	95.0
softmax(FR)	3.7M	1	Yes	99.0	93.82
$L_2$ -S (FR)	3.7M	1	Yes	99.60	95.54
$L_2$ -S (R101)	3.7M	1	Yes	99.67	96.02
$L_2$ -S (RX101)	3.7M	1	Yes	<b>99.78</b>	<b>96.08</b>

ate a common vector representation by media pooling the individual face descriptors, as explained in [27]. Table 6 lists the performance of recent DCNN-based methods on IJB-A dataset. We achieve state-of-the-art result for both verification and the identification protocols. Since the  $L_2$ -softmax loss can be coupled with any other auxiliary loss, we use the Triplet Probabilistic Embedding (TPE) [27] to learn a 128-dimensional embedding using the training splits of IJB-A. It further improves the performance and achieves a record TAR of 0.909 @ FAR = 0.0001. To the best of our knowledge, we are the first ones to surpass TAR of 0.9 @ FAR of 0.0001 on IJB-A. Our method performs significantly better than existing methods in most of the other metrics as well. The results on LFW [14], YTF [34] and IJB-A [16] datasets clearly suggests the effectiveness of the proposed  $L_2$ -softmax loss.

## 6. Conclusions

In this paper, we added a simple, yet effective,  $L_2$ -constraint to the regular softmax loss for training a face verification system. The constraint enforces the features to lie on a hypersphere of a fixed radius characterized by parameter  $\alpha$ . We also provided bounds on the value of  $\alpha$  for achieving a consistent performance. Experiments on LFW, YTF and IJB-A datasets show that the proposed  $L_2$ -softmax loss provides a significant and consistent boost over the regular softmax loss and achieves the state-of-the-art result on IJB-A [16] dataset. In conclusion,  $L_2$ -softmax loss is a valuable replacement for the existing softmax loss, for the task of face verification. In the future, we would further explore the possibility of exploiting the geometric structure of the feature encoding using manifold-based metric learning.



Table 6. Face Identification and Verification Evaluation on IJB-A dataset

Method	IJB-A Verification (TAR@FAR)				IJB-A Identification			
	0.0001	0.001	0.01	0.1	FPIR=0.01	FPIR=0.1	Rank=1	Rank=10
GOTS [16]	-	0.2(0.008)	0.41(0.014)	0.63(0.023)	0.047(0.02)	0.235(0.03)	0.443(0.02)	-
B-CNN [6]	-	-	-	-	0.143(0.027)	0.341(0.032)	0.588(0.02)	-
LSFS [31]	-	0.514(0.06)	0.733(0.034)	0.895(0.013)	0.383(0.063)	0.613(0.032)	0.820(0.024)	-
VGG-Face [24]	-	0.604(0.06)	0.805(0.03)	0.937(0.01)	0.46(0.07)	0.67(0.03)	0.913(0.01)	0.981(0.005)
$DCNN_{manual}$ +metric [4]	-	-	0.787(0.043)	0.947(0.011)	-	-	0.852(0.018)	0.954(0.007)
Pose-Aware Models [21]	-	0.652(0.037)	0.826(0.018)	-	-	-	0.840(0.012)	0.946(0.007)
Chen et al. [3]	-	-	0.838(0.042)	0.967(0.009)	0.577(0.094)	0.790(0.033)	0.903(0.012)	0.977(0.007)
Deep Multi-Pose [2]	-	-	0.876	0.954	0.52	0.75	0.846	0.947
Masi et al. [22]	-	0.725	0.886	-	-	-	0.906	0.977
Triplet Embedding [27]	-	0.813(0.02)	0.90(0.01)	0.964(0.005)	0.753(0.03)	0.863(0.014)	0.932(0.01)	0.977(0.005)
Template Adaptation [8]	-	0.836(0.027)	0.939(0.013)	0.979(0.004)	0.774(0.049)	0.882(0.016)	0.928(0.01)	0.986(0.003)
All-In-One Face [25]	-	0.823(0.02)	0.922(0.01)	0.976(0.004)	0.792(0.02)	0.887(0.014)	0.947(0.008)	0.988(0.003)
NAN [37]	-	0.881(0.011)	0.941(0.008)	0.979(0.004)	0.817(0.041)	0.917(0.009)	0.958(0.005)	0.986(0.003)
TDFF [36]	0.875(0.013)	0.919(0.006)	0.961(0.007)	0.988(0.003)	0.878(0.035)	0.941(0.010)	0.964(0.006)	<b>0.992(0.002)</b>
TDFF [36]+TPE [27]	0.877(0.018)	0.921(0.005)	0.961(0.007)	<b>0.989(0.003)</b>	0.881(0.039)	0.940(0.009)	0.964(0.007)	<b>0.992(0.003)</b>
softmax (FR)	0.730(0.076)	0.851(0.021)	0.926(0.01)	0.972(0.004)	0.788(0.048)	0.892(0.015)	0.953(0.008)	0.984(0.004)
$L_2$ -S (FR)	0.832(0.027)	0.906(0.016)	0.952(0.007)	0.981(0.003)	0.852(0.042)	0.930(0.01)	0.963(0.007)	0.986(0.002)
$L_2$ -S (FR)+TPE [27]	0.863(0.012)	0.910(0.013)	0.951(0.006)	0.979(0.003)	0.873(0.024)	0.931(0.01)	0.961(0.007)	0.983(0.003)
$L_2$ -S (R101)	0.879(0.028)	0.937(0.008)	0.967(0.005)	0.984(0.002)	0.895(0.055)	0.953(0.007)	0.973(0.005)	0.987(0.003)
$L_2$ -S (R101)+TPE [27]	0.898(0.019)	0.942(0.006)	0.969(0.004)	0.983(0.003)	0.910(0.045)	<b>0.956(0.007)</b>	0.971(0.005)	0.986(0.003)
$L_2$ -S (RX101)	0.883(0.032)	0.938(0.008)	0.968(0.004)	0.987(0.002)	0.903(0.046)	0.955(0.007)	<b>0.975(0.005)</b>	0.990(0.002)
$L_2$ -S (RX101)+TPE [27]	<b>0.909(0.007)</b>	<b>0.943(0.005)</b>	<b>0.970(0.004)</b>	0.984(0.002)	<b>0.915(0.041)</b>	<b>0.956(0.006)</b>	0.973(0.005)	0.988(0.003)

## 7. ACKNOWLEDGMENTS

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.

## References

- [1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. **1, 2**
- [2] W. AbdAlmageed, Y. Wu, S. Rawls, S. Harel, T. Hassner, I. Masi, J. Choi, J. Lekust, J. Kim, P. Natarajan, et al. Face recognition using deep multi-pose representations. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016. **9**
- [3] J.-C. Chen, V. M. Patel, and R. Chellappa. Unconstrained face verification using deep cnn features. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016. **1, 2, 3, 9**
- [4] J.-C. Chen, R. Ranjan, A. Kumar, C.-H. Chen, V. M. Patel, and R. Chellappa. An end-to-end system for unconstrained face verification with deep convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 118–126, 2015. **9**
- [5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005. **2**
- [6] A. R. Chowdhury, T.-Y. Lin, S. Maji, and E. Learned-Miller. One-to-many face recognition with bilinear cnns. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016. **9**
- [7] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. **1, 2**
- [8] N. Crosswhite, J. Byrne, O. M. Parkhi, C. Stauffer, Q. Cao, and A. Zisserman. Template adaptation for face verification and identification. *arXiv preprint arXiv:1603.03958*, 2016. **2, 9**
- [9] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*, pages 87–102. Springer, 2016. **5**
- [10] A. Hasnat, J. Bohné, S. Gentric, and L. Chen. Deepvisage: Making face recognition simple yet with powerful general-

- ization skills. *arXiv preprint arXiv:1703.08388*, 2017. 2, 8
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 6
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 5, 8
- [13] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014. 2
- [14] 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, 2, 5, 6, 8
- [15] 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. 1, 2, 5
- [16] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1931–1939, 2015. 1, 2, 3, 6, 8, 9
- [17] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998. 4
- [18] W.-A. Lin, J.-C. Chen, and R. Chellappa. A proximity-aware hierarchical clustering of faces. *arXiv preprint arXiv:1703.04835*, 2017. 6
- [19] J. Liu, Y. Deng, T. Bai, Z. Wei, and C. Huang. Targeting ultimate accuracy: Face recognition via deep embedding. *arXiv preprint arXiv:1506.07310*, 2015. 2, 8
- [20] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphreface: Deep hypersphere embedding for face recognition. *arXiv preprint arXiv:1704.08063*, 2017. 2, 8
- [21] I. Masi, S. Rawls, G. Medioni, and P. Natarajan. Pose-aware face recognition in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4838–4846, 2016. 9
- [22] I. Masi, A. T. Trn, T. Hassner, J. T. Leksut, and G. Medioni. Do we really need to collect millions of faces for effective face recognition? In *European Conference on Computer Vision*, pages 579–596. Springer, 2016. 9
- [23] C. J. Parde, C. Castillo, M. Q. Hill, Y. I. Colon, S. Sankaranarayanan, J.-C. Chen, and A. J. O’Toole. Deep convolutional neural network features and the original image. *arXiv preprint arXiv:1611.01751*, 2016. 1
- [24] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, volume 1, page 6, 2015. 1, 2, 3, 8, 9
- [25] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa. An all-in-one convolutional neural network for face analysis. *arXiv preprint arXiv:1611.00851*, 2016. 3, 6, 7, 8, 9
- [26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 8
- [27] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa. Triplet probabilistic embedding for face verification and clustering. In *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on*, pages 1–8. IEEE, 2016. 1, 2, 3, 8, 9
- [28] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 1, 2, 3, 8
- [29] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2892–2900, 2015. 1, 2, 8
- [30] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014. 2, 8
- [31] D. Wang, C. Otto, and A. K. Jain. Face search at scale: 80 million gallery. *arXiv preprint arXiv:1507.07242*, 2015. 9
- [32] Y. Wen, Z. Li, and Y. Qiao. Latent factor guided convolutional neural networks for age-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4901, 2016. 1, 2
- [33] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016. 1, 2, 3, 4, 5, 6, 8
- [34] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011. 8
- [35] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 8
- [36] L. Xiong, J. Karlekar, J. Zhao, J. Feng, S. Pranata, and S. Shen. A good practice towards top performance of face recognition: Transferred deep feature fusion. *arXiv preprint arXiv:1704.00438*, 2017. 2, 9
- [37] J. Yang, P. Ren, D. Chen, F. Wen, H. Li, and G. Hua. Neural aggregation network for video face recognition. *arXiv preprint arXiv:1603.05474*, 2016. 2, 8, 9